What Can Be Computed Locally Revisited
— First-Order Logic on Sparse Graphs in Distributed Computing*—

Lélia Blin[1], Fedor V. Fomin[2], Pierre Fraigniaud[1], Sylvain Gay[1,6], Petr Golovach[2], Pedro Montealegre[3], Ivan Rapaport[4], and Ioan Todinca[5]

[1]Université Paris Cité, CNRS, IRIF, Paris, France
[2]University of Bergen, Bergen, Norway
[3]Universidad Adolfo Ibañez, Santiago, Chile
[4]DIM-CMM, Universidad de Chile, Santiago, Chile
[5]LIFO, Université d'Orléans and INSA Centre-Val de Loire, Orléans, France
[6]École Normale Supérieure, Paris, France

Abstract

The question of "what can be computed locally?" lies at the heart of *distributed computing in networks*. As established in Naor and Stockmeyer's seminal paper (STOC 1993, Edsger W. Dijkstra Prize in Distributed Computing 2025), this question is undecidable, even for graph problems whose solutions can be checked locally. In this paper, we adopt a novel perspective on the question, by asking for which classes Π of problems, and for which classes $\mathcal{G}$ of graphs, *all* problems in Π can be solved efficiently in a distributed manner in *all* graphs of $\mathcal{G}$. This paper focuses on two natural candidates for such an approach, namely the class of problems expressible in *first-order logic (FO)*, because they possess an intrinsic form of locality thanks to Gaifman's theorem, and the class of graphs with *bounded expansion*, because they form a large class of graphs encompassing, e.g., planar, bounded-genus, bounded-treewidth, and bounded-degree graphs, as well as graphs excluding a fixed minor or topological minor, sparse Erdős–Rényi graphs (a.a.s.), and several network models such as stochastic block models for suitable parameter ranges.

The starting point of our work is the decade-old open question of Nešetřil and Ossona de Mendez (Distributed Computing 2016) on the distributed complexity of *local* FO formula on graphs of bounded expansion, in the standard CONGEST model of distributed computing. Recall that a formula $\varphi(x)$ is local if the satisfaction of $\varphi(x)$ depends only on the r-neighborhood of its free variable x, for some fixed r. For instance, the formula "x belongs to a triangle" is local. We resolve the open problem of Nešetřil and Ossona de Mendez positively by showing that, for

*The research leading to these results has been supported by the Research Council of Norway via the project BWCA (314528), and by Université Paris Cité via the Emergence project METALG. F. Fomin partially supported by European Research Council (ERC) via grant NewPC, reference 101199930. P. Fraigniaud was partially supported by the ANR projects DUCAT (ANR-20-CE48-0006), PREDICTIONS (ANR-23-CE48-0010), ENEDISC (ANR-24-CE48-7768-01), and the French PEPR integrated project EPiQ (ANR-22-PETQ-0007). P. Montealegre was supported by ECOS-ANID ECOS240020, STIC-AMSUD ECODIST AMSUD240005, Centro de Modelamiento Matemático (CMM), FB210005, BASAL funds for centers of excellence from ANID-Chile, FONDECYT 1230599 and ANID-MILENIO-NCN2024_103. I. Rapaport has received funding from FB210005, BASAL funds for centers of excellence from ANID-Chile, and FONDECYT 1220142.

every local FO formula $\varphi(x)$, and for every graph class $\mathcal{G}$ of bounded expansion, there exists a deterministic algorithm that identifies, for every n-vertex graph $G \in \mathcal{G}$, all vertices v of G such that $G \models \varphi(v)$, in $\mathcal{O}(\log n)$ rounds. The requirement of locality is unavoidable, as even the simple FO formula "there exist two vertices of degree 3" requires $\Omega(D)$ rounds in CONGEST, even on trees of diameter D. Nevertheless, we establish a second result, which goes beyond the question of Nešetřil and Ossona de Mendez. We show that $\mathcal{O}(D+\log n)$ rounds are sufficient for deciding *any* FO formula φ on graphs of bounded expansion. That is, the overhead to be paid over the diameter is just $\mathcal{O}(\log n)$. We underline that the techniques behind our two distributed "meta-theorems" extend to distributed *counting*, *optimization*, and *certification* problems.

Our results are tight in several ways. Regarding the choice of the graph class $\mathcal{G}$, we show that deciding FO formulas may have high round complexity in CONGEST on larger classes of graphs, even if they remain *sparse*. For instance, the simple local FO formula expressing C_6-freeness requires $\tilde{\Omega}(\sqrt{n})$ rounds to be decided in graphs of degeneracy 2 with constant diameter. Regarding the choice of the class Π of problems, we show that deciding problems expressible in monadic second-order (MSO) logic may have high round complexity in CONGEST, even in classes of graphs with bounded expansion. For example, deciding non-3-colorability requires $\tilde{\Omega}(n)$ rounds in bounded-degree graphs with logarithmic diameter.

Keywords: Distributed computing, CONGEST model, Graph algorithms.

Contents

1 Introduction

1.1 Context and Objective

Distributed computing in networks is primarily concerned with identifying which graph problems can be efficiently solved "by the graphs themselves." That is, each vertex of the graph is assumed to be a computing device, referred to as a *node*, and nodes communicate by exchanging messages along the edges of the graph. Initially, each node knows only local information (e.g., its identifier, its degree, the weights of its incident edges if any, etc.), and it must compute its own output (e.g., whether it belongs to the computed minimum dominating set, or which of its incident edges belong to the computed minimum-weight spanning tree). Each node must do so autonomously, based only on the information obtained by exchanging data with its neighbors.

In this framework, the nodes are typically subject to two constraints: *distance* and *bandwidth*. The former merely expresses that exchanging information between two nodes at distance d in the graph requires at least d rounds of communication, for traversing a path of length d between these two nodes. The latter is motivated by the fact that network links may have limited capacity. In n-node graphs, this is usually captured by limiting communication to transmitting a single message of $\mathcal{O}(\log n)$ bit per edge in each direction at each round. To sum up, distributed computation in networks proceeds as a sequence of synchronous rounds where, at each round, every node (1) sends $\mathcal{O}(\log n)$-bit messages to each of its neighbors, (2) receives the messages sent by its neighbors, and (3) performs some individual computation. This model is commonly referred to as $\mathsf{CONGEST}$ in the literature [70]. In $\mathsf{CONGEST}$, each node is given an $\mathcal{O}(\log n)$-bit identifier, which is unique in the network.

Local and Global Problems. Two categories of graph problems can be identified with respect to their solvability in $\mathsf{CONGEST}$. *Global problems* are problems that require $\Omega(D)$ rounds to be solved in graphs of diameter D. This includes, e.g., minimum-weight spanning tree (MST), as deciding whether an edge belong to the tree may depend on the weight of another edge, at distance $\Omega(D)$. In contrast, *local problems* are problems that can be solved in polylog(n) rounds, even in graphs with diameter $\Omega(n)$.

- For a global problem, the typical objective is to express its round complexity as $\mathcal{O}(D + h(n))$ rounds, and to minimize the overhead function $h(\cdot)$. For instance, the overhead for MST is $\tilde{\mathcal{O}}(\sqrt{n})$ rounds[1] [36], and this is optimal [71]. For other examples, such as SSSP, Min-Cut, routing, min-weight cycles, etc., see, e.g., [18, 60, 63, 64], and see [15] for lower bounds.

- For local problems, the challenging question of "what can be computed locally?", to quote Naor and Stockmeyer's seminal paper [65], has been mostly addressed by considering *locally checkable labeling* (LCL) problems[2] Roughly, LCL problems are problems for which an assignment of labels to the nodes or edges is a correct solution if and only if this assignment is locally correct at each node. This is for instance the case of proper k-coloring, maximal independent set (MIS), maximal matching, etc. In other words, LCL problems are problems whose solutions can be checked at each node by inspecting its ball of constant radius around

[1] The $\tilde{\mathcal{O}}$ and $\tilde{\Omega}$ notation ignores polylog(n) factors.

[2] The paper [65] actually considered the LOCAL model [62], which is $\mathsf{CONGEST}$ without any restriction on the size of the messages. Nevertheless, since then, the question as been addressed in both LOCAL and $\mathsf{CONGEST}$ in the literature.

it. Determining which LCL problems are locally solvable (i.e., in polylog(n) rounds) has given rise of an enormous amount of work during the last two decades (see, e.g., [26, 29, 41]).

In this paper, we tackle local and global problems from a different perspective, which has the side effect of placing these two types of problems under the same umbrella whereas they are so far somehow treated separately in the literature (see above).

The Novel "Logical Approach" of Distributed Locality. Another line of research aiming at identifying which problems can be solved locally in $\mathsf{CONGEST}$ has emerged in the years 2010s [68], motivated by Gaifman's locality theorem [35]. Recall that Gaifman's theorem states that every first-order (FO) sentence is equivalent to a Boolean combination of local sentences. That is, in graphs, every FO sentence is equivalent to a sentence saying that there exist nodes that are far apart from one another in the graph, and each satisfies some local condition described by an FO formula referring to the vicinity of the node.

Considering graph problems expressible in FO logic opens a completely new approach of locality in distributed computing. Note that the class of problems expressible in FO is quite rich. In particular, it includes problems as diverse as subgraph isomorphism (induced or not), k-dominating set, k-independent set, diameter at most k, etc. For instance, triangle-freeness can be expressed as

$$\neg\Big(\exists x\,\exists y\,\exists z\ \big(\mathsf{adj}(x,y)\wedge\mathsf{adj}(y,z)\wedge\mathsf{adj}(z,x)\big)\Big), \tag{1}$$

where adj denotes the adjacency predicate. Clearly, every FO problem expressed as the combination of at least two local sentences cannot be decided in less than $\Omega(D)$ rounds in graphs of diameter D, as the two local sentences may be satisfied by far away nodes in the network. For instance, the mere problem "there exist at least two nodes of degree 3" requires $\Omega(D)$ rounds to be checked in $\mathsf{CONGEST}$, even in trees. On the other hand, FO problems expressed as a unique local sentence may or may not be decidable in polylog(n) rounds. For instance, deciding[3] whether the graph does not contain a given tree T as a subgraph (i.e., deciding T-freeness) can be done in a constant number of rounds [25]. However, deciding C_4-freeness cannot be decided in less than $\widetilde{\Omega}(\sqrt{n})$ rounds [20]. This gives rise to a new question, complementary to the aforementioned question by Naor and Stockmeyer.

Instead of asking which problems can be solved locally in *all* graphs, we focus on a broad class of problems, namely those expressible in FO, and ask for which classes of graphs *all such problems* can be solved efficiently.

More precisely, the paper addresses the following problem.

Problem 1.1. *Which graph classes $\mathcal{G}$ satisfy the following two conditions?*

- *Every* local *FO property can be decided in* $\mathcal{O}(\text{polylog}(n))$ *rounds in the* $\mathsf{CONGEST}$ *model for every n-node graph belonging to the class $\mathcal{G}$;*

[3] In distributed decision, a positive instance, i.e., an instance satisfying the predicate, must be accepted by all nodes, whereas a negative instance, i.e., an instance not satisfying the predicate, must be rejected by at least one node.

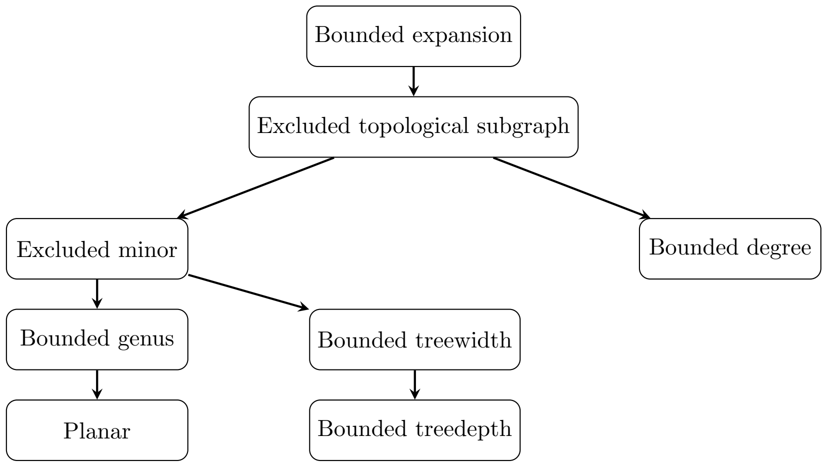

Figure 1: Different classes of graphs, with their mutual inclusion relations. An arrow from a class $\mathcal{G}$ to a class $\mathcal{G}'$ indicates that $\mathcal{G}'$ is (strictly) included in $\mathcal{G}$.

- *Every FO property can be decided in $\mathcal{O}(D + \mathrm{polylog}(n))$ rounds in the* $\mathsf{CONGEST}$ *model for every n-node graph of diameter D belonging to the class $\mathcal{G}$.*

For instance, do the above two conditions hold for the class of planar graphs? Or for random graphs (with some statistical guarantees)? Or for graphs with constant maximum degree Δ? Nešetřil and Ossona de Mendez [68] partially addressed this question in the more general setting of graph classes with *bounded expansion*, as part of the development of their *sparsity theory* [66, 67]. Their findings have important applications to the subgraph isomorphism problem.

Deciding Subgraph Isomorphism in Graphs of Bounded Expansion. Graph classes with bounded expansion form a rich graph family (see Figure 1). For instance, the class of planar graphs, the class of graphs with bounded genus, the class of graphs of constant maximum degree Δ, and the class of minor-free graphs and topologically minor-free graphs are all of bounded expansion, as well as sparse Erdős–Rényi graphs (a.a.s.). Moreover, [17] demonstrated that many network models (e.g., stochastic block models) exhibit bounded expansion for some ranges of parameters, and these findings are even supported with empirical measurements on a corpus of real-world networks. All these points make graph classes of bounded expansion quite attractive from both conceptual and practical perspectives.

In particular, the study of graphs with bounded expansion has already driven major advances in model checking [21], parameterized complexity [66], kernelization [19, 23] and approximation algorithms [51]. In the context of distributed computing, the subgraph isomorphism can be efficiently solved in graphs of bounded expansion. Specifically, the proposition below states that, for any fixed connected graph H (e.g., a triangle C_3), H-freeness can be decided in $\mathcal{O}(\log n)$ rounds in any n-node graph G picked in a class of graphs with bounded expansion. That is, if G contains H as a subgraph, at least one node rejects, otherwise all nodes accept.

Proposition 1.2 (Nešetřil and Ossona de Mendez [68])**.** *For every graph class $\mathcal{G}$ of bounded expansion, and for every connected graph H, there exists a distributed algorithm that, for every n-node graph $G \in \mathcal{G}$, decides whether G does not contain H as a subgraph in $\mathcal{O}(\log n)$ rounds in* $\mathsf{CONGEST}$*.*

Actually, the algorithm in [68] does more than just deciding H-freeness. It actually marks all nodes of the input graph G belonging to an isomorphic copy of H (these nodes are the ones that reject whenever G is not H-free). Note that it is known [8, 20] that, for some H (e.g., $H = C_{2k+1}$ for $k > 1$), deciding H-freeness requires $\widetilde{\Omega}(n)$ rounds in arbitrary graphs. Instead, for all graphs H, H-freeness can be decided in just $\mathcal{O}(\log n)$ rounds in graphs of bounded expansion, and thus in particular in planar graphs.

Intuition for Planar Graphs. The algorithm of Nešetřil and Ossona de Mendez for subgraph isomorphism [68] is important for our work, so we devote some space to highlight their approach. Their algorithm works roughly as follows. We provide the intuition for the case of planar graphs. It is known [67] that there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that, for every integer $k \geq 1$, every planar graph G can be properly colored with $f(k)$ colors so that, for every set S of at most k colors, the subgraph of G induced by the nodes colored with a color in S has *treedepth* at most $|S|$. Intuitively, treedepth measures how far a graph is from being a forest of bounded depth. (More formally, a graph has treedepth d if there exists a rooted forest F on the same set of vertices as the graph, in which all trees have depth at most d, such that every edge of G is between a node and one of its ancestors in F.) Graphs of bounded treedepth are therefore appealing from an algorithmic perspective. As a side remark, the existence of such a function f leading to colored subgraphs of bounded treedepth characterizes the graph classes of bounded expansion, but we focus on planar graphs here for concreteness.

Let us fix $k = |V(H)|$, i.e., k is the number of nodes of H. By coloring the input graph G with $f(k)$ colors, there must exist a set S of k colors such that, if H is contained in G as a subgraph, then at least one copy of H appears in the subgraph $G[S]$, where $G[S]$ denotes the subgraph of G induced by the nodes with colors in S. Without going into details, since $G[S]$ has bounded treedepth, deciding H-freeness in $G[S]$ can actually be done in $O(1)$ rounds [31]. It is therefore sufficient to test all choices of k colors among the $f(k)$ colors of the palette, and for a node to reject if it belongs to an isomorphic copy of H for one of these $\binom{f(k)}{k}$ choices. The main contribution in [68] is in fact the design of a distributed algorithm for $\mathsf{CONGEST}$ that, given any positive integer k, computes an appropriate $g(k)$-coloring of any n-node planar graph G in $\mathcal{O}(\log n)$ rounds, for some function $g : \mathbb{N} \to \mathbb{N}$. In fact, the algorithm works not only for planar graphs, but also for any graph class $\mathcal{G}$ of expansion $f : \mathbb{N} \to \mathbb{N}$. For a given k, the number of colors $g(k)$ used by the algorithm may be larger than $f(k)$, but the crucial point is that the subgraph induced by any set of k colors among the $g(k)$ colors has bounded treedepth.

Beyond Subgraph Isomorphism. At this point, it is worth noticing that H-freeness for a connected graph H is expressible by an FO formula of a very specific form: it is merely the negation of an existential formula; see, e.g., Eq. (1). Therefore, checking whether such a formula holds boils down to searching for $k = |V(H)|$ nodes satisfying the adjacency conditions specified by H. Since these k nodes must appear in one of the $\binom{f(k)}{k}$ subgraphs $G[S]$ of bounded treedepth, this is (to some extent) straightforward. However, the situation becomes immediately more challenging once universal quantifiers are introduced. For instance, consider checking the existence of *twins* in planar graphs, that is, the existence of two nodes v and v' with the same neighborhood, i.e., satisfying $N_G(v) \smallsetminus \{v'\} = N_G(v') \smallsetminus \{v\}$. The existence of twin nodes can be written as (assuming all variables refer to distinct vertices):

$$\exists x\,\exists y\,\forall z\;\Big(\big(\mathsf{adj}(x,z)\wedge \mathsf{adj}(y,z)\big)\;\vee\;\big(\neg\mathsf{adj}(x,z)\wedge\neg\mathsf{adj}(y,z)\big)\Big).$$

The formula above has only three variables, as is the case for Eq. (1). However, here the variable z is universally quantified, whereas all variables in Eq. (1) are existentially quantified. As a consequence, it is no longer true that it suffices to search for three nodes in each of the $\binom{f(3)}{3}$ subgraphs $G[S]$ of bounded treedepth in order to determine whether there are twin nodes in a planar graph G. Indeed, two twins may each have up to $\Omega(n)$ neighbors, which may not appear together in any subgraph of G induced by a constant number of colors.

Nešetřil and Ossona de Mendez [68] therefore posed a broader and more challenging question, later reiterated by Pilipczuk, Siebertz, and Toruńczyk [73], and by Fomin, Fraigniaud, Montealegre, Rapaport, and Todinca [31]. Recall that an FO formula $\varphi(x)$ is *local* if the satisfaction of $\varphi(x)$ depends only on the r-neighborhood of its free variable x, for some fixed r. For example, in graphs, the formula "node x has a twin" is local. Another example of a local formula $\varphi(x)$ is the following, for fixed positive integers d and k: "the distance-d neighborhood of x in G has a dominating set of size at most k". The question in [68] can be formulated as follows:

Question 1.3 (Nešetřil and Ossona de Mendez [68])**.** *Is there, for every local first-order (FO) formula $\varphi(x)$ and for every graph class $\mathcal{G}$ of bounded expansion, a distributed algorithm that, for every n-node graph $G\in\mathcal{G}$, marks all vertices $v\in V(G)$ such that $G\models\varphi(v)$ in $\mathcal{O}(\log n)$ rounds in the $\mathsf{CONGEST}$ model?*

Note that a positive answer to this question implies that all graph classes of bounded expansion satisfy the first item of Problem 1.1.

1.2 Our Results

Our first main result resolves Question 1.3.

Theorem 1.4. *For every local FO formula $\varphi(x)$ on graphs, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a deterministic algorithm that, for every n-node network $G\in\mathcal{G}$, marks all vertices v of G satisfying $G\models\varphi(v)$, in $\mathcal{O}(\log n)$ rounds in the $\mathsf{CONGEST}$ model.*

In fact, the result above also holds for FO formulas on *labeled* graphs, i.e., graphs in which every node is provided with labels drawn from a finite set. In other words, formulas may also include *unary* predicates on node labels (see Theorem 4.1 for the formal statement). Before summarizing the techniques used to prove Theorem 1.4, it is worth mentioning that we extend this result in several directions.

1.2.1 Extension 1: Deciding FO formulas

Our second result concern general FO formulas, for which the dependence on the diameter D is unavoidable. Again, the result below holds for labeled graphs too. For the sake of clarity, we state it here for unlabeled graphs only (see Theorem 5.1 for the more general statement).

Theorem 1.5. *For every FO formula φ on graphs, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a distributed algorithm that, for every n-node network $G\in\mathcal{G}$ of diameter D, decides whether $G\models\varphi$ in $\mathcal{O}(D+\log n)$ rounds under the $\mathsf{CONGEST}$ model.*

Note that bounded degeneracy does not suffice to establish the same result as above[4]. Indeed, we show that the simple FO formula expressing C_6-freeness requires $\tilde{\Omega}(\sqrt{n})$ rounds to be checked in $\mathsf{CONGEST}$, even in graphs of degeneracy 2 and constant diameter (see Theorem 8.4). Moreover, our result is optimal with respect to the class of formulas, in the sense that it cannot be extended to, e.g., MSO. In particular, we show that non-3-colorability requires $\tilde{\Omega}(n)$ rounds to be checked in bounded-degree graphs with logarithmic diameter (see Theorem 8.2).

Take Away Message. Theorems 1.4 and 1.5 state that there are large classes of graphs satisfying the two conditions of Problem 1.1, namely, the graph classes with bounded expansion.

In the $\mathsf{CONGEST}$ model, for every graph class $\mathcal{G}$ of bounded expansion,

- every local FO property can be decided in $\mathcal{O}(\log n)$ rounds, and
- every FO property can be decided in $\mathcal{O}(D + \log n)$ rounds.

While the tight connections between problems expressible in FO logic and graph classes of bounded expansion in the context of sequential model checking are well established and well understood (see Section 1.4), prior to our work this was not the case for distributed computing. In the latter setting, our results demonstrate that:

1. Locality of FO formulas implies locality of distributed decision, within just $\mathcal{O}(\log n)$ rounds;
2. Gaifman's locality theorem for FO formulas implies only a small $\mathcal{O}(\log n)$-round overhead beyond the unavoidable round complexity D.

Moreover, our contribution goes even beyond distributed decision, as detailed next.

1.2.2 Extension 2: Counting, Optimization, and Certification

Counting Solutions. We consider the mere problem of *counting* solutions of FO formulas with free variables. We actually consider two forms of counting:

- *Local counting*: Given a local formula $\varphi(x_1, \dots, x_k)$ with $k \geq 1$ free variables (e.g., $\mathsf{adj}(x_1, x_2) \wedge \mathsf{adj}(x_2, x_3) \wedge \mathsf{adj}(x_3, x_1)$, i.e., x_1, x_2, x_3 form a triangle), each node v of the considered graph G outputs a (possibly negative) integer $\nu(v)$ satisfying that the number of tuples $v_1, \dots, v_k$ for which $G \models \varphi(v_1, \dots, v_k)$ equals $\sum_{v \in V} \nu(v)$.
- *Global counting*: Given a general formula $\varphi(x_1, \dots, x_k)$ with $k \geq 1$ free variables, all nodes output a same value equal to the number of tuples $v_1, \dots, v_k$ for which $G \models \varphi(v_1, \dots, v_k)$.

The next theorem extends Theorems 1.4 and 1.5 to local and global counting. Again, the results below also hold for labeled graphs (cf. Theorem 6.1).

Theorem 1.6. *For every FO formula $\varphi(x_1, \dots, x_k)$ on graphs with $k \geq 1$ free variables $x_1, \dots, x_k$, and for every class of graphs $\mathcal{G}$ of bounded expansion:*

[4] Any class of graphs with bounded expansion has bounded degeneracy, but the reciprocal does not hold. For instance, the class of cliques with all edges subdivided once has degeneracy 2, but is not of bounded expansion.

- *There exists a distributed algorithm that, for every n-node network $G \in \mathcal{G}$ of diameter D, performs global counting in $O(D + \log n)$ rounds in CONGEST;*
- *If the formula $\varphi(x_1, \ldots, x_k)$ is local, then there exists a distributed algorithm that performs local counting in $\mathcal{O}(\log n)$ rounds in CONGEST.*

Note that, in contrast to our Theorem 1.6, the best known algorithm for (locally) counting triangles in general graphs takes $\tilde{\mathcal{O}}(n^{1/3})$ rounds [10].

Optimization Problems. Some optimization problems can also be expressed as finding optimal solutions for FO formulas with free variables, over weighted graphs, e.g., finding a maximum-weight triangle. (If no triangles exist in the graph, then all nodes reject.) Note that, as usual in the literature on CONGEST [70], the weights assigned to the nodes or edges are assumed to be polynomial in the network size so that any weight can be transmitted between neighbors in a single round of communication. Our next theorem extends Theorem 1.5 to weighted problems. For the version of labeled graphs, see Theorem 6.2.

Theorem 1.7. *For every FO formula $\varphi(x_1, \ldots, x_k)$ on graphs with $k \geq 1$ free variables $x_1, \ldots, x_k$, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a distributed algorithm that, for every n-node network $G = (V, E) \in \mathcal{G}$ of diameter D, and every weight function $\omega : V \to \mathbb{N}$, computes a k-tuple of vertices $(v_1, \ldots, v_k)$ such that*

$$G \models \varphi(v_1, \ldots, v_k), \quad \textit{and} \quad \sum_{i=1}^{k} \omega(v_i) \textit{ is maximum,}$$

in $\mathcal{O}(D + \log n)$ rounds under the CONGEST model. (The same holds if replacing maximum by minimum.) If no tuples $(v_1, \ldots, v_k)$ exist such that $G \models \varphi(v_1, \ldots, v_k)$, then all nodes reject.

Theorem 1.7 allows to find, for example, a monochromatic triangle of minimum weight, or a multi-colored k-independent set of maximum weight, in a class $\mathcal{G}$ of c-colored graphs of bounded expansion.

Distributed Certification. Last but not least, our results extend to distributed certification. Roughly, a *distributed certification scheme* [44, 56] for a graph property φ is a pair prover-verifier. The prover is a centralized, computationally unlimited oracle that assigns *certificates* to the nodes. The verifier is a 1-round distributed algorithm whose role is to check whether the certificates assigned to the nodes form a proof that the graph satisfies the property. A distributed certification scheme is correct if it satisfies:

Completeness: if $G \models \varphi$, then the prover can assign certificates such that the verifier accepts at all nodes;

Soundness: if $G \not\models \varphi$, then, no matter the certificates assigned to the nodes, the verifier rejects in at least one node.

The main complexity measure is the *size* of the certificates assigned by the prover on legal instances. In recent years, a vibrant and rapidly evolving line of research has emerged around algorithmic meta-theorems for distributed certification. It began with the seminal result by Bousquet, Feuilloley and Pierron [7] showing that any MSO property can be certified with $O(\log n)$-bit certificates on graphs of bounded treedepth. This was then extended in [33] to bounded treewidth graphs, where all MSO properties admit $O(\log^2 n)$-bit certificates. Further extensions covered dense graph classes of bounded clique-width for MSO_1 properties [32]. More recently, Baterisna and Chang [2] achieved $O(\log n)$-bit certificates on graphs of bounded pathwidth for MSO_2 properties, and Cook, Kim and Masařík [11] proved that on graphs of bounded treewidth, every MSO_2 property admits $O(\log n)$-bit certification.

Our work contributes to this active line of research by significantly broadening its scope. We show that every first-order (FO) property can be certified with $O(\log n)$-bit certificates on all graphs of bounded expansion. As in the previous theorems, the theorem below extends to labeled graphs (cf. Theorem 7.1).

Theorem 1.8. *For every FO formula φ on graphs, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a distributed certification scheme that, for every n-node network $G \in \mathcal{G}$, certifies $G \models \varphi$ using certificates on $\mathcal{O}(\log n)$ bits.*

Note that, with $O(\log n)$-bit certificates, the verifier can be implemented in a single round of communication in the CONGEST model, as desired. Note also that our certification scheme is actually a *proof-labeling scheme* [56], which is a more demanding form of distributed certification than, e.g., *locally checkable proofs* [44]. As a direct consequence of Theorem 1.8, certifying C_4-freeness can be done with $\mathcal{O}(\log n)$-bit certificates in graphs of bounded expansion. In contrast, certifying C_4-freeness requires certificates on $\tilde{\Omega}(\sqrt{n})$ bits in general graphs, using the reduction from [20]. More generally, [27] shows that every MSO formula can be *certified* with $O(\log n)$-bit certificates in graphs of bounded treedepth. Theorem 1.8 extends this result. Indeed, while Theorem 1.8 is for FO formulas, and the result of [27] is for MSO, both logics have identical expressive power on graphs of bounded treedepth. Last but not least, Theorem 1.8 is the best that can be achieved. Indeed, we show that it does not extend to MSO, as certifying non-3-colorability requires certificates of $\tilde{\Omega}(\sqrt{n})$ bits in planar graphs (see Theorem 8.3), and even certificates of $\tilde{\Theta}(n)$ bits in bounded degree graphs (see Theorem 8.2).

Remark. The open problem by Nešetřil and Ossona de Mendez was actually stated for the BROADCAST CONGEST model, i.e., a weaker version of the CONGEST model in which, at every round, each node v must send the *same* $\mathcal{O}(\log n)$-bit message to all its neighbors (see, e.g., [20, 55]). However, (1) the problem is not necessarily easier to solve in the general version of the CONGEST model, and (2) we solve it anyway by the affirmative using an algorithm designed for the BROADCAST CONGEST model, and the complexity of our algorithm is expressed under the latter. For the sake of simplifying the notations, we simply refer to CONGEST in the statements of our theorems, but all our upper bound results (except Theorem 1.7) hold under the BROADCAST CONGEST model too[5].

1.3 Our Techniques

In this subsection, we provide a general overview of our techniques, postponing a more detailed description to Section 2. In a nutshell, our technical contributions cover several aspect of distributed computing and logic:

[5]The main reason why Theorem 6.2 does not hold in BROADCAST CONGEST is that our algorithm involves an information dissemination phase going downward a tree, in which different messages are sent to different children.

- Distributed implementation of the quantifier-elimination method.
- A new "locality-preserving technique" for quantifier-elimination.
- A new "controlled quantifier-elimination technique" that enables extending quantifier-elimination to both local and global counting, and solving these problems in a distributed manner.

Let us first clarify why the approach by Nešetřil and Ossona de Mendez [68] for marking all nodes v satisfying $G \models \varphi(v)$ fails beyond the case of local *and* existential FO formula, and how we address the challenge of going further than this specific type of formulas. Given a local *existential* FO formula

$$\varphi = \exists x_1 \ldots \exists x_p \, \psi(x_1, \ldots, x_p),$$

where ψ is quantifier-free, the fact that the input graph G belongs to a graph class $\mathcal{G}$ of expansion $f : \mathbb{N} \to \mathbb{N}$ can be exploited somehow "directly" for checking whether $G \models \varphi$. Indeed, this checking is achieved by:

1. Computing a coloring $c : V(G) \to \{1, \ldots, f(p)\}$ satisfying that, for every set $U \subseteq \{1, \ldots, f(p)\}$ of p colors, the subgraph $G[U]$ induced by the nodes of G colored with colors in U has treedepth p (such a coloring can be computed in $\mathcal{O}(\log n)$ rounds in $\mathsf{CONGEST}$ [68]).
2. Observing that $G \models \varphi$ if and only if there exists such a set U such that $G[U] \models \varphi$, it is sufficient to test all sets $U \subseteq \{1, \ldots, f(p)\}$ of cardinality p, and, for each of them, to check whether $G[U] \models \varphi$. The latter can be performed in $\mathcal{O}(1)$ rounds as $G[U]$ has bounded treedepth [31].

This approach works but under two conditions only. First, the formula φ must be *local*. Second, it must be *existential*. Instead, in this paper, we address the general case of deciding whether $G \models \varphi$ where φ is general, and in particular not necessarily local. Moreover, for local formulas, we address the case of marking all nodes v satisfying $G \models \varphi(v)$ for general local formulas, not necessarily existential. To do so, our first contribution is a distributed algorithm for quantifier-elimination, which is challenging as it must run under the $\mathsf{CONGEST}$ model, in which only adjacent nodes can exchange messages, and where the amount of information two adjacent nodes can exchange is limited.

Distributed quantifier-elimination. The goal is to transform φ into an equivalent *quantifier-free* formula $\widehat{\varphi}$. That is, $\widehat{\varphi}$ is quantifier-free, and it satisfies $G \models \varphi$ if and only if $\widehat{G} \models \widehat{\varphi}$ where $\widehat{G}$ is the same graph as G but with nodes labeled with labels taken from a finite set. One first contribution of this paper is to prove that quantifier elimination can be performed in $\mathcal{O}(D)$ rounds in graphs of bounded expansion and diameter D, using the framework developed in [73] for parameterized circuit complexity. Computing $\widehat{\varphi}$ can be done by every node without communication. The challenge is to assign the new labels to the nodes. We show that this is doable in $\mathcal{O}(D)$ rounds in $\mathsf{CONGEST}$.

In essence, the reassignment of labels is performed by going from graphs of bounded expansion to graphs of bounded treedepth (thanks to the coloring $c : V(G) \to \{1, \ldots, f(p)\}$ where p is the number of variables of φ), then from graphs of bounded treedepth to so-called *skeletons* (i.e., collections of forests), and then from skeletons to rooted trees of bounded depth. One quantifier can be eliminated at this level, thanks to assigning new labels to each node. Handling the next quantifier requires to return back to the level of graphs with bounded treedepth. This back-and-forth process must be executed a constant number of times, for eliminating each quantifier one by one. The main difficulty here is to execute this protocol, which proceeds at different levels of abstraction, while insuring that all communications occur between neighbors in the graphs, and that the exchanged messages are of limited size. The exact number of executions depends on the formula φ only, and each execution is performed in $\mathcal{O}(D)$ rounds in graphs of diameter D.

Overall, our distributed quantifier-elimination algorithm consumes $\mathcal{O}(D+\log n)$ rounds in total, where $\mathcal{O}(\log n)$ rounds are used for coloring the graphs with $f(p)$ colors, and $\mathcal{O}(D)$ rounds are consumed for eliminating the quantifiers, and eventually evaluating the formula on the resulting labeled graph.

Local FO formulas. For *local* FO formula $\varphi(x)$ with one free variable, marking all nodes v of the (possibly labeled) graph satisfying the formula, i.e., for which $G \models \varphi(v)$ holds, is also performed via quantifier elimination. However, quantifier elimination faces an obstacle when it comes to eliminating universal quantifiers in subformulas of the form $\forall y\, \psi(x, y, z_1, \dots, z_q)$. A basic example is marking all nodes having a twin. Indeed, quantifier elimination proceeds in a standard way by rewriting the formula using only existential quantifiers, and negations. E.g., the formula is rewritten as $\neg\exists y\, \neg\psi(x, y, z_1, \dots, z_q)$. As a result, locality is lost. For instance,

$$\textsc{Twin}(x) = \exists y\, \forall z\, \big(\mathsf{adj}(x, z) \Leftrightarrow \mathsf{adj}(y, z)\big)$$

can be rewritten as $\textsc{Twin}(x) = \exists y\, \neg\textsc{NoTwins}(x, y)$, with

$$\textsc{NoTwins}(x, y) = \exists z\, \big[(\mathsf{adj}(x, z) \wedge \neg\mathsf{adj}(y, z)) \vee (\neg\mathsf{adj}(x, z) \wedge \mathsf{adj}(y, z))\big].$$

The formula $\textsc{NoTwins}(x, y)$ is however not local around x, even with x fixed, as there may exist y and z at arbitrarily large distance from x such that $\textsc{NoTwins}(x, y)$ holds. As a consequence, performing quantifier elimination blindly incurs an overhead of $\Omega(D)$ rounds. This is not an issue for general formulas, for which a round-complexity $\Omega(D)$ cannot be overcame, but this is a severe problem for local formulas.

The main challenge here is to show how to perform quantifier-elimination in a distributed manner while preserving locality. To prevent dependencies between distant vertices introduced during quantifier elimination, we adopt a redundant *relativization* of quantifiers. Instead of ranging over the entire structure, each quantified variable is evaluated only within a bounded region. This requires to generalize locality with respect to more than one variable, and to revisit the entire quantifier elimination process. We show that our *local* quantifier-elimination process can be implemented in $\mathcal{O}(\log n)$ rounds in $\mathsf{CONGEST}$, which allows us to eventually solve the open problem by Nešetřil and Ossona de Mendez (cf. Question 1.3).

Controlled quantifier-elimination for counting. Quantifier elimination is not sufficient in itself, even in its local form, when it comes to solving counting problems. Let us explain why. In the context of counting, quantifier elimination is applied to a formula $\varphi(\overline{x})$ that involves several free variables, $\overline{x} = (x_1, \dots, x_k)$, $k \geq 1$. This produces an equivalent quantifier-free formula $\widehat{\varphi}(\overline{x})$ defined over a *skeleton* structure (i.e., a collection of rooted forests) that captures all bounded–distance relations among the vertices of $\overline{x}$. Indeed, quantifier-elimination actually occurs at the level of the skeleton. However, counting operates at the level of labeled graphs. Therefore, $\widehat{\varphi}$ must be translated back to the world of graphs, which requires to overcome many obstacles. In particular, formulas in skeletons use predicates such as $\mathsf{lca}_i(x, y)$, expressing "the common ancestor of x and y is at depth i" (of some tree in a forest). Reformulating such predicates with adjacency and equality predicates can be done thanks to an existential formula expressing, e.g., "there exist $w_1, \ldots, w_q$ forming a path between x and y such that one of the w's is at depth i". The issue is that when one performs counting in the resulting formula $\exists \overline{w}\, \psi(\overline{x}, \overline{w})$, all the variables are involved, including the newly introduced variable $\overline{w} = (w_1, \ldots, w_q)$. If these $w_1, \ldots, w_q$ were unique, we would be fine, but this is not necessarily the case. Controlling uniqueness is one of the challenges that we had to address.

For limiting the effect of new variables introduced in the process of quantifier-elimination, we design a new mechanism referred to as *controlled* quantifier-elimination. The main idea is to use the uniqueness of a node's parent in each tree of the skeleton to enforce the uniqueness of each w_i (given fixed $x_1, \ldots, x_k$). To that end, we encode the structure of the skeleton into additional labels for each node of the graph. Then, we rewrite our existential formula $\exists \overline{w}\, \psi(\overline{x}, \overline{w})$ to make sure that each w_i is the parent of some w_j, $j \neq i$, or of one of the x_js in some tree of the skeleton. This guarantees that, for each tuple $\overline{x}$ satisfying $\exists \overline{w}\, \psi(\overline{x}, \overline{w})$, there exists a *unique* tuple $\overline{w}$ such that $\psi(\overline{x}, \overline{w})$ holds.

Note that, even for quantifier-free formulas, another important issue must be addressed. When testing all subset U of colors, i.e., all subgraphs $G[U]$ of bounded treedepth, the same tuple $\overline{v}$ may appear in several subsets U as it may not be the case that the vertices $v_1, \ldots, v_k$ instantiating the variables $x_1, \ldots, x_k$ are colored with distinct colors. This causes over counting. For avoiding over counting, we use the inclusion–exclusion principle over the family of color sets. Finally, since each subgraph $G[U]$ has bounded treedepth, the existential formula can be evaluated by performing dynamic programming over a rooted forest decomposition of bounded depth, which takes $\mathcal{O}(1)$ rounds [31].

Optimization and Certification. For optimization problems, we follow the same guideline as the one sketched above, where every vertex v has a weight $w(v)$ provided as input. Dynamic programming can be used to compute the optimal value for every subset U of colors, and the global optimum is the maximum (or minimum) over all sets U. For each U, the maximum (or minimum) value can be computed by aggregating the values of all connected components (as $G[U]$ may not be connected) in $\mathcal{O}(D)$ rounds thanks to a breadth-first search (BFS) tree.

For certification we let the prover provide each vertex with the transcript of the execution of quantifier elimination as certificate. This includes, among other parameters, colors, forest membership, labels, etc. The verifier checks consistency of the transcript using $\mathcal{O}(\log n)$-bit certificates for bounded-treedepth graphs [7], and it verifies any residual global aggregation along a certified BFS tree [56].

1.4 Related Work

The results of our paper lie at the intersection of two research directions. The first is the study of distributed computing and distributed certification schemes on sparse networks, such as planar graphs, or graphs excluding a fixed minor, for fundamental graph optimization problems. The second is the development of meta-theorems for sparse graph classes in centralized computation.

1.4.1 Distributed Computing on Sparse Networks

The class of graphs with bounded expansion was introduced by Nešetřil and Ossona de Mendez as part of the sparsity theory they developed [67]. One of the major research areas in distributed computing concerns the study of classical combinatorial problems on sparse networks, including planar graphs, bounded-degree graphs, graphs of bounded treewidth, and graphs excluding a fixed minor. In this context, a central goal is to design distributed algorithms that exploit structural sparsity to achieve improved efficiency—compared to general networks—particularly in terms of the number of communication rounds. Typical problems studied in this context include coloring, matching, dominating sets, cuts, spanning trees, and network decompositions. Research directions focus on understanding how sparsity affects the complexity of these problems in standard distributed models such as LOCAL and CONGEST, and on designing algorithms with optimal or near-optimal round complexity. This line of work often builds on structural properties of sparse networks, revealing deep and fruitful connections between structural graph theory and distributed computation.

Examples of work in this direction include distributed algorithms for maximal matchings, maximal independent sets (MIS) and graph colorings in bounded-degree graphs [69]; depth-first search (DFS) in planar graphs [42]; minimum spanning tree (MST) and min-cut in planar, bounded genus, and H-minor-free graphs [38, 39, 40, 48, 49, 50]; minimum dominating set on planar and H-minor-free graphs [1, 4, 13, 14, 61], and MIS for graphs excluding a fixed minor [9].

The $\tilde{\mathcal{O}}(D)$ round complexity we achieved in Theorem 1.5, is considered a "gold standard" for algorithms on sparse graphs in the CONGEST model. This complexity has been obtained, for instance, for MST on planar graphs and, more generally, on H-minor-free graphs [38, 39, 40, 48, 49, 50]. Algorithms with $\tilde{\mathcal{O}}(D)$ complexity in the CONGEST model have also been developed for bounded-treewidth graphs for a variety of problems, including matching, directed single-source shortest paths, bipartite unweighted maximum matching, and girth [54].

We are not aware of any meta-theorem in the CONGEST model that combines both logical and combinatorial constraints, with one exception. The paper [31] shows that every MSO formula can be decided in a constant number of rounds on graphs of bounded treedepth. The results of [31] are incomparable to Theorems 1.4 and 1.5. Indeed, Theorems 1.4 and 1.5 apply to graph classes that are significantly more general than graphs of bounded treedepth, whereas the number of rounds in the algorithm of [31] is completely independent of the number of vertices. We incorporate some of the methods developed in [31] into our work.

Regarding distributed algorithms on graphs of bounded expansion, the foundational work in this area is due to Nešetřil and Ossona de Mendez [68]. In the CONGEST model, they presented a distributed algorithm that computes a low-treedepth decomposition of graphs of bounded expansion in $O(\log n)$ rounds. We use this construction as a subroutine in our algorithms. In the same paper, Nešetřil and Ossona de Mendez also used this decomposition to design an algorithm for checking whether a graph of bounded expansion contains a given *connected* subgraph H of constant size, also in $O(\log n)$ rounds. Theorem 1.4 significantly extends this result.

Approximation algorithms for the minimum dominating set problem on graphs of bounded expansion in the LOCAL model were developed in [1, 52, 58]. In [76], Siebertz and Vigny study the complexity of the model-checking problem for first-order logic on graphs of bounded expansion, in the CONGESTED CLIQUE model.

Distributed certification is a very active area of research; see [28] for a survey. Several meta-theorems are known in this setting. A significant result in the field was achieved in [27], which proved that any MSO property can be locally *certified* on graphs of bounded treedepth using only a logarithmic number of bits per node, a result widely regarded as a gold standard in certification. This theorem has numerous implications; for more details we refer to [27]. In particular, the FO property of C_4-freeness and the MSO property of non-3-colorability, which in general graphs require polynomial-size certificates, can be certified using just $\mathcal{O}(\log n)$ bits per node on bounded treedepth graphs. The result by Bousquet et al. has since been extended to broader graph classes, including graphs excluding a small minor [7], graphs of bounded *treewidth* [2, 11, 33], and graphs of bounded *cliquewidth* [32].

1.4.2 Centralized Meta-Theorems

Meta-theorems, according to Grohe and Kreutzer [46], are results stating that certain model-checking problems for formulas of a given logic are tractable (in some sense) on specific classes of graphs. One of the most celebrated examples of a meta-theorem is Courcelle's theorem, which asserts that graph properties definable in MSO are decidable in linear time on graphs of bounded treewidth [12]. There is a long line of meta-theorems for FO logic on sparse graph classes in the centralized model of computation. The starting point is Seese's theorem [75], which establishes the linear-time decidability of FO properties on graphs of bounded degree. Fricke and Grohe [34] presented linear-time algorithms for planar graphs and all apex-minor-free graph classes, as well as $\mathcal{O}(n^{1+\varepsilon})$ algorithms for graphs of bounded local treewidth. Flum and Grohe [30] proved that deciding FO properties is fixed-parameter tractable on graph classes with excluded minors. The next step, to classes of graphs locally excluding a minor was achieved by Dawar, Grohe, and Kreutzer [16].

The sparsity theory of [67] opened new horizons for model checking on sparse graphs. In [21] it was proved that FO properties can be decided in linear time on graph classes of bounded expansion. Further extensions of these results are known for nowhere-dense graphs [47] and graphs of bounded twinwidth [5]. See [57] and [45] for surveys.

From a technical perspective, the most important aspect for our work is the quantifier elimination procedure introduced in [21] and [46], as well as in [73]. Similar to these works, we apply the idea of modifying the structure by adding new unary relations and functions without changing the Gaifman graph.

2 High Level Description of the Proofs

Let us first focus on sketching the proof of Theorem 1.4.

2.1 Deciding Local FO Formulas Locally

We first recall structural tools that allow us to work with bounded-expansion graphs in the distributed setting. These lead naturally to *low-treedepth colorings*, which will be the foundation for the construction of skeletons and for the distributed quantifier-elimination method developed later.

Low-treedepth colorings. We recall the notion of *low-treedepth colorings* introduced by Nešetřil and Ossona de Mendez. Given a graph G and an integer $p \in \mathbb{N}$, a proper coloring $c : V(G) \to \mathbb{N}$ is a *p-treedepth coloring* if, for every set $U \subseteq \mathbb{N}$ with $|U| \leq p$, the subgraph of G induced by the vertices $\{u \in V(G) : c(u) \in U\}$ has tree-depth at most $|U|$. The minimum number of colors required for such a coloring is denoted by $\chi_p(G)$. A graph class $\mathcal{G}$ has *bounded expansion* if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that, for every $G \in \mathcal{G}$ and every $p \in \mathbb{N}$, $\chi_p(G) \leq f(p)$, i.e., every $G \in \mathcal{G}$ admits a p-treedepth coloring using at most $f(p)$ colors.

For the rest of the section, let us fix a graph class $\mathcal{G}$ of expansion bounded by $f : \mathbb{N} \to \mathbb{N}$. We denote by $\mathsf{Col}(p)$ the family of sets $U \subset [1, f(p)]$ with cardinality at most p.

Subgraph detection via treedepth colorings. Let H be a connected graph on p vertices, and let us first assume we are given a p-treedepth coloring of a graph G. To detect whether G contains H as a subgraph, we can then check, for every $U \in \mathsf{Col}(p)$, whether H appears in $G[U]$, i.e., in the subgraph of G induced by the nodes with colors in U. If G contains a copy of H, then there must exist a color set U such that $G[U]$ contains H. Moreover, as H is connected, H lies in a single component of $G[U]$. Since $G[U]$ has bounded treedepth, this check can be done efficiently in the $\mathsf{CONGEST}$ model [31]. Nešetřil and Ossona de Mendez [68] showed that, for every bounded-expansion class $\mathcal{G}$ with expansion function f, there is a $\mathsf{CONGEST}$ algorithm that computes a p-treedepth coloring with $f(p)$ colors, in $O(\log n)$ rounds. It follows that there exists a $\mathsf{CONGEST}$ algorithm that marks every vertex belonging to a copy of H in G, in $\mathcal{O}(\log n)$ rounds [68].

The reasoning above extends to every property expressible by an *existential* first-order formula ($\exists$FO) that is local, i.e., a formula of the form

$$\varphi(x) = \exists y_1 \dots \exists y_k \; \psi(x, y_1, \dots, y_k),$$

where (1) ψ is quantifier-free with $k{+}1$ free variables, and (2) the satisfiability of $\varphi(u)$ depends only on the structure of a ball of constant radius around vertex u in the graph. As in subgraph detection, a solution is determined by $k{+}1$ vertices, and thus $k{+}1$ colors. Given a $(k{+}1)$-treedepth coloring, it suffices to search within connected subgraphs of bounded treedepth. It is known [31] that any FO property can be decided in $O(1)$ rounds on graphs of bounded treedepth in $\mathsf{CONGEST}$. Therefore, as for subgraph detection, combining [68] and [31] yields an $O(\log n)$-round $\mathsf{CONGEST}$ algorithm for any local existential FO property.

Beyond existential FO properties. For general local FO formulas, with both existential and universal quantifiers, the situation is more intricate. Indeed, the solution may no longer be determined by a fixed set of vertices, and thus restricting the search to a bounded-treedepth subgraph may not be sufficient. Let us revisit the aforementioned twin problem, with free variable x, asking whether there is a node y with exactly the same neighborhood as x. That is, let us consider the formula

$$\textsc{Twin}(x) = \exists y \, \forall z \, \big(\mathsf{adj}(x,z) \Leftrightarrow \mathsf{adj}(y,z)\big),$$

where it is assumed that x, y, and z refer to different vertices (for the sake of clarifying the presentation, we omit these constraints in the definition of $\textsc{Twin}(x)$). Since the neighborhood of a vertex may have $\Omega(n)$ nodes, not all neighbors z of x and y may lie in a region of bounded treedepth resulting from a low-treedepth coloring. $\textsc{Twin}$ is however local as it suffices to inspect the ball of radius at most 3 around vertex u to decide $\textsc{Twin}(u)$. Yet, the logical structure of $\textsc{Twin}$, which includes the variable z quantified by a universal quantifier, prevents a direct application of the technique applicable to local existential FO formula. To proceed further, let us rewrite $\textsc{Twin}$ using De Morgan's laws:

$$\begin{aligned}\textsc{Twin}(u) &= \exists v \, \neg\exists w \, \neg\big(\mathsf{adj}(u,w) \Leftrightarrow \mathsf{adj}(v,w)\big)\\ &= \exists v \, \neg\exists w \, \Big[\big(\mathsf{adj}(u,w) \wedge \neg\mathsf{adj}(v,w)\big) \vee \big(\neg\mathsf{adj}(u,w) \wedge \mathsf{adj}(v,w)\big)\Big].\end{aligned}$$

Note that the negation between the first and second quantifier still prevents us from using the reasoning for local ∃FO. Let us thus focus on the inner existential part of $\textsc{Twin}(u)$, denoted by

$$\textsc{NoTwins}(u,v) = \exists w \ \big[\big(\mathsf{adj}(u,w) \wedge \neg\mathsf{adj}(v,w)\big) \vee \big(\neg\mathsf{adj}(u,w) \wedge \mathsf{adj}(v,w)\big)\big].$$

The trick is to apply *quantifier elimination*, that is, removing the existential quantifier from $\textsc{NoTwins}$, as explained hereafter.

Quantifier elimination. From this point on, we consider *labeled graphs* (G, ℓ), where each vertex u carries a set of labels $\ell(u)$ where the labels are taken from a set Λ of constant size. We now consider labeled graphs because quantifier-elimination induces labeling the nodes. First-order formulas over (G, ℓ) use adjacency, equality, and *unary label predicates* $\mathsf{lab}_\lambda(x)$, which hold whenever $\lambda \in \ell(x)$. A first-order formula over unlabeled graphs can thus be regarded as a particular case without unary predicates.

Quantifier elimination removes quantifiers while preserving equivalence within the class of structures under consideration. This technique is at the core of recent advances in algorithmic model checking on sparse graphs [21, 46, 73]. Let φ be an existential FO formula with free variables $x_1, \ldots, x_k$, and quantified variables $y_1, \ldots, y_t$, i.e.,

$$\varphi(x_1, \ldots, x_k) = \exists y_1 \ \ldots \ \exists y_t \, \psi(x_1, \ldots, x_k, y_1, \ldots, y_t),$$

where ψ is quantifier-free. Quantifier elimination maps φ into a quantifier-free FO formula $\widehat{\varphi}$ over an *enriched structure*, which, in our setting, will later correspond to the so-called *skeletons* [73].

Skeletons. Given an integer p, which corresponds to the size of the sets $U \in \mathsf{Col}(p)$, a p-skeleton is a collection $\{F_1, \ldots, F_s\}$ of labeled rooted forests, each of depth at most d, where both s and d depend only on p. All forests share the same vertex set V, corresponding to that of the underlying labeled graph (G, ℓ). Each vertex $u \in V$ retains its set of labels $\ell(u) \subseteq \Lambda$, while the structural relations between vertices are captured by the following predicates:

- *Label predicates* $\mathsf{lab}_\lambda(x)$, already defined to hold whenever $\lambda \in \ell(x)$.
- *Least-common-ancestor (lca) predicates* $\mathsf{lca}^i_j(x, y)$, which hold when x and y belong to the same tree of the i-th forest F_i, and their least common ancestor $\mathrm{lca}(x, y)$ lies at depth j in F_i. The predicate $\mathsf{lca}^i_{-1}(x, y)$ holds when x and y lie in distinct trees of F_i.

The sets of label predicates, and lca predicates are both finite, depending only on p and $|\Lambda|$, where Λ is the set of original labels. An FO formula over p-skeletons uses *only* these two kinds of predicates. In particular, it does *not* use adjacency predicates. It was proved in [73] that, for any graph class $\mathcal{G}$ of bounded expansion, any existential formula

$$\varphi(x_1, \ldots, x_k) = \exists y_1 \ \ldots \ \exists y_t \, \psi(x_1, \ldots, x_k, y_1, \ldots, y_t)$$

and any $G \in \mathcal{G}$, there exist a *quantifier-free* FO formula $\widehat{\varphi}$ on skeletons and a $(t+k)$-skeleton $\widehat{S}(G)$ such that, for every k-tuple $(v_1, \ldots, v_k)$ of vertices of G,

$$(G, \ell) \models \varphi(v_1, \ldots, v_k) \iff \widehat{S}(G) \models \widehat{\varphi}(v_1, \ldots, v_k).$$

A central result (Lemma 5.5) in our paper further shows that, for local existential FO formulas, the skeleton $\widehat{S}(G)$ can be constructed in a distributed manner in $\mathcal{O}(\log n)$ rounds in the $\mathsf{CONGEST}$ model.

Before performing quantifier elimination, we first define a canonical p-skeleton $S(G)$ associated with each graph G. The purpose of this construction is to provide a representation of G by a bounded number of forests of bounded depth. This intermediate structure serves as a bridge between graphs of bounded expansion and trees, allowing us to apply the known quantifier-elimination techniques that operate on tree-like structures. Once quantifiers are eliminated, we obtain a new skeleton $\widehat{S}(G)$ that captures, in a quantifier-free way, the properties expressed in the original formula.

Construction of the Skeleton. For $G \in \mathcal{G}$ and $p \in \mathbb{N}$, the p-skeleton $S(G)$ is built as follows. Let us fix a p-treedepth coloring $c : V(G) \to [f(p)] = \{1, \ldots, f(p)\}$ of G. For every $U \in \mathsf{Col}(p)$, we compute a spanning forest F^U of $G[U]$ by performing a depth-first search (DFS) traversal in each connected component of $G[U]$, starting from the node of minimum identifier in that component. Since $G[U]$ has treedepth at most p, the trees of F^U have depth at most $d = 2^p$ (see Proposition 4.11). The forests of $S(G)$ are those in the set $\{F^U : U \in \mathsf{Col}(p)\}$. New labels are assigned as follows.

- For every node u, its color label $c(u)$ is added to its new set of labels $\ell(u)$, that is, the color $c(u)$ of node u is included in its collection of labels $\ell(u)$.
- For every $U \in \mathsf{Col}(p)$, if u is at depth $i \in [d]$ in F^U, then the label (depth, i, U), denoted by depth_i^U for short, is added to $\ell(u)$, that is, every vertex colored by U has its depth in F^U included in its collection of labels.
- F^U is a collection of DFS trees. Therefore, every edge in $E(G[U])$ connects a vertex to one of its ancestors in F^U. For every such edge $\{u, v\}$ with u a descendant of v, and v at depth i, the label (level, i, U), abbreviated into level_i^U, is added to $\ell(u)$. That is, for each ancestor v of u that is adjacent to u in $G[U]$, $\ell(u)$ contains level_i^U, where i is the depth of v in F^U.

Remark. It follows from the last item that every edge of G is represented in $S(G)$ by a label attached to one of its endpoints. Indeed, let $e = \{u, v\}$ be an edge of G, and let $U \in \mathsf{Col}(p)$ be a color set containing the colors of both endpoints of e. Assume that u lies at depth j and v at depth $i < j$ in the forest F^U. Then the edge e is encoded as the label level_i^U attached to the deeper vertex u.

Any FO formula φ over labeled graphs can then be translated into an equivalent FO formula $\widetilde{\varphi}$ over p-skeletons by rewriting adjacency and equality predicates as Boolean combinations of lca and label predicates. For instance,

$$\begin{aligned}\mathsf{adj}(x, y) = \bigvee_{U \in \mathsf{Col}(p)} \bigvee_{0 \le i < j \le d} \Big[&\Big(\mathsf{lab}_{\mathsf{depth}_i^U}(x) \wedge \mathsf{lab}_{\mathsf{depth}_j^U}(y) \wedge \mathsf{lca}_i(x, y) \wedge \mathsf{lab}_{\mathsf{level}_i^U}(y)\Big) \\ &\vee \Big(\mathsf{lab}_{\mathsf{depth}_i^U}(y) \wedge \mathsf{lab}_{\mathsf{depth}_j^U}(x) \wedge \mathsf{lca}_i(x, y) \wedge \mathsf{lab}_{\mathsf{level}_i^U}(x)\Big) \Big].\end{aligned}$$

The equality predicates can be rewritten in a similar way. Therefore, for every assignment $\overline{v}$ to the free variables,

$$(G, \ell) \models \varphi(\overline{v}) \iff S(G) \models \widetilde{\varphi}(\overline{v}),$$

That is, φ and $\widetilde{\varphi}$ are semantically equivalent. One refers to labeled graphs while the other refers to skeletons, i.e., $\widetilde{\varphi}$ is the syntactic rewriting of the formula φ into the language of skeletons. We explain next how to remove quantifiers from $\widetilde{\varphi}$ to get the desired quantifier-free formula $\widehat{\varphi}$. Quantifier elimination is performed on skeletons.

Quantifier elimination in skeletons. Recall that we start from an existential formula

$$\varphi(x_1,\dots,x_k) = \exists y_1 \,\dots\, \exists y_t\, \psi(x_1,\dots,x_k,y_1,\dots,y_t),$$

where ψ is quantifier-free. We have that, for every assignment $\overline{v} = v_1,\dots,v_k$ and $\overline{u} = u_1,\dots,u_t$ of the variables such that $(G,\ell) \models \psi(\overline{v},\overline{u})$ holds, there exists $U \in \mathsf{Col}(p)$ where $p = t+k$ such that the colors of all u_i and v_j are in U. Hence, it follows from the previous discussion that

$$\begin{aligned}(G,\ell) \models \psi(\overline{x},\overline{y}) &\iff \exists U \in \mathsf{Col}(p) \,:\, (G[U],\ell) \models \psi(\overline{x},\overline{y})\\ &\iff \exists U \in \mathsf{Col}(p) \,:\, S(F^U) \models \widetilde{\psi}^U(\overline{x},\overline{y}),\end{aligned}$$

where $\widetilde{\psi}^U$ refers to nodes colored with colors in U only, and $S(F^U)$ denotes the skeleton $S(G)$ restricted to F^U. Thus the elimination of existential quantifiers from formulas on skeletons reduces to the elimination of these quantifiers from formulas on forests of bounded depth.

Quantifier elimination in bounded-depth forests. We eliminate existential quantifiers one at a time. Concretely, that is, we consider a subformula of the form

$$\exists y\ \psi(x_1,\dots,x_k,y),$$

and we remove the quantifier for this single variable y before proceeding to the next one. The quantifier-elimination procedure on bounded-depth forests relies on the notion of *types* (see Definition 4.3). For a rough definiton of what we mean by type, recall that, in forests of bounded depth, the structural relations between vertices can be expressed using the predicates $\mathsf{lca}_i(x,y)$ indicating that the lowest common ancestor of x and y lies at depth i (in what follows we omit the super-indices on the lca predicates). On a forest of depth d, a first-order formula therefore depends only on (i) the labels that are held by each vertex, and (ii) how the vertices of the tuple $\overline{x} = (x_1,\dots,x_k)$ are positioned with respect to one another in this ancestor hierarchy. These local configurations are captured thanks to different types. For a tuple of variables $\overline{x}$, a type is determined by

$$\text{a } \textit{label map } \gamma : [k] \to 2^{\Lambda}, \text{ and an } \textit{ancestry map } \delta : [k]\times[k] \to \{-1,0,\dots,d-1\}$$

that jointly describe which labels each variable x_i carries, and at which depths its ancestors stand. Each consistent pair (γ,δ) defines a conjunction $\zeta(\overline{x},y)$ of atomic predicates fixing both the label and ancestral structure of the tuple. Since there are finitely many such combinations, every quantifier-free formula over lca and label predicates can be rewritten as a finite disjunction of types. This presentation is referred to as the *basic normal form* of the formula (see Lemma 4.5). To eliminate an existential quantifier $\exists y\, \psi(\overline{x},y)$, we proceed type by type. For a fixed type $\zeta(\overline{x},y)$, we identify a variable x_s closest to y, and we distinguish three cases.

- If y is an ancestor of x_s at depth t, then we mark each vertex with an ancestor at depth t and carrying the appropriate labels by a new unary predicate good. We then replace the formula by the restriction of the type to $\overline{x}$ conjoined with the predicate $\mathsf{lab}_{\mathsf{good}}(x_s)$.

- If y lies in the subtree of a child of x_s, then we mark all nodes whose subtrees contain a potential *witness*[6], and attach a counter with bounded values indicating how many such subtrees exist, reducing the existential to a finite disjunction over child patterns.

- If y belongs to a different tree than all $\overline{x}$, then we label as *active* the roots of all trees that contain a suitable witness, and we propagate this information across components by communicating along the edges of G.

In all cases, we obtain a quantifier-free formula $\widehat{\psi}(\overline{x})$ with a few additional labels, unary predicates, and bounded counters.

Local vs. non-local formulas. Before going further, let us clarify how the locality of the formula affects the running time of quantifier elimination. Recall that our algorithm runs in $O(\log n)$ rounds for local formulas, and in $O(D + \log n)$ rounds for general ones, where D denotes the diameter of the communication graph. The following intuition explains this difference.

If ψ is local, then all occurrences of its variables lie within the same tree, at a bounded distance from one another. Hence, whether a node v should receive a new label (as part of eliminating the quantifier in ψ) can be decided by v using only local information from its own tree, whose depth is bounded. In the distributed setting, both the computation of these new labels and the evaluation of the resulting quantifier-free formula take place entirely within each tree, requiring only $\mathcal{O}(1)$ communication rounds in the $\mathsf{CONGEST}$ model.

If ψ is non-local, however, a witness for some tuple $\overline{x}$ may lie in a different tree. Then, the label assigned to a node v after eliminating the quantifier may depend on information carried by *active roots* located outside v's component, potentially at distance $\Omega(D)$ in the original graph G. Quantifier elimination for non-local formulas may thus require communication across components, along edges of G, at distances up to D, leading to a total complexity of $\mathcal{O}(D + \log n)$ rounds.

From existential to general formulas. So far, we have described how to eliminate quantifiers from a formula of the form $\exists y_1 \ldots \exists y_k \, \psi(x_1, \ldots, x_k, y_1, \ldots, y_k)$. To handle arbitrary FO formulas, which may alternate existential and universal quantifiers, we follow a standard transformation: every universal quantifier is rewritten as the negation of an existential one. This allows us to apply the existential elimination procedure iteratively. The process can be illustrated with the search for twins. Recall that

$$\textsc{Twin}(x) = \exists y \, \forall z \, (\mathsf{adj}(x, z) \Leftrightarrow \mathsf{adj}(y, z)).$$

We first rewrite the universal quantifier as a negated existential:

$$\textsc{Twin}(x) = \exists y \, \neg \exists z \, \neg(\mathsf{adj}(x, z) \Leftrightarrow \mathsf{adj}(y, z)).$$

We then focus on the innermost existential subformula,

$$\textsc{NoTwins}(x, y) = \exists z \, \neg(\mathsf{adj}(x, z) \Leftrightarrow \mathsf{adj}(y, z)).$$

[6] In this context, a *witness* for a tuple $\overline{x}$ in a formula $\exists y \, \psi(\overline{x}, y)$ is simply a vertex v such that $\psi(\overline{x}, v)$ holds. That is, v is a node whose existence makes the existential statement true. During quantifier elimination we never select this vertex explicitly. Instead, we locally encode, through additional labels, whether such a witness exists in the relevant part of the forest.

At this point, we eliminate the quantifier $\exists z$ using the procedure described before, producing a quantifier-free formula
$$\widehat{\text{NoTwins}}(x, y),$$
which, through new labels, encodes whether there exists a vertex z distinguishing x and y by adjacency. Our elimination procedure never identifies this witness explicitly. Instead, it records locally, via additional labels, whether such a witness exists within the neighborhood of the relevant variable. Substituting this expression into the original one yields the formula

$$\exists y\, \neg\widehat{\text{NoTwins}}(x, y).$$

Before eliminating the next quantifier $\exists y$, the formula $\neg\widehat{\text{NoTwins}}(x, y)$, which is expressed over p-skeletons, must be translated back into the standard structure of labeled graphs. This translation replaces each lca-predicate by an equivalent existential formula using only adjacency, equality and label predicates (cf. Lemma 5.5). After this translation, we obtain again an existential formula $\psi'(x, y)$ over labeled graphs. Adding the outer quantifier $\exists y$ preserves this existential form, yielding again a formula of the type $\exists y\, \psi'(x, y)$. We can therefore apply the same quantifier-elimination procedure once more. The outcome of this second elimination is a quantifier-free formula with a single free variable x, denoted by
$$\widehat{\text{Twin}}(x).$$
The resulting formula is expressed as a Boolean combination of unary predicates, each corresponding to a label on x that records whether a suitable witness y existed at a previous step.

This iterative process, alternating quantifier elimination and negation, extends the existential elimination method to arbitrary FO formulas. Each quantifier, regardless of its type, is ultimately replaced by local labels and Boolean combinations of unary predicates. But, there is a big but:

this procedure does not necessarily preserve locality!

Additional care is thus required for ensuring that each intermediate formula remains local. This is our second contribution, described next.

2.2 Locality-preserving quantifier elimination

Standard quantifier elimination at the core of model checking on sparse graphs [21, 46, 73] may *not preserve locality.* That is, even if the input formula is local, the resulting quantifier-free formula may refer to nodes that are arbitrarily far apart in the graph. Before illustrating this issue, let us first formalize what we mean by locality for formulas with *several* free variables.

Local formulas and distance predicates. Let $r \geq 0$ be an integer. A formula $\psi(x, \bar{y})$, with $\bar{y} = (y_1, \ldots, y_k)$, is said to be *$r$-local around* x if the following two conditions hold:

(i) Whenever $(G, \ell) \models \psi(x, \bar{y})$, all variables $\bar{y}$ are interpreted within the r-ball of x, that is,

$$(G, \ell) \models \psi(x, \bar{y}) \implies \bar{y} \subseteq B_G(x, r),$$

where $B_G(x, r)$ denotes the set of vertices at distance at most r from x in G.

(ii) The truth of ψ depends only on the labeled structure inside this ball: for any two labeled graphs (G, ℓ) and (G', ℓ'), and vertices $x \in V(G)$, $x' \in V(G')$, if there exists a label-preserving isomorphism $h : B_G(x, r) \to B_{G'}(x', r)$ with $h(x) = x'$, then

$$(G, \ell) \models \psi(x, \bar{y}) \iff (G', \ell') \models \psi(h(x), h(\bar{y})).$$

In formulas with several free variables, the distinguished variable x is called the *reference vertex*, and locality is always defined with respect to it.

In practice, enforcing r-locality syntactically is difficult. To make the dependence explicit, we use *distance predicates* $\mathrm{ball}(y, x, r)$, meaning that y is at distance at most r from x. By extension, for a tuple of variables $\overline{y}$, we write $\mathrm{ball}(\overline{y}, x, r)$ for $\bigwedge_{z \in \overline{y}} \mathrm{ball}(z, x, r)$. During quantifier elimination, quantified variables are guarded by appropriate ball predicates, and these guards are kept redundantly so that locality is preserved even under negation.

Failure of standard elimination. The loss of locality can be seen on the Twin property:

$$\textsc{Twin}(x) = \exists y\, \forall z\, \big(\mathsf{adj}(x, z) \Leftrightarrow \mathsf{adj}(y, z)\big),$$

which can be rewritten as

$$\textsc{NoTwins}(x, y) = \exists z\, \big[(\mathsf{adj}(x, z) \wedge \neg\mathsf{adj}(y, z)) \vee (\neg\mathsf{adj}(x, z) \wedge \mathsf{adj}(y, z))\big],$$

so that

$$\textsc{Twin}(x) = \exists y\, \neg\textsc{NoTwins}(x, y).$$

The formula $\textsc{NoTwins}(x, y)$ is not local *around* x: even with x fixed, there may exist y (and the witnessing z) at arbitrarily large distance from x.

Relativizing quantifiers. To prevent dependencies between distant vertices introduced during quantifier elimination, we adopt a redundant *relativization* of quantifiers. Instead of ranging over the entire structure, each quantified variable is evaluated only within the bounded region centered at the reference vertex, which is then translated to a tree of bounded depth. The relativization is expressed syntactically through distance predicates.

Formally, we define a *local form* for local formulas by induction on quantifier depth, as follows. A formula $\varphi(x, \overline{y})$ is in r-local form if one of the following conditions holds:

- $\varphi(x, \overline{y}) = \psi(x, \overline{y}) \wedge \mathrm{ball}(\overline{y}, x, r)$, where ψ is quantifier-free,
- $\varphi(x, \overline{y}) = \exists z\ \psi(x, \overline{y}, z)$, where ψ is in r-local form, or
- $\varphi(x, \overline{y}) = \mathrm{ball}(\overline{y}, x, r) \wedge \neg\exists z\ \mathrm{ball}((\overline{y}, z), x, r) \wedge \neg\psi(x, \overline{y}, z)$ where ψ is in r-local form.

Note how the universal quantifier requires a particular care, as it is handled as a negated existential quantifier. This formulation introduces negations, which do not preserve locality, as illustrated by the Twin example. This is why we introduce *redundant* ball predicates that guarantee syntactically the locality property.

Given a labeled graph (G,ℓ), and an FO formula φ in local form, our *locality-preserving quantifier elimination procedure* produces an updated labeling $\widehat{\ell}$ and a quantifier-free formula $\widehat{\varphi}$ such that, (1) for every vertex $v \in V(G)$,

$$(G,\ell) \models \varphi(v) \iff (G,\widehat{\ell}) \models \widehat{\varphi}(v),$$

and (2) every intermediate formula produced by the process remains in local form. That is we maintain explicit ball predicates throughout the transformation. All computations and communications therefore occur within the bounded-depth tree (or r-ball) associated with the reference vertex.

A key result in this paper is that any local formula can be rewritten in local form. Let us run through this transformation for the Twin formula.

Example: rewriting Twin in local form. Recall that $\textsc{Twin}(x) = \exists y\, \forall z \,\big(\mathsf{adj}(x,z) \Leftrightarrow \mathsf{adj}(y,z)\big)$ is 3-local. We start by *relativizing* each quantifier. This way, the locality of y and z with respect to x is explicit:

$$\textsc{Twin}^r(x) = \exists y \,\Big(\mathrm{ball}(y,x,3) \wedge \forall z \,\Big[\neg\mathrm{ball}(z,x,3) \vee \big(\mathsf{adj}(x,z) \Leftrightarrow \mathsf{adj}(y,z)\big)\Big]\Big).$$

Note that the formula above is logically equivalent to Twin, i.e., $\textsc{Twin}(x) \equiv \textsc{Twin}^r(x)$. For handling the universal quantifier $\forall z$, that is, rather, for handling the negated existential quantifier $\neg\exists z$, we introduce a redundant $\mathrm{ball}(y,x,3)$ predicate. This yields

$$\begin{aligned}\textsc{Twin}(x) &\equiv \exists y \,\Big(\mathrm{ball}(y,x,3) \wedge \neg\exists z \,\Big[\mathrm{ball}(z,x,3) \wedge \neg\big(\mathsf{adj}(x,z) \Leftrightarrow \mathsf{adj}(y,z)\big)\Big]\Big)\\ &\equiv \exists y \,\Big(\mathrm{ball}(y,x,3) \wedge \neg\exists z \,\Big[\mathrm{ball}(y,x,3) \wedge \mathrm{ball}(z,x,3) \wedge \neg\big(\mathsf{adj}(x,z) \Leftrightarrow \mathsf{adj}(y,z)\big)\Big]\Big)\\ &\equiv \exists y \,\Big(\mathrm{ball}(y,x,3) \wedge \neg\exists z \,\Big[\mathrm{ball}((y,z),x,3) \wedge \neg\big(\mathsf{adj}(x,z) \Leftrightarrow \mathsf{adj}(y,z)\big)\Big]\Big)\end{aligned}$$

At this point, one more step is needed for obtaining a formula in local form, as the innermost part of the formula, i.e., $\mathsf{adj}(x,z) \Leftrightarrow \mathsf{adj}(y,z)$, still requires a ball predicate to enforce its locality. This yields

$$\textsc{Twin}(x) \equiv \exists y \,\Big(\mathrm{ball}(y,x,3) \wedge \neg\exists z \,\Big[\mathrm{ball}((y,z),x,3) \wedge \neg\big(\mathrm{ball}((y,z),x,3) \wedge \mathsf{adj}(x,z) \Leftrightarrow \mathsf{adj}(y,z)\big)\Big]\Big).$$

This process successfully eventually produces a formula in 3-local form, as desired. In particular, note that, wherever there is a negation, there is a corresponding ball predicate whose role is to enforce locality not only before but also *after* the negation.

2.3 Model Checking, Counting, Optimization, and Certification

Model Checking General FO Predicates. For checking whether $G \models \varphi$ in $\mathsf{CONGEST}$, for a given FO formula φ, and a given labeled graph $G \in \mathcal{G}$, we use the same techniques as the ones used to mark all nodes satisfying a given local FO formula. That is, the proof of Theorem 1.5 follows the same guidelines as the ones used to establish Theorem 1.4. On the one hand, the approach is simplified as one does not need to care about relativization. On the other hand, the "non-locality" of the formula requires checking properties that potentially occur at nodes that are far away from each other in the graph (e.g., the presence of at least two nodes of degree 3). In the context of quantifier elimination, this translates into counting the number of nodes in which unary predicates are satisfied. This counting is merely achieved along the edges of a breadth-first search (BFS) tree, in $\mathcal{O}(D)$ rounds. We show that all other operations involved in quantifier elimination can be performed in $\mathcal{O}(\log n)$ rounds, for a total of $\mathcal{O}(D + \log n)$ rounds.

Counting and Optimization. Establishing Theorem 1.6, i.e., showing how to efficiently count the number of k-tuples of vertices $(v_1, \ldots, v_k)$ such that $(G, \ell) \models \varphi(v_1, \ldots, v_k)$, requires more work. For the global counting problem, we proceed in two stages.

Stage 1: Quantifier-free formulas. As for model checking, we use the fact that there exists an integer function g depending only on the class $\mathcal{G}$ such that, for any graph $G \in \mathcal{G}$, and any constant p, G admits a coloring with $g(p)$ colors such that any subset of at most p colors induces a subgraph of treedepth at most p. For $p = k$, each k-uple of vertices $(v_1, \ldots, v_k)$ of the labeled graph in (G, ℓ) satisfying the formula $\varphi(x_1, \ldots, x_k)$ covers some set U of colors, with $|U| \leq p$. Assuming given an $\mathcal{O}(D+\log n)$-round algorithm counting the solutions of $\varphi(x_1, \ldots, x_k)$ on labeled subgraphs $(G[U], \ell)$ of treedepth at most k, we show how to count the solutions on the whole labeled graph (G, ℓ). For any two sets of colors U_1, U_2, the set of solutions of $\varphi(x_1, \ldots, x_k)$ on $(G[U_1 \cap U_2], \ell)$ is exactly the intersection of the set of solutions for $(G[U_1], \ell)$ and for $(G[U_2], L)$. This holds thanks to the fact that $\varphi(x_1, \ldots, x_k)$ has no variables other than $(x_1, \ldots, x_k)$. As a consequence, by counting the solutions on $(G[U], \ell)$ for every set U of at most k colors, and then using the inclusion-exclusion principle, we are able to count solutions of φ on (G, ℓ).

Stage 2: General formulas. In order to use the previous stage on quantifier free formulas, we first proceed with quantifier elimination. This introduces label and lca predicates, which prevents us from directly applying the previous stage. To overcome this difficulty, we eliminate the lca predicates by introducing new variables, which are existentially quantified. To avoid over-counting, we enforce unicity, i.e., at most one instance of each newly introduced variable may satisfy the formula. Informally, it is as if every existential quantifier "there exists y such that" was replaced by "there exists *a unique* y such that".

To establish Theorem 1.7, we first utilize the entire quantifier elimination machinery developed for distributed model-checking in $\mathsf{CONGEST}$. We emphasize that, during the process, the variables of the quantifier-free formula $\widehat{\varphi}$ are precisely the free variables of $\varphi(\bar{x})$, but $\widehat{\varphi}$ may have a new set $\overline{y}$ of existentially-quantified variables. Let $p = |\overline{x}| + |\overline{y}|$. We construct now a coloring of G using $g(p)$ colors in $\mathcal{O}(\log n)$ rounds, and we observe that, for any k-uple of vertices $(v_1, \ldots, v_k)$ of G, we have $(G, \ell) \models \varphi(v_1, \ldots, v_k)$ if and only if there exists a set U of at most p colors such that $(G[U], \widehat{\ell}) \models \widehat{\varphi}(v_1, \ldots, v_k)$. Indeed, U is obtained by guessing not only the colors of $v_1, \ldots, v_k$, but also the colors of the tuple of vertices $u_1, \ldots, u_{p-k}$ assigned to $\overline{y}$. Optimizing the solution of φ on G boils down to optimizing over all optimal solutions of $\widehat{\varphi}$ on graphs $(G[U], \widehat{\ell})$ of treedepth at most p.

Based on the above, it remains to deal with optimization and global counting for FO formulas on labeled subgraphs $G[U]$ of G, each of which has a constant treedepth. For this purpose, we extend the results from [31] stating that counting and optimization of FO formulae can be achieved on *connected* labeled graphs of bounded treedepth, in a constant number of rounds. Indeed, we show that one can perform optimization and counting for FO formulae over an induced (not necessarily connected) subgraph of constant treedepth of the labeled graph G, in $\mathcal{O}(D)$ rounds. In [31], the algorithm starts by constructing a tree decomposition of small width of the graph, and then performs dynamic programming over this tree decomposition. If the input graph of small treedepth is connected, this directly enables the construction of a rooted tree decomposition of constant depth and constant width. Moreover, the decomposition tree is a spanning tree of the graph, which allows communications to proceed with dynamic programming bottom-up the tree in a constant number of rounds. The root of the tree can then compute the output by simulating the centralized algorithm for optimization and counting on graphs of bounded treewidth of [6]. However, in our case, the graphs $G[U]$ may not be connected, and thus we must deal with a forest decomposition F instead of a mere tree decomposition. A BFS tree rooted at an arbitrary vertex can span all the roots of the trees in the forest F, resulting in a rooted tree of depth $D + \mathcal{O}(1)$. This tree can serve as a base for constructing a tree-decomposition of $G[U]$. Moreover, since the nodes of the tree-decomposition correspond to vertices of G, the centralized dynamic programming algorithm of [6] can be simulated for counting and optimization on $G[U]$, in $D + \mathcal{O}(1)$ rounds. The information stored in the dynamic programming tables can indeed be computed in a sequence of rounds, starting from the leaves, upward to the root of the tree decomposition by communicating along the edges of G. Altogether, the whole process for counting and optimization for formula φ on the pair (G, ℓ) costs $\mathcal{O}(D + \log n)$ rounds in $\mathsf{CONGEST}$.

Our method for local counting follows the same lines as that for global counting. That is, we first reduce the study to quantifier-free formulas, using the (locality preserving) quantifier elimination method. Similarly to the case of global counting, this reduction introduces lca-predicates, which need to be eliminated in a locality-preserving fashion. It turns out that the method applied to the case of global counting happens to preserve locality, although the proof is somewhat technical. For quantifier-free formulas, we essentially apply the same idea as for global counting, based on the inclusion-exclusion principle. However, we also use the fact that the formula is local, by considering some set of colors U, and a tuple of vertices $\overline{v}$ such that $(G[U], \ell)$ satisfies $\varphi(\overline{v})$. Then, since φ is local, all vertices in $\overline{v}$ belong to the same connected component of $G[U]$, which implies that we only need to count the solutions to φ in the connected components of $G[U]$. The latter are connected graphs of bounded treedepth, and thus counting is achieved in $\mathcal{O}(1)$ rounds using the algorithm in [31].

Distributed Certification. At a high level, our certification scheme for an FO formula φ and a class of graphs $\mathcal{G}$ with bounded expansion essentially consists of the prover providing each node with the transcript of the execution of our distributed model-checking algorithm at this node. The role of the verifier is then to check (in a distributed manner) whether this transcript encodes a correct execution of our algorithm. Note that the certification scheme does not have to certify the bounded expansion property of $\mathcal{G}$, which is a promise. However, given a coloring of the input graph $G = (V, E) \in \mathcal{G}$, the certification scheme must certify that any set U of $k = |\varphi|$ colors does induce a subgraph with treedepth k. This is done in a specific manner, using an algorithm by Bousquet et al. [7] for certifying MSO formulas on graphs of bounded treedepth with certificates of size $\mathcal{O}(\log n)$. The main difficulty is to encode the transcript of the quantifier elimination procedure, i.e., the transformation from (φ, G) to $(\widehat{\varphi}, \widehat{\ell})$ with $\widehat{\varphi}$ quantifier-free. Each lca predicate refers to a local condition, and the value of such a predicate can be certified locally at each node, in forests of bounded depth. On the other hand, the unary predicates may refer to non-local properties. Nevertheless, we demonstrate that verifying the correctness of the unary predicates can be accomplished using a spanning tree whose root is capable of making decisions. (Certify a spanning tree is standard using $\mathcal{O}(\log n)$-bit certificates [56].)

3 Model and Definitions

In this section, we recall basic concepts of distributed computing, logic, and graphs. Readers familiar with these frameworks may skip the first two sub-sections, but Section 3.3, which introduces and extends the notion of *local* formulas, and Section 3.4, which recalls properties of graphs with *bounded expansion*, may require special attention.

3.1 The CONGEST Model

The $\mathsf{CONGEST}$ model is a standard model of distributed computation in networks [70]. The network is modeled as a connected simple (i.e., no self-loops nor multiple edges) graph $G = (V, E)$. The $n \geq 1$ vertices of G model the processing nodes in the networks, and its edges models the communication links. In the following, we may indistinguishably use "node" of "vertex" for referring to the processing elements in the network. Each node is assigned an identifier, which is assumed to be unique in the network. The identifiers may not be between 1 and n, but the $\mathsf{CONGEST}$ model assumes that each identifier can be stored on $\mathcal{O}(\log n)$ bits in n-node networks. In fact, it is usually assumed that all the nodes of a graph G are given a polynomial upper bound $N = \text{poly}(n)$ on the number n of vertices in G, with all identifiers in $\{1, \dots, N\}$.

Computation proceeds in lockstep, as a sequence of synchronous rounds. All nodes start simultaneously, at the same round. At each round, every node v sends an $\mathcal{O}(\log n)$-bit message to each of its neighbors, receives the messages sent by all its neighbors, and performs some individual computation. The round-complexity of an algorithm A is the function $R_A(n) = \max_{|V(G)| \leq n} R_A(G)$ where $R_A(G)$ denotes the number of rounds performed by the nodes executing A in G until all of them terminate.

As said before, it is worth mentioning the $\mathsf{BROADCAST\ CONGEST}$ model, that is, the weaker variant of the $\mathsf{CONGEST}$ model in which, at every round, each node must send the *same* $\mathcal{O}(\log n)$-bit message to all its neighbors. For the sake of simplifying the notations, we solely refer to the $\mathsf{CONGEST}$ model in the statements of our results. Yet, all our upper bound results also holds in the weaker $\mathsf{BROADCAST\ CONGEST}$ model, apart from Theorem 6.2.

Remark. Throughout the paper, we will use the following convention regarding the inputs and outputs of our algorithms. When an algorithm receives a "global value" as input, whether it be an integer p, a logical formula φ, or a function (e.g., a bound $f : \mathbb{N} \to \mathbb{N}$ on the expansion), we assume that all nodes are given this global value as input. Instead, when an algorithm receives as input a node labeling function ℓ, we assume that each node v is only given its label $\ell(v)$. Similarly, when an algorithm returns a node-labeling function $\widehat{\ell}$, we mean that each node v returns its label $\widehat{\ell}(v)$.

3.2 First-Order Formulas on Labeled Graphs

3.2.1 Definition

Recall that a *first-order* (FO) formula over graphs is a combination (conjunction, disjunction, and negations) of Boolean predicates over variables, which may be quantified or not. Every variable represents a vertex of a graph, and there are two binary predicates: equality $x = y$, and adjacency $\mathsf{adj}(x, y)$, where the latter states whether or not there is an edge between the two vertices x and y. The variables of an FO formula can be quantified with existential or universal quantifiers. Non-quantified variables are called *free variables*. Given an FO formula φ with free variables $x_1, \dots, x_k$, we make explicit that the latter are free by denoting $\varphi = \varphi(x_1, \ldots, x_k)$. In the following, FO denotes the set of all first order formulas over undirected graphs.

Throughout the paper, a tuple $(x_1, \ldots, x_k)$ is abbreviated in $\overline{x}$. The length k of a tuple $(x_1, \ldots, x_k)$ is then denoted by $|\overline{x}|$. Expressions such as $\exists\overline{x}$ or $\forall\overline{x}$ are shorthand for $\exists x_1 \ldots \exists x_k$ and $\forall x_1 \ldots \forall x_k$, respectively. We may also abuse notation in context where no confusions may occur by letting $\overline{x}$ denote the *set* $\{x_1, \ldots, x_k\}$.

Given an FO formula φ and a graph G, $G \models \varphi$ denotes that G satisfies φ (i.e., G is a *model* for φ). The negation of $G \models \varphi$ is denoted by $G \not\models \varphi$. Given an FO formula $\varphi(\overline{x})$ with free variables, a graph $G = (V, E)$, and a set of vertices $\overline{v} \in V^k$ where $k = |\overline{x}|$, we say that G satisfies $\varphi(\overline{x})$ for vertices $\overline{v}$, denoted by $G \models \varphi(\overline{v})$, if the formula is satisfied by G whenever assigning $\overline{v}$ to the free variables of $\varphi(\overline{x})$.

A graph property is called a *first-order* (FO) property if it can be expressed by a first-order formula with no free variables. For example, the property "the graph contains a triangle" is clearly an FO property: it holds whenever there is a set of three pairwise adjacent vertices in the graph. Indeed, it can be written as the formula

$$\varphi \;=\; \exists x_1\, \exists x_2\, \exists x_3\, \big(\mathrm{adj}(x_1, x_2) \;\wedge\; \mathrm{adj}(x_2, x_3) \;\wedge\; \mathrm{adj}(x_1, x_3)\big).$$

More generally, the property of containing a fixed graph H as a subgraph (sometimes called "subgraph isomorphism") is an FO property. Its negation—namely, the property that a graph does not contain H as a subgraph (often called "H-freeness")—is also an FO property, since first-order logic is closed under negation.

A formula with free variables expresses properties over graphs and tuples of vertices. As an example, the following formula $\psi(x)$ holds whenever x is a node in a triangle, formulated as

$$\psi(x) = \exists y_1 \exists y_2, \mathsf{adj}(x, y_1) \wedge \mathsf{adj}(x, y_2) \wedge \mathsf{adj}(y_1, y_2).$$

The graph G with vertex set $V = \{a, b, c, d\}$, and edge set $E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{c, d\}\}$ satisfies $G \models \varphi$ as the vertices $\{a, b, c\}$ form a triangle. Similarly, $G \models \psi(a)$, but $G \not\models \psi(d)$ as d is not in any triangle.

3.2.2 FO Formulas for Labeled Graphs

The formalism of FO properties can be extended to labeled graphs. Let Λ be a finite set, whose elements are called *labels*. A Λ-labeled graph is a pair (G, ℓ) where $G = (V, E)$ is a graph, and $\ell : V \to 2^{\Lambda}$ is an assignment of a set of labels to each vertex of G. An FO formula over Λ-labeled graphs is a first order formula φ over graphs, extended with predicates of the form $\mathsf{lab}_{\lambda}(x)$ for every $\lambda \in \Lambda$. The predicate $\mathsf{lab}_{\lambda}(x)$ is true if vertex x contains λ as one of its labels. Such predicates are called *label predicates*. Whenever $\mathsf{lab}_{\lambda}(x)$ holds, we say that x is labeled with λ. Note that since ℓ assigns a set of labels to each vertex, it may be the case that $\mathsf{lab}_{\lambda}(x) \wedge \mathsf{lab}_{\lambda'}(x)$ holds for two different labels λ and λ'. It may thus be convenient to define the predicates lab_S for sets $S \subseteq \Lambda$, as follows. Given $S \in \Lambda$, $\mathsf{lab}_S(x)$ holds if and only if $\ell(x) = S$. In particular, $\mathsf{lab}_S(x) \wedge \mathsf{lab}_{S'}(x)$ is necessarily false whenever $S \neq S'$. The set of all first-order logic formulas over Λ-labeled graphs is denoted FO$[\Lambda]$.

Given a Λ-labeled graph (G, ℓ), and a subgraph G' of G, we abuse notation and denote (G', ℓ) the Λ-labeled graph defined by the restriction of ℓ to the nodes in G'. For instance, let $\Lambda = \{0, 1\}$, and let us consider the following graph property on labeled graphs: "existence of a triangle with a node labeled 1". This property can be expressed by the formula:

$$\varphi = \exists x_1 \exists x_2 \exists x_3, \mathsf{adj}(x_1, x_2) \wedge \mathsf{adj}(x_2, x_3) \wedge \mathsf{adj}(x_1, x_3) \wedge \mathsf{lab}_1(x_1).$$

We may also consider formula with free variables, such as

$$\psi(x) = \exists y_1 \exists y_2, \mathsf{adj}(x, y_1) \wedge \mathsf{adj}(x, y_2) \wedge \mathsf{adj}(y_1, y_2) \wedge \mathsf{lab}_1(x).$$

The graph G with vertex set $V = \{a, b, c, d\}$ and edge set $E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{c, d\}\}$ with labels $\ell(a) = \{0\}, \ell(b) = \{0\}, \ell(c) = \{1\}$, and $\ell(d) = \{1\}$ satisfies $(G, \ell) \models \varphi$ as the nodes $\{a, b, c\}$ form a triangle where c is labeled 1. For the same reason, we have that $(G, \ell) \models \psi(c)$. However, $(G, \ell) \not\models \psi(a)$ as a is not labeled 1 (even if it is in a triangle), and $(G, \ell) \not\models \psi(d)$ as node d is not in a triangle (even if it is labeled 1).

For $\varphi(x_1, \dots, x_k) \in \mathrm{FO}[\Lambda]$, and a Λ-labeled graph (G, ℓ), we denote by $\mathsf{true}(\varphi, G, \ell)$ the set of all tuples of vertices $(v_1, \dots, v_k)$ satisfying that $(G, \ell) \models \varphi(v_1, \dots, v_k)$. If φ has no free variables then $\mathsf{true}(\varphi, G, \ell)$ is equal to the truth value of $(G, \ell) \models \varphi$. Two $\mathrm{FO}[\Lambda]$ formulas φ_1 and φ_2 are *equivalent* if $\mathsf{true}(\varphi_1, G, \ell) = \mathsf{true}(\varphi_2, G, \ell)$ for every Λ-labeled graph (G, ℓ). The following basic definitions play a central role in this paper.

- A formula $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$ is called *quantifier-free* if it has no quantifiers, i.e., it can be expressed as a boolean combination of predicates involving only the free variables in $\overline{x}$.

- A formula $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$ is *existential* if it can be written as $\varphi(\overline{x}) = \exists \overline{y}\, \psi(\overline{x}, \overline{y})$, where $\psi(\overline{x}, \overline{y})$ is quantifier-free formula in $\mathrm{FO}[\Lambda]$.

Furthermore, a formula $\varphi(\overline{x})$ is in *prenex normal form* if it can be written in the form

$$\varphi(\overline{x}) = Q_1 y_1 \, \dots \, Q_t y_t \; \psi(\overline{x}, \overline{y}),$$

where $\psi(\overline{x}, \overline{y}) \in \mathrm{FO}[\Lambda]$, and $Q_1, \dots, Q_t$ are existential or universal quantifiers. The value of t is the *quantifier depth* of φ. Every FO formula admits an equivalent formula in prenex normal form (see, e.g., [24]). This fact can be easily established by induction on the length of the formula, and it can be extended to $\mathrm{FO}[\Lambda]$, i.e., first order formulas for labeled graphs, in a straightforward manner.

Notation. For every family $\mathcal{G}$ of graphs, and every finite set Λ, we denote by $\mathcal{G}[\Lambda]$ the set of all Λ-labeled graphs (G, ℓ) with $G = (V, E) \in \mathcal{G}$, and $\ell : V \to 2^\Lambda$.

3.3 Local Formulas

Local formulas can be defined as follows. For a graph $G = (V, E)$ and a vertex $v \in V$, let $B_G(v, r)$ denote the r-neighborhood of v in G, that is, the subgraph of G induced by vertices at distance at most r from v. By extension, when considering a labeled graph (G, ℓ), $B_G(v, r)$ is also a labeled graph: each node $u \in B_G(v, r)$ is labeled by the same set of labels $\ell(u)$.

Definition 3.1. *Let Λ be a finite set of labels, and let $r \geq 0$ be an integer. A formula $\varphi(x) \in FO[\Lambda]$ with one free variable is* r-local *if, for any two Λ-labeled graphs (G, ℓ) and (G', ℓ'), and for any two vertices $v \in V(G)$ and $v' \in V(G')$, we have*

$$B_G(v, r) = B_{G'}(v', r) \implies \Big((G, \ell) \models \varphi(v) \iff (G', \ell') \models \varphi(v') \Big).$$

where $B_G(v,r) = B_{G'}(v',r)$ means that there exists a label-preserving isomorphism $h : B_G(v,r) \to B_{G'}(v',r)$ such that $h(v) = v'$.

We say $\varphi(x)$ is local *if there exists $r \geq 0$ such that $\varphi(x)$ is r-local.*

For technical reasons that will appear clear later, we need to extend the notion of local formulas to formulas $\varphi(x, \overline{y})$ with more than a single free variable. Our notion of locality is defined with respect to the first free variable x of the formula: intuitively, we require that each variable is in a ball centered on x. This is different from the notion of Gaifman-locality [35], in which *quantified* variables are required to be in a ball centered on *any* of the free variables.

Definition 3.2. *Let Λ be a finite set of labels, and let $r \geq 0$ be an integer. A formula $\varphi(x, \overline{y}) \in FO[\Lambda]$ is r-*local *if, for any Λ-labeled graph (G, ℓ) with $G = (V, E)$, the following two properties are satisfied:*

- *for every pair $(v, \overline{u}) \in \mathsf{true}(\varphi, G, \ell)$, it holds that $\overline{u} \subseteq B_G(v, r)$;*
- *for every Λ-labeled graph (G', ℓ') with $G' = (V', E')$, for every vertices $v \in V, v' \in V'$, $\overline{u} \subseteq B_G(v,r)$ and $\overline{u'} \subseteq B_{G'}(v', r)$, if there exists a label-preserving isomorphism $h : B_G(v,r) \to B_{G'}(v',r)$ such that $h(v) = v'$ and $h(\overline{u}) = \overline{u'}$, then:*

$$(G, \ell) \models \varphi(v, \overline{u}) \iff (G', \ell') \models \varphi(v', \overline{u'})$$

We shall often write $\varphi(\overline{x})$ instead of $\varphi(x, \overline{y})$, in which case it is x_1 that takes the particular role of x in the definition above.

The above definition is, unfortunately, very difficult in practice, because it provides little information on the form of the considered formula. We therefore define the notion of *local form* in Definition 3.3. To that end, let us introduce the new predicate $\mathrm{ball}(y, x, r)$ defined as

$$G \models \mathrm{ball}(u, v, r) \iff u \in B_G(v, r).$$

More generally, for $\overline{y} = (y_1, \ldots, y_k)$, we define

$$\mathrm{ball}(\overline{y}, x, r) = \bigwedge_{i=1}^{|\overline{y}|} \mathrm{ball}(y_i, x, r).$$

These predicates will be referred to as *distance predicates*. Note that they can easily be written as existential FO formulas, see Lemma 3.5. We use distance predicates to *relativize* quantifiers with respect to the r-neighborhood of a given vertex (relativization of quantifiers is standard technique, see for example [53, 22]): let us then consider an existential r-local formula $\varphi(x) = \exists \overline{y}\, \psi(x, \overline{y})$. Since $\varphi(x)$ is r-local, it can be rewritten as

$$\varphi(x) = \exists \overline{y}\ \big(\psi(x, \overline{y}) \wedge \mathrm{ball}(\overline{y}, x, r)\big).$$

Such a formula, where the locality is enforced explicitly by a distance predicate, is referred to as a formula in *r-local form.* The following definition formalizes this notion.

Definition 3.3. *Let $r \geq 0$ be an integer. A formula $\varphi(x, \overline{y}) \in FO[\Lambda]$ with at least one free variable is in r-*local form *if*

- $\varphi(x,\overline{y}) = \psi(x,\overline{y}) \wedge \mathrm{ball}(\overline{y},x,r)$, *where* $\psi \in FO[\Lambda]$ *is quantifier-free, or*
- $\varphi(x,\overline{y}) = \exists z\ \psi(x,\overline{y},z)$, *where* $\psi \in FO[\Lambda]$ *is in* r*-local form, or*
- $\varphi(x,\overline{y}) = \mathrm{ball}(\overline{y},x,r) \wedge \forall z\ \big(\neg\mathrm{ball}((\overline{y},z),x,r) \vee \psi(x,\overline{y},z)\big)$ *where* $\psi \in FO[\Lambda]$ *is in* r*-local form.*

We say $\varphi(x,\overline{y})$ *is in* local form *if there exists* $r \geq 0$ *such that* $\varphi(x,\overline{y})$ *is in* r*-local form.*

When considering a formula with several free variables, we say it is (r-)local if it can be rewritten in (r-)local form.

Remark. The last case of Definition 3.3 introduces a lot of redundancy, with two distance predicates, in addition to requiring that ψ must be in r-local form. The main motivation for this redundancy is to guarantee that every "subformula" of φ remains local. Thanks to this redundancy, all the formulas below are local, despite the fact that negation does *not* preserve locality for formulas with several free variables. In particular, the following two formulas are equivalent, where the universal quantifier is merely replaced by a negated existential quantifier:

$$\begin{aligned}&\mathrm{ball}(\overline{y},x,r) \wedge \forall z\ \big(\neg\mathrm{ball}((\overline{y},z),x,r) \vee \psi(x,\overline{y},z)\big)\\ &\qquad\qquad \equiv \mathrm{ball}(\overline{y},x,r) \wedge \neg\exists z\ \big(\mathrm{ball}((\overline{y},z),x,r) \wedge \neg\psi(x,\overline{y},z)\big).\end{aligned}$$

Similarly, the formulas

$$\exists z\ \big(\mathrm{ball}((\overline{y},z),x,r) \wedge \neg\psi(x,\overline{y},z)\big),\ \text{and}\ \mathrm{ball}((\overline{y},z),x,r) \wedge \neg\psi(x,\overline{y},z)$$

are both local. In other words, the implicit or explicit presence of negations forces us to introduce redundancies in the distance predicates for "balancing" the potential loss of the locality property.

The following result establishes the consistency of Definitions 3.2 and 3.3.

Lemma 3.4. *For every integer* $r \geq 0$*, every* r*-local formula can be rewritten in* r*-local form.*

Proof. Let $\varphi(x,\overline{y})$ be r-local formula, and assume, w.l.o.g, that it is written in prenex normal form $\varphi(x,\overline{y}) = Q_1 z_1 \dots Q_t z_t\ \psi(\overline{x},\overline{y},\overline{z})$. First, we rewrite φ by *relativizing* the quantifier with respect to the r-neighborhood of x. Intuitively, we replace existential quantification $\exists z$ by $\exists z \in B_G(x,r)$, i.e., z must be in the ball of radius r around x in G. Similarly universal quantification $\forall z$ is replaced by $\forall z \in B_G(x,r)$, i.e., the formula is checked only for nodes z close to x. Formally, for the r-local formula $\varphi(x,\overline{y})$, we define $\varphi^r(x,\overline{y})$ by induction on quantifier depth. If φ is quantifier free, then $\varphi^r = \varphi$. If $\varphi(x,\overline{y}) = \exists z, \zeta(x,\overline{y},z)$, then $\varphi^r(x,\overline{y}) = \exists z, \mathrm{ball}(z,x,r) \wedge \zeta^r(x,\overline{y},z)$. Finally, if $\varphi(x,\overline{y}) = \forall z, \zeta(x,\overline{y},z)$, then $\varphi^r(x,\overline{y}) = \neg\exists z, \mathrm{ball}(z,x,r) \wedge \neg\zeta^r(x,\overline{y},z)$.

By definition, for $v \in V$ and $\overline{u} \subseteq B_G(v,r)$,

$$G \models \varphi^r(v,\overline{u}) \iff B_G(v,r) \models \varphi^r(v,\overline{u}) \iff B_G(v,r) \models \varphi(v,\overline{u}).$$

By locality of φ, we deduce that

$$G \models \mathrm{ball}(\overline{u},v,r) \wedge \varphi^r(v,\overline{u}) \iff B_G(v,r) \models \varphi(v,\overline{u}) \iff G \models \varphi(v,\overline{u}).$$

It only remains to show that the formula $\mathrm{ball}(\overline{y},x,r) \wedge \varphi^r(x,\overline{y})$ can be rewritten in r-local form. Once again, we proceed by induction on quantifier depth. By definition of φ^r, there are only three cases to consider.

Case 1: φ is quantifier free. Then $\varphi^r = \varphi$ and $\mathrm{ball}(\overline{y}, x, r) \wedge \varphi(x, \overline{y})$ is already in local form.

Case 2: $\varphi(x, \overline{y}) = \exists z\, \psi(x, \overline{y}, z)$. Then,

$$\begin{aligned}\mathrm{ball}(\overline{y}, x, r) \wedge \varphi^r(x, \overline{y}) &= \mathrm{ball}(\overline{y}, x, r) \wedge \exists z \left(\mathrm{ball}(z, x, r) \wedge \psi^r(x, \overline{y}, z)\right) \\ &= \exists z \left(\mathrm{ball}((\overline{y}, z), x, r) \wedge \psi^r(x, \overline{y}, z)\right)\end{aligned}$$

By induction, we can rewrite $\mathrm{ball}((\overline{y}, z), x, r) \wedge \psi^r(x, \overline{y}, z)$ in r-local form.

Case 3: $\varphi(x, \overline{y}) = \forall z, \psi(x, \overline{y}, z)$. Then,

$$\begin{aligned}\mathrm{ball}(\overline{y}, x, r) \wedge \varphi^r(x, \overline{y}) &= \mathrm{ball}(\overline{y}, x, r) \wedge \neg\exists z \left(\mathrm{ball}(z, x, r) \wedge \neg\psi^r(x, \overline{y}, z)\right) \\ &= \mathrm{ball}(\overline{y}, x, r) \wedge \neg\Big(\mathrm{ball}(\overline{y}, x, r) \wedge \exists z \left(\mathrm{ball}(z, x, r) \wedge \neg\psi^r(x, \overline{y}, z)\right)\Big) \\ &= \mathrm{ball}(\overline{y}, x, r) \wedge \neg\exists z \left(\mathrm{ball}((\overline{y}, z), x, r) \wedge \neg\psi^r(x, \overline{y}, z)\right) \\ &= \mathrm{ball}(\overline{y}, x, r) \wedge \forall z \left(\neg\mathrm{ball}((\overline{y}, z), x, r) \vee \psi^r(x, \overline{y}, z)\right) \\ &= \mathrm{ball}(\overline{y}, x, r) \wedge \forall z \Big(\neg\mathrm{ball}((\overline{y}, z), x, r) \vee \left(\mathrm{ball}((\overline{y}, z), x, r) \wedge \psi^r(x, \overline{y}, z)\right)\Big)\end{aligned}$$

By induction, $\mathrm{ball}((\overline{y}, z), x, r) \wedge \psi^r(x, \overline{y}, z)$ can be rewritten in r-local form.

The study of these three cases completes the proof. □

We conclude this section by some simple observation that will be useful at several places in the paper.

Lemma 3.5. *Let $r \in \mathbb{N}$, and let $\overline{x} = (x_1, \dots, x_k)$. There exists an existential FO formula*

$$\beta_r(\overline{x}) = \exists \overline{y}\, \psi(\overline{x}, \overline{y}),$$

where ψ is quantifier free such that, for every graph G, and every k-tuple $\bar{v}$ of vertices of G,

$$G \models \beta_r(\overline{v}) \iff G \models \mathrm{ball}(\overline{v}, v_1, r).$$

Moreover, for every $r' \geq r$, every graph G, and every k-tuple $\bar{v}$ of vertices of G,

$$G \models \beta_r(\overline{v}) \iff G \models \beta_{r,r'}(\overline{v}),$$

where $\beta_{r,r'}(\overline{x}) = \exists \overline{y} \left(\psi(\overline{x}, \overline{y}) \wedge \mathrm{ball}((\overline{x}, \overline{y}), x_1, r')\right)$.

Proof. It is sufficient to describe $\beta_r(x, y)$ with two variables only, as $\beta_r(x_1, \dots, x_k)$ can be written as $\bigwedge_{j=2}^{k} \beta_r(x_1, x_j)$. We define

$$\beta_r(x, y) = \exists z_0 \dots \exists z_r \left[(x = z_0) \wedge \bigvee_{j=0}^{r} \Big((y = z_j) \wedge \bigwedge_{i=0}^{j-1} \mathsf{adj}(z_i, z_{i+1})\Big)\right].$$

In this formula, j is the distance between x and y, and $z_0, \dots, z_j$ represents the path from x to y. This formula satisfies the first item of Lemma 3.5. Let us now prove that it satisfies the second item. Let $r' \geq r$. By definition, if $G \models \beta_{r,r'}(u, v)$ then $G \models \beta_r(u, v)$. Conversely, let us assume that $G \models \beta_r(u,v)$. Then, there exists some $\overline{w} = w_0, \dots, w_r$ such that $G \models \psi(u,v,\overline{w})$. Moreover, there exists $j \in [0,r]$ such that

$$G \models (u = w_0) \wedge (v = w_j) \wedge \bigwedge_{i=0}^{j-1} \mathsf{adj}(w_i, w_{i+1}).$$

For every $i \in [0,j]$, let $w'_i = w_i$, and, for every $i \in [j+1,r]$, let $w'_i = u$. By definition, every vertex w'_i satisfies $\mathrm{ball}(w'_i, u, r)$, and thus $\mathrm{ball}(w'_i, u, r')$ since $r' \geq r$. This implies that

$$G \models (u = w'_0) \wedge (v = w'_j) \wedge \bigwedge_{i=0}^{j-1} \mathsf{adj}(w'_i, w'_{i+1}) \wedge \mathrm{ball}((u,v,\overline{w'}), u, r').$$

As a consequence, $G \models \psi(u,v,\overline{w'}) \wedge \mathrm{ball}((u,v,\overline{w'}), u, r')$, and thus $G \models \beta_{r,r'}(u,v)$. □

Lemma 3.6. *Let Λ be a set of labels, and let $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$ be a r-local, quantifier-free formula. If $\varphi(\overline{x}) = \varphi_1(\overline{x}) \vee \varphi_2(\overline{x})$ for some formulas φ_1 and φ_2, then both φ_1 and φ_2 must be r-local.*

Proof. Since φ_1 and φ_2 play symmetric roles, we only prove that φ_1 is local. Since φ is quantifier free, and therefore φ_1 as well, checking the r-locality of φ_1 is equivalent to checking that, for all Λ-labeled graphs (G,ℓ),

$$(G,\ell) \models \varphi_1(\overline{v}) \implies \overline{v} \subseteq B_G(v_1, r).$$

This holds trivially as, by locality of φ,

$$(G,\ell) \models \varphi_1(\overline{v}) \implies (G,\ell) \models \varphi(\overline{v}) \implies \overline{v} \subseteq B_G(v_1, r),$$

which completes the proof. □

3.4 Graphs of Bounded Expansion

Recall that a graph $G = (V,E)$ has *treedepth* at most d if there exists a forest F of rooted trees with depth at most d satisfying that there exists a one-to-one mapping $f : V \to V(F)$ such that, for every edge $\{u,v\} \in E$, $f(u)$ is an ancestor of $f(v)$ in a tree of F, or $f(v)$ is an ancestor of $f(u)$ in a tree of F. The treedepth of a graph is the smallest d such that such a mapping f exists.

It has been recently shown in [31] that decision, counting and optimization of FO formulas can be performed in $\mathcal{O}(1)$ rounds in $\mathsf{CONGEST}$, for *connected* labeled graphs of bounded treedepth.

Proposition 3.7 ([31])**.** *Let $\mathcal{G}[\Lambda]$ be a class of* connected *labeled graphs of bounded treedepth, and let $\varphi(x_1, \dots, x_k) \in FO[\Lambda]$. Then, for any and any $(G,\ell) \in \mathcal{G}[\Lambda]$, the following problems can be solved by a distributed algorithm running in $\mathcal{O}(1)$ rounds in $\mathsf{CONGEST}$:*

- **Deciding φ:** *deciding whether it exists a k-uple of vertices $(v_1, \dots, v_k)$ such that $(G,\ell) \models \varphi(v_1, \dots, v_k)$.*
- **Counting the solutions of φ:** *computing the number of k-uples $(v_1, \dots, v_k)$ such that $(G,\ell) \models \varphi(v_1, \dots, v_k)$.*
- **Optimizing φ:** *assuming n-node graphs $G = (V,E)$ are provided with a weight function $\omega : V \to \mathbb{N}$ satisfying $\omega(v) = O(n^c)$ for some $c \geq 0$, computing a k-tuple of vertices $(v_1, \dots, v_k)$ such that $G \models \varphi(v_1, \dots, v_k)$, and $\sum_{i=1}^{k} \omega(v_i)$ is maximum (or minimum).*

Graphs of bounded expansion [67] can be defined in term of local minors, and degeneracy. It is however more convenient for us to use the following characterization. A class $\mathcal{G}$ of graphs has *bounded expansion* if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that, for every $G \in \mathcal{G}$, and for every $k \in \mathbb{N}$, there exists a coloring of the vertices of G with colors in $[f(k)] = \{1, \dots, f(k)\}$ such that, for every set $U \subseteq [f(k)]$ of cardinality at most k the subgraph of G induced by the nodes colored by a color in U has treedepth at most $|U|$. In particular, each subset of at most k colors induces a subgraph of treedepth at most k. For every positive integers p and k, a (p, k)*-treedepth coloring* of a graph G is a k-coloring of G satisfying that each set $U \subseteq [k]$ of at most p colors induces a graph of treedepth at most $|U|$.

Definition 3.8. *Given a function $f : \mathbb{N} \to \mathbb{N}$, a class of graphs $\mathcal{G}$ has* expansion f *if, for every $p \in \mathbb{N}_{>0}$, every $G \in \mathcal{G}$ has a $(p, f(p))$-treedepth coloring. A class of graphs has* bounded *expansion if it has expansion f for some function $f : \mathbb{N} \to \mathbb{N}$.*

A class of graphs has *effective* bounded expansion if it has expansion f for some Turing-computable function f. In the following, we always assume that the considered functions f are Turing-computable. Interestingly, treedepth colorings can be efficiently computed in the $\mathsf{CONGEST}$ model.

Proposition 3.9 ([68])**.** *Let $\mathcal{G}$ be a class of graphs of bounded expansion f. There exists a function $g : \mathbb{N} \to \mathbb{N}$ such that, for any graph $G \in \mathcal{G}$, and any constant positive integer p, a $(p, g(p))$-treedepth coloring of G can be computed in $\mathcal{O}(\log n)$ rounds in the $\mathsf{CONGEST}$ model.*

The algorithm of [68] takes as input the function f, and an integer p, and outputs a $(p, g(p))$-treedepth coloring of the actual graph G in a distributed manner. The function g may however be different from the given bound f on the expansion of the class $\mathcal{G}$, which may itself be different from the "best" expansion function for the class $\mathcal{G}$.

4 Distributed Model Checking of Local Formulas

We have now all the ingredients to solve the open problem in [68].

Theorem 4.1. *Let Λ be a finite set. For every local FO formula $\varphi(x)$ on Λ-labeled graphs, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a deterministic algorithm that, for every n-node network $G = (V, E) \in \mathcal{G}$ and every labeling $\ell : V \to 2^\Lambda$, marks all vertices $v \in V$ such that $(G, \ell) \models \varphi(v)$, in $\mathcal{O}(\log n)$ rounds in the $\mathsf{CONGEST}$ model.*

Our algorithm runs on labeled graphs $(G, \ell) \in \mathcal{G}[\Lambda]$. Initially, every node $v \in V$ of an n-node labeled graph $(G, \ell) \in \mathcal{G}[\Lambda]$ is only given its identifier $\mathsf{id}(v)$ on $\mathcal{O}(\log n)$ bits, which is unique in the graph, and its labels $\ell(u) \in 2^\Lambda$ (in addition to a polynomial upper bound on the number n of vertices, as specified in the $\mathsf{CONGEST}$ model). At the end of the execution of our algorithm, every node v outputs $\top$ if $(G, \ell) \models \varphi(v)$, and $\perp$ otherwise.

We prove Theorem 4.1 by adapting the (centralized) quantifier elimination technique of [73] (see also [21, 46]) to the distributed setting. Let (G, ℓ) be a labeled graph with $G \in \mathcal{G}$ and $\ell : V \to 2^\Lambda$, on which we aim at checking a local formula $\varphi(x)$ at every node. Quantifier elimination proceeds by induction on the depth of the formula. The formula considered at any stage of the induction may thus have several free variables $\overline{x}$ with $x_1 = x$. We therefore describe a distributed algorithm that, in $\mathcal{O}(\log n)$ rounds, transforms both the formula $\varphi(\overline{x})$ and the node-labeling ℓ into a new formula $\widehat{\varphi}(\overline{x})$ and a new node labeling $\widehat{\ell}$ such that:

- $\widehat{\varphi}(\overline{x})$ is quantifier-free, but uses new unary and binary predicates,
- $\widehat{\ell}$ uses a new set of labels $\widehat{\Lambda} \supseteq \Lambda$ for the nodes of G, using additional labels involved in the new unary and binary predicates, and
- the transformation preserves locality, and, for any tuple $\overline{v}$ of $|\overline{x}|$ vertices of G, we have:

$$(G, \ell) \models \varphi(\overline{v}) \iff (G, \widehat{\ell}) \models \widehat{\varphi}(\overline{v}).$$

Note that, while the transformation preserves locality, it might be the case that φ is r-local for some r, where as $\widehat{\varphi}$ is r'-local for some $r' \geq r$. Quantifiers are eliminated one by one, and the main difficulty consists in the elimination of the existential quantifiers. As in [73], the proof works in three steps.

- First, the transformation from (φ, ℓ) to $(\widehat{\varphi}, \widehat{\ell})$ is performed on rooted forests of constant depth (cf. Subsection 4.1). This is the most technical part of the quantifier elimination process.
- Second, the quantifier elimination is performed on graphs of bounded treedepth (cf. Subsection 4.2), by reducing the analysis of graphs of bounded treedepth to the analysis of rooted forests. This is where the set of labels is enriched.
- Third, the analysis of graphs of bounded expansion is reduced to the analysis of graphs of bounded treedepth thanks to a $(p, f(p))$-treedepth coloring (cf. Subsection 4.3), for an appropriate p. This is the steps for which the preservation of the locality property needs particular care.

We emphasis that, at each step of the transformation of the original formula φ, and the original labeled graph (G, ℓ), the computations must be efficiently performed in the CONGEST model. Typically, one needs to make sure that adding new labels to the graph can be done efficiently in CONGEST. The rooted forests involved in Subsection 4.1, as well as the graphs of bounded treedepth involved in Subsection 4.2, are subgraphs of the communication graph G, and thus all communications passing through the edges of the former can be implemented along the edges of the latter.

4.1 Rooted Forests of Bounded Depth

In this section we adapt the quantifier elimination procedure from [73] for rooted forests of constant depth (i.e., every tree in the forest is rooted and has constant depth) to the case of local formulas, and show how to implement this procedure in CONGEST. The basic idea for eliminating the quantifiers is to encode information about the satisfiability of the formula in node-labels. This approach does increase the size of the label set. Nevertheless, by taking advantage of the simple structure of FO formulas on forests of constant depth, it is possible to bound the number of labels by a constant. This constant depends solely on the size of the formula, and on the depth of the trees.

As the locality property focuses on distances between nodes, the diameter of each tree plays an important role. Therefore, instead of considering forests with depth at most d for some d, we consider forests in which each tree has diameter at most $d-1$ for some d. For every $d \in \mathbb{N}$, let us denote by $\mathcal{F}_d$ the set of forests in which each tree has diameter at most $d-1$. Note that, in particular, all forests of $\mathcal{F}_d$ have depth at most d. For a set of labels Λ, let us denote by $\mathcal{F}_d[\Lambda]$ the set of all Λ-labeled graphs (G, ℓ) such that $G \in \mathcal{F}_d$.

4.1.1 First-Order Formulas on Forests of Bounded Depth

Let Λ be a (finite) set of labels, and let d be a positive integer. It is convenient to consider a specific way of encoding formulas, suitable for $\mathcal{F}_d[\Lambda]$. For each $i \in \{-1, 0, 1, \ldots, d-1\}$, we define the predicate $\mathsf{lca}_i(x, y)$ (for least common ancestor) as the predicate with free variables x and y that holds if the path P_x from x to the root of the tree containing x, and the path P_y from y to the root of the tree containing y share exactly $i+1$ nodes. Moreover, we set $\mathsf{lca}_{-1}(x, y)$ to be true whenever x and y are in different trees. Note that $\mathsf{lca}_i(x, x)$ holds if and only if x is a node at depth i in its tree. In particular, $\mathsf{lca}_0(x, x)$ holds if and only if x is a root. The predicates lca_i are called *lca-predicates*. As opposed to the label predicates, for which it may be the case that $\mathsf{lab}_\lambda(x)$ and $\mathsf{lab}_{\lambda'}(x)$ both hold for two different labels $\lambda \neq \lambda'$, we have that, for every x and y, $\mathsf{lca}_i(x, y)$ holds for exactly one index $i \in \{-1, 0, 1, \ldots, d-1\}$.

A formula $\varphi \in FO[\Lambda]$ is said to be lca-expressed if it contains solely predicates of the form $\mathsf{lca}_i(x, y)$ (lca predicates), and $\mathsf{lab}_\lambda(x)$ (label predicates). In other words, in a lca-expressed formula, there are no predicates of type $\mathsf{adj}(x, y)$ or $x = y$. The following is a direct consequence of the definitions.

Lemma 4.2. *Every formula $\varphi \in \mathrm{FO}[\Lambda]$ to be evaluated in $\mathcal{F}_d[\Lambda]$ can be lca-expressed by replacing every equality predicate $x = y$ by*

$$\bigvee_{i \in [0, d-1]} \Big(\mathsf{lca}_i(x, x) \wedge \mathsf{lca}_i(x, y) \wedge \mathsf{lca}_i(y, y) \Big),$$

and every adjacency predicate $\mathsf{adj}(x, y)$ by

$$\bigvee_{i \in [0, d-2]} \Big(\mathsf{lca}_i(x, y) \wedge \big((\mathsf{lca}_i(x, x) \wedge \mathsf{lca}_{i+1}(y, y)) \vee (\mathsf{lca}_{i+1}(x, x) \wedge \mathsf{lca}_i(y, y)) \big) \Big).$$

Given a fixed set Λ of labels, a formula $\varphi \in \mathrm{FO}[\Lambda]$ is called *lca-reduced* if (1) it is lca-expressed, and (2) it is quantifier-free. In other words, an lca-reduced formula is a Boolean combination of label and lca predicates on free variables.

Definition 4.3. *Let $\overline{x} = (x_1, \ldots, x_k)$ be a k-tuple of variables. For any two functions $\gamma : [k] \to 2^\Lambda$, and $\delta : [k] \times [k] \to [-1, d-1]$, let us define the formula*

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

as the lca-reduced formula defined by the conjunction of the following two predicates:

$$\text{Predicate 1: } \forall i \in \{1, \ldots, k\}, \bigwedge_{\lambda \in \gamma(i)} \mathsf{lab}_\lambda(x_i) \wedge \bigwedge_{\lambda \in \Lambda \smallsetminus \gamma(i)} \neg \mathsf{lab}_\lambda(x_i),$$

and

$$\text{Predicate 2: } \forall (i, j) \in \{1, \ldots, k\}^2,\ \mathsf{lca}_{\delta(i,j)}(x_i, x_j).$$

We call such predicate lca-type, and we denote by $\mathsf{Type}(k, d, \Lambda)$ the set of all lca-type predicates with k variables.

Observe that the size of $\mathsf{Type}(k,d,\Lambda)$ is at most $2^{k|\Lambda|}(d+1)^{k^2}$, and that it can be computed from k, d and Λ. Note also that, given γ and δ, it may be the case that the formula $\mathsf{type}_{\gamma,\delta}(\overline{x})$ is always false. This is typically the case when the function δ yields a conjunction of incompatible lca predicates (e.g., $\delta(i,j) = 5$, $\delta(i,i) = 2$, and $\delta(j,j) = 1$, or merely any function δ such that $\delta(i,j) \neq \delta(j,i)$). In such cases, i.e., when $\mathsf{type}_{\gamma,\delta}(\overline{x})$ is always false, we say that the formula is *trivial*.

For any non-trivial lca-type formulas $\varphi(\overline{x}) = \mathsf{type}_{\gamma,\delta}(\overline{x})$, we have that δ and γ induce a labeled forest $(F_\varphi, \ell_\varphi) \in \mathcal{F}_d[\Lambda]$, as follows. Each $i \in [k]$ is identified with a node u_i in F_φ, and each leaf of F_φ belongs to the k-tuple $\overline{u}_\varphi = (u_1,\dots,u_k)$. Furthermore, for every $(i,j) \in [k]\times[k]$, $\mathsf{lca}_{\delta(i,j)}(u_i,u_j)$ holds in F_φ. Finally, for every $i \in [k]$, $\ell_\varphi(u_i) = \gamma(i)$, and $\ell_\varphi(v) = \emptyset$ whenever $v \notin \overline{u}_\varphi$.

For any labeled forest (F,ℓ) with $F = (V,E)$, any integer k, and any set of vertices $\overline{v} \in V^k$, let us denote by $F[\overline{v}]$ the subgraph of F induced by all the nodes in $\overline{v}$, and their ancestors.

Lemma 4.4. *Let $(F,\ell) \in \mathcal{F}_d[\Lambda]$ with $F = (V,E)$, and let $\varphi(\overline{x}) = \mathsf{type}_{\gamma,\delta}(\overline{x})$ be a non-trivial lca-type formula with k free variables. For every $\overline{v} = (v_1,\dots,v_k) \in V^k$, $(F,\ell) \models \varphi(\overline{v})$ if and only if there exists a graph isomorphism f between $F[\overline{v}]$ and F_φ such that, for every $i \in [k]$, $\ell_\varphi(f(v_i)) = \ell(v_i)$.*

Proof. Recall that $\overline{u}_\varphi = (u_1,\dots,u_k)$ includes the leaves of F_φ. Let $\overline{v} = (v_1,\dots,v_k)$ be a set of k vertices of F. Suppose first that $(F,\ell) \models \varphi(\overline{v})$. Then, for every $(i,j) \in [k]\times[k]$,

$$\gamma(i) = \ell(v_i), \text{ and } \mathsf{lca}(v_i,v_j) = \delta(i,j).$$

Let f be the function that, for every $i \in [k]$, $f(v_i) = u_i$, and, for every $r \in \{1,\dots,\delta(i,i)-1\}$, f maps the ancestor of v_i in F at distance r from v_i to the corresponding ancestor of u_i in F_φ. By construction f is an isomorphism between $F[\overline{v}]$ and F_φ. Moreover, for every $i \in [k]$, we do have $\ell_\varphi(f(v_i)) = \ell_\varphi(u_i) = \gamma(i) = \ell(v_i)$, as claimed.

Conversely, let us assume that there exists a graph isomorphism f between $F[\overline{v}]$ and F_φ such that, for every $i \in [k]$, $\ell_\varphi(f(v_i)) = \ell(v_i)$. Then we directly infer that, for every $(i,j) \in [k]\times[k]$, $\gamma(i) = \ell(v_i)$ and $\mathsf{lca}(v_i,v_j) = \delta(i,j)$. This implies that $(F,\ell) \models \varphi(\overline{v})$, as claimed. □

The following statement establishes that every lca-reduced formula can be expressed as a disjunction of lca-types.

Lemma 4.5. *For every lca-reduced formula $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$, there exists a set $I \subseteq \mathsf{Type}(|\overline{x}|,d,\Lambda)$, and a formula*

$$\varphi'(\overline{x}) = \bigvee_{\psi\in I} \psi(\overline{x})$$

such that, for every $(F,\ell) \in \mathcal{F}_d[\Lambda]$, $\mathsf{true}(\varphi,F,\ell) = \mathsf{true}(\varphi',F,\ell)$. Moreover, $\varphi'(\overline{x})$ can be computed given only d, Λ, and φ. The formula $\varphi'(\overline{x})$ is called the basic normal form *of φ.*

Proof. Let $\varphi(\overline{x})$ be an lca-reduced formula with k free variables, and let $(F,\ell) \in \mathcal{F}_d[\Lambda]$. Let $\overline{v} = (v_1,\dots,v_k)$ be a tuple of nodes such that $(F,\ell) \models \varphi(\overline{v})$. Let us define the function

$$\delta_{\overline{v}} : [k]\times[k] \to [-1, d-1]$$

by $\delta_{\overline{v}}(i,j) = h$ if the path from v_i to the root of the tree of $F[\overline{v}]$ containing v_i, and the path from v_j to the root of the tree of $F[\overline{v}]$ containing v_j share exactly $h+1$ nodes. We also define the function $\gamma_{\overline{v}}$ by

$$\gamma_{\overline{v}}(i) = \ell(v_i)$$

for every $i \in [k]$. We obtain that

$$(F, \ell) \models \mathsf{type}_{\gamma_{\overline{v}}, \delta_{\overline{v}}}(\overline{v}).$$

Now let $I \subseteq \mathsf{Type}(k, d, \Lambda)$ be the set of all possible lca-types $\mathsf{type}_{\gamma_{\overline{v}}, \delta_{\overline{v}}}$ that can be defined from a labeled forest $(F, \ell) \in \mathcal{F}_d$, and a tuple of vertives $\overline{v} \in \mathsf{true}(\varphi, F, \ell)$. By construction, the function

$$\varphi'(\overline{x}) = \bigvee_{\psi \in I} \psi(\overline{x})$$

satisfies $\mathsf{true}(\varphi, F, \ell) = \mathsf{true}(\varphi', F, \ell)$ for every $(F, \ell) \in \mathcal{F}_d[\Lambda]$. $\square$

Example. Let us consider the following formula using labels in $\{0, 1\}$, hence the two predicates lab_0 and lab_1:

$$\varphi(x_1, x_2, x_3) = \mathsf{adj}(x_1, x_2) \wedge \mathsf{adj}(x_1, x_3) \wedge \mathsf{lab}_1(x_1) \wedge \mathsf{lab}_0(x_2)) \wedge \mathsf{lab}_0(x_3),$$

which holds when x_1 is adjacent to x_2, x_1 is labeled 1, and x_2 and x_3 are both labeled 0. This formula can be rewritten as the lca-reduced formula

$$\tilde{\varphi}(x_1, x_2, x_3) = \left(\bigvee_{i \in [0, d-2]} \xi_i^1(x_1, x_2, x_3) \right) \vee \left(\bigvee_{i \in [0, d-3]} \xi_i^2(x_1, x_2, x_3) \right) \vee \left(\bigvee_{i \in [0, d-3]} \xi_i^3(x_1, x_2, x_3) \right)$$

where

$$\begin{aligned} \xi_i^1(x_1, x_2, x_3) = \Big(& \mathsf{lca}_i(x_1, x_1) \wedge \mathsf{lca}_{i+1}(x_2, x_2) \wedge \mathsf{lca}_{i+1}(x_3, x_3) \wedge \\ & \mathsf{lca}_i(x_1, x_2) \wedge \mathsf{lca}_i(x_1, x_3) \wedge \mathsf{lca}_i(x_2, x_3) \wedge \\ & \mathsf{lab}_1(x_1) \wedge \mathsf{lab}_0(x_2) \wedge \mathsf{lab}_0(x_3) \Big), \end{aligned}$$

$$\begin{aligned} \xi_i^2(x_1, x_2, x_3) = \Big(& \mathsf{lca}_{i+1}(x_1, x_1) \wedge \mathsf{lca}_i(x_2, x_2) \wedge \mathsf{lca}_{i+2}(x_3, x_3) \wedge \\ & \mathsf{lca}_i(x_1, x_2) \wedge \mathsf{lca}_{i+1}(x_1, x_3) \wedge \mathsf{lca}_i(x_2, x_3) \wedge \\ & \mathsf{lab}_1(x_1) \wedge \mathsf{lab}_0(x_2) \wedge \mathsf{lab}_0(x_3) \Big), \end{aligned}$$

and

$$\begin{aligned} \xi_i^3(x_1, x_2, x_3) = \Big(& \mathsf{lca}_{i+1}(x_1, x_1) \wedge \mathsf{lca}_{i+2}(x_2, x_2) \wedge \mathsf{lca}_i(x_3, x_3) \wedge \\ & \mathsf{lca}_{i+1}(x_1, x_2) \wedge \mathsf{lca}_i(x_1, x_3) \wedge \mathsf{lca}_i(x_2, x_3) \wedge \\ & \mathsf{lab}_1(x_1) \wedge \mathsf{lab}_0(x_2) \wedge \mathsf{lab}_0(x_3) \Big). \end{aligned}$$

Observe that $\tilde{\varphi}$ is the basic normal form of φ. Indeed, for every $i \in [0, d-2]$, ξ_i^1 is an lca-type formula, and, for every $i \in [0, d-3]$, ξ_i^2 and ξ_i^3 are lca-type formulas. For instance, $\xi_i^1 = \mathsf{type}_{\gamma, \delta}$ with

$$\gamma(1) = \{1\},\ \gamma(2) = \{0\},\ \gamma(3) = \{0\},$$

and with

$$\delta(1,1)=i,\ \delta(2,2)=i+1,\ \delta(3,3)=i+1,\ \delta(1,2)=i,\ \delta(1,3)=i \text{ and } \delta(2,3)=i.$$

Figure 2 depicts $F_{\xi_i^1}$, $F_{\xi_i^2}$, and $F_{\xi_i^3}$. If a labeled forest (F,ℓ) with label set $\{0,1\}$ satisfies φ, then F contains a forest isomorphic to $F_{\xi_i^1}$, $F_{\xi_i^2}$ or $F_{\xi_i^3}$, for some $i \in [0, d-2]$.

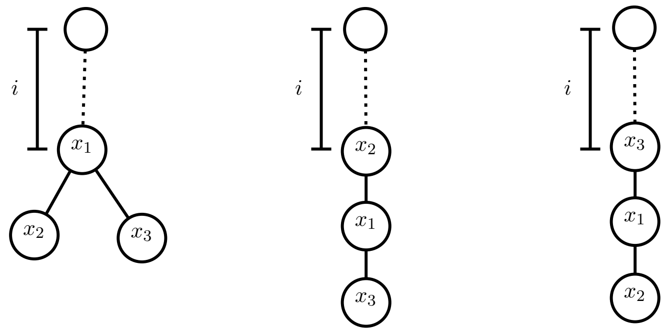

Figure 2: $F_{\xi_i^1}$ (left), $F_{\xi_i^2}$ (middle), and $F_{\xi_i^3}$ (right).

4.1.2 Quantifier Elimination of lca-Expressed Formulas

In this section, we show that it is possible to eliminate existential quantifiers on lca-expressed formulas, by augmenting the set of labels.

Definition 4.6. *Let Λ be a finite set of labels, and let d be a positive integer. A formula $\varphi(\overline{x}) \in$ FO$[\Lambda]$ is said* reducible on $\mathcal{F}_d[\Lambda]$ *if there exists a set $\widehat{\Lambda}$, and an lca-reduced formula $\widehat{\varphi}(\overline{x}) \in$ FO$[\widehat{\Lambda}]$ such that (1) $\widehat{\Lambda}$ and $\widehat{\varphi}$ depend only on Λ, φ, and d, and (2) for every $(F,\ell) \in \mathcal{F}_d[\Lambda]$, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ of F such that*

$$\mathit{true}(\varphi, F, \ell) = \mathit{true}(\widehat{\varphi}, F, \widehat{\ell}).$$

Each element in the triplet $(\widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ of Definition 4.6 is called the *reduction on $\mathcal{F}_d[\Lambda]$* of Λ, φ, and ℓ, respectively. We define a similar notion for local formulas.

Definition 4.7. *Let Λ be a finite set of labels, and let d be a positive integer. A formula $\varphi(\overline{x}) \in FO[\Lambda]$ is said* r-local reducible on $\mathcal{F}_d[\Lambda]$ *if there exists a set $\widehat{\Lambda}$, and an lca-reduced formula $\widehat{\varphi}(\overline{x}) \in$ FO$[\widehat{\Lambda}]$ such that (1) $\widehat{\Lambda}$ and $\widehat{\varphi}$ depend only on Λ, φ, and d, and (2) for every $(F,\ell) \in \mathcal{F}_d[\Lambda]$, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ of F such that*

$$\mathit{true}(\varphi, F, \ell) = \mathit{true}(\widehat{\varphi}, F, \widehat{\ell}) = \mathit{true}(\widehat{\varphi} \wedge \mathrm{ball}(\overline{x}, x_1, r), F, \widehat{\ell}).$$

In Definition 4.7, one insists on the fact that the reduction $\widehat{\varphi}$ of φ must be r-local. This is called a *r-local reduction.*

Remark. The fact that both Definitions 4.6 and 4.7 require $\widehat{\Lambda}$ and $\widehat{\varphi}$ to depend only on Λ, φ and d is not crucial for deciding (local) formulas, but will be used for the design of our certification scheme later in the paper (the prover gives $\widehat{\Lambda}$ and $\widehat{\varphi}$ to the nodes as constant-size certificates).

The following lemma is at the core of quantifier elimination.

Lemma 4.8. *Every formula $\varphi(\overline{x}) = \exists \overline{y}\ \zeta(\overline{x}, \overline{y})$ in* FO[Λ]*, where ζ is an* lca*-reduced formula, is reducible on $\mathcal{F}_d[\Lambda]$.*

Proof. Without loss of generality, let us assume that $|x| = k \geq 1$. (In the case where φ has no free-variables, we can create a formula $\varphi(z) = \varphi$ for a dummy variable z.) We only deal with the case where $|y| = 1$, i.e., with the elimination of a single quantified variable. Indeed, for the case where $|y| > 1$ we can iteratively eliminate the quantifiers one by one, until we obtain a quantifier-free (actually an lca-reduced) formula. Using Lemma 4.2 we can assume without loss of generality that ζ is lca-reduced. Moreover, thanks to Lemma 4.5, we can also assume that $\zeta(\overline{x}, y)$ is expressed in the basic normal form. That is, there exist $I \subseteq \mathsf{Type}(k+1, d, \Lambda)$ such that $\zeta(\overline{x}, y) = \bigvee_{\psi \in I} \psi(\overline{x}, y)$. Since the existential quantifier commutes with the disjunction, we get that

$$\varphi(\overline{x}) = \bigvee_{\psi \in I} \exists y\ \psi(\overline{x}, y).$$

Let us fix an arbitrary labeled graph $(F, \ell) \in \mathcal{F}_d[\Lambda]$. Let us first assume that each formula $\psi \in I$ is reducible on $\mathcal{F}_d[\Lambda]$ (we show how to do this reduction later in the proof). Under this assumption, let us denote by $\widehat{\Lambda}^{\psi}$, $\widehat{\psi}$, and $\widehat{\ell}^{\psi}$ the resulting reductions. We can relabel every set Λ^{ψ}, for $\psi \in I$, so that the sets $\widehat{\Lambda}^{\psi} \smallsetminus \Lambda$ are pairwise disjoint. Let us then define:

$$\widehat{\Lambda} = \bigcup_{\psi \in I} \widehat{\Lambda}^{\psi},\ \widehat{\varphi} = \bigvee_{\psi \in I} \widehat{\psi},\ \text{and, for every node } u \in F,\ \ \widehat{\ell}(u) = \bigcup_{\psi \in I} \widehat{\ell}^{\psi}(u).$$

We get that $\widehat{\Lambda}$, $\widehat{\varphi}$, and $\widehat{\ell}$ are reductions on $\mathcal{F}_d[\Lambda]$ of Λ, φ, and ℓ, respectively. Therefore, it just remains to prove that every $\psi \in \mathsf{Type}(k+1, d, \Lambda)$ is reducible.

Let γ and δ be such that $\psi(\overline{x}, y) = \mathsf{type}_{\gamma,\delta}(\overline{x}, y) \in \mathsf{Type}(k+1, d, \Lambda)$, where y is identified with variable numbered $k+1$ in the definitions of γ and δ. Let $s \in [k]$ such that $\delta(s, k+1)$ is maximum (this value exists because φ has at least one free variable). We define

$$h = \delta(s, k+1),\ h_s = \delta(s, s),\ \text{and}\ h_y = \delta(k+1, k+1).$$

In other words, x_s is at depth h_s, y is of depth h_y, and the least common ancestor of x_s and y (if exists) is at depth h in F_{ψ}. We consider three cases, depending on the value of $\delta(s, k+1)$, and on the relative values of h_y and h.

Case 1: $\delta(s, k+1) \geq 0$ and $h_y = h$. In this case y is an ancestor of x_s in F_{ψ}. Let $(F, \ell) \in \mathcal{F}_d[\Lambda]$. A node w at depth h is called a *candidate* if $\ell(w) = \gamma(k+1)$. A node at depth h_s having an ancestor that is a candidate is called a *good node*. Let $\widehat{\Lambda}^{\psi} = \Lambda \cup \{\mathsf{good}\}$. We define a $\widehat{\Lambda}^{\psi}$-labeling $\widehat{\ell}^{\psi}$ of F as follows:

$$\widehat{\ell}^{\psi}(u) = \begin{cases} \ell(u) \cup \{\mathsf{good}\} & \text{if } u \text{ is a good node,} \\ \ell(u) & \text{otherwise.} \end{cases}$$

Let us now construct an lca-reduced formula $\widehat{\psi}$ that is equivalent to ψ. Let us denote by $\widehat{\gamma} : [k] \to 2^{\Lambda}$ and $\widehat{\delta} : [k] \times [k] \to [-1, d-1]$ the restrictions of γ and δ to $[k]$ (i.e., for each $i, j \in [k]$, $\widehat{\gamma}(i) = \gamma(i)$ and $\widehat{\delta}(i,j) = \delta(i,j)$). Let $\xi = \mathsf{type}_{\widehat{\gamma},\widehat{\delta}}$. Since $h_y = h$, we have that $F_{\xi} = F_{\psi}$, and $\ell_{\xi}(\overline{u}_{\xi}) = \ell_{\psi}(\overline{u}_{\psi})$. Now, given a tuple of vertices $\overline{v} = (v_1, \ldots, v_k)$, we have that $(F, \ell) \models \exists y, \psi(\overline{v}, y)$ if and only if the following conditions hold:

- $F[\overline{v}]$ is isomorphic to F_ψ (which equals F_ξ),
- for every $i \in [k]$, $\ell(v_i) = \gamma(i) = \widehat{\gamma}(i)$, and
- the ancestor of v_s at level h in $F[\overline{v}]$ is labeled $\gamma(k+1)$, i.e., v_s is a good node.

From Lemma 4.4 we obtain that the first two conditions imply that $(F, \ell) \models \psi(\overline{v})$. The third condition can be expressed as $\mathsf{good} \in \ell(v_1)$. Therefore, $\widehat{\psi}$ is defined as the conjunction of the two previous expressions, that is,

$$\widehat{\psi}(\overline{x}) = \mathsf{type}_{\widehat{\gamma},\widehat{\delta}}(\overline{x}) \wedge \mathsf{lab}_{\mathsf{good}}(x_s).$$

Therefore, $\mathsf{true}(\psi, F, \ell) = \mathsf{true}(\widehat{\psi}, F, \widehat{\ell}^\psi)$ for every Λ-labeled forest (F, ℓ) of depth at most d.

Case 2: $\delta(s, k+1) \geq 0$ and $h_y > h$. Let $(F, \ell) \in \mathcal{F}_d[\Lambda]$. In this case, a node v at depth h_y is called a *candidate* if $\ell(v) = \gamma(k+1)$. A node v at depth greater than h is called a *good node* if at least one of its descendants is a candidate node. Finally, a node v at depth h is called a *pivot*, and we denote by $\kappa(v)$ the number of children of v that are good nodes.

We define $\widehat{\gamma}$, $\widehat{\delta}$, and $\xi = \mathsf{type}_{\widehat{\gamma},\widehat{\delta}}$ the same way than in Case 1. Under the assumptions of Case 2, F_ξ is however not necessarily equal to F_ψ. In fact F_ξ is a sub-forest of F_ψ. Indeed, F_ψ can be obtained from F_ξ by adding a path that connects a pivot node in F_ψ to a candidate node not in F_ψ.

Given a tuple $\overline{v}$ of k vertices, checking whether $F[\overline{v}]$ is isomorphic to F_ξ can be achieved by merely using formula ξ. Let z be the pivot in $F[\overline{v}]$, that is, z is the ancestor of v_s at level h. Let $\kappa(z, \overline{v})$ be the set of good children of z contained in $F[\overline{v}]$. To eliminate the existential quantifier in $\varphi(\overline{x})$, it is sufficient to check whether one can extend $F[\overline{v}]$ by plugging a path from z to a candidate node not contained in $F[\overline{v}]$. This condition holds if $\kappa(z, \overline{v}) < \kappa(z)$, and it is exactly what is checked when considering the formula $\widehat{\psi}$.

Let us define $\widehat{\Lambda}^\psi = \Lambda \cup [0, k+1] \cup \{\mathsf{good}\}$, and let us consider the $\widehat{\Lambda}^\psi$-labeling $\widehat{\ell}^\psi$ of F defined as follows. Initially, for every node u of F, $\widehat{\ell}^\psi(u) = \ell(u)$. We then update the label $\widehat{\ell}^\psi(u)$ of every node u as follows.

- If u is a node at depth h_s, then let us denote v the ancestor of u at depth h. The label λ is added to $\widehat{\ell}^\psi(u)$ where
$$\lambda = \begin{cases} \kappa(v) & \text{if } \kappa(v) \leq k \\ k+1 & \text{otherwise.} \end{cases}$$
- If u is a good node, the label good is added to $\widehat{\ell}^\psi(u)$.

Observe that the two conditions above are not necessarily exclusive, and it may be the case that both labels λ and good are added to the set of labels of a the same node u.

Let us suppose that there exists an lca-reduced $\widehat{\Lambda}^\psi$-formula $\alpha(\overline{x})$ that checks whether $\kappa(z) > \kappa(z, \overline{x})$ where z is the ancestor at level h of x_s. We can then define $\widehat{\psi}$ as the lca-reduced $\widehat{\Lambda}^\psi$-formula

$$\widehat{\psi}(\overline{x}) = \mathsf{type}_{\widehat{\gamma},\widehat{\delta}}(\overline{x}) \ \wedge \ \alpha(\overline{x}),$$

where

$$\alpha(\overline{x}) = \bigvee_{q \in [p]} \Bigg[\Bigg(\bigvee_{q \leq m \leq k+1} \mathsf{lab}_m(x_s) \Bigg) \wedge \bigvee_{S \in K(p, q-1)} \Bigg(\bigwedge_{i \in S} \bigvee_{j \in X_i} \mathsf{lab}_{\mathsf{good}}(x_j) \wedge \bigwedge_{i \notin S} \bigwedge_{j \in X_i} \neg \mathsf{lab}_{\mathsf{good}}(x_j) \Bigg) \Bigg].$$

We claim that $\mathsf{true}(\psi, F, \ell) = \mathsf{true}(\widehat{\psi}, F, \widehat{\ell}^{\psi})$ for every Λ-labeled forest $(F, \ell) \in \mathcal{F}_d$. Indeed, observe first that $\kappa(z, \overline{x}) \leq k$ as the forest F_ξ has degree at most k (because it has at most k leaves). Therefore, $\alpha(\overline{x})$ holds whenever $\kappa(z) > k$, which is equivalent to the predicate $\mathsf{lab}_{k+1}(x_s)$. If $\kappa(z) \leq k$ then the value of $\kappa(z)$ is stored in the label of x_s. Let us denote by $z_1, \dots, z_p$ the $p \leq k$ children of z in F_ψ. For every $i \in [p]$, let us denote by X_i the set of variables in $\overline{x}$ that are descendants of z_i. For every $q \in [p]$, we also denote by $K(p, q)$ the set of all subsets of $[p]$ of at most q elements. If $q = \kappa(z)$ and $t = \kappa(z, \overline{x})$, then checking $q > t$ can be done by checking the existence of set $S \in K(p, q-1)$ satisfying that, for every $i \in S$, X_i contains a good node, and, for every $i \notin S$, X_i does not contain a good node.

Case 3: $\delta(s, k+1) = -1$**.** In this case, the node y in F_ψ has no common ancestors with any of the nodes $x_1, \dots, x_k$. In other words, y is in a tree of F_ψ that is different from any of the trees containing $x_1, \dots, x_k$. Let $(F, \ell) \in \mathcal{F}_d[\Lambda]$. Like in previous cases, a node u at depth h_y is called a *candidate* if $\ell(u) = \gamma(k+1)$. A root r of F is said to be *active* if some descendant of r is a candidate node. A node is called *good* if it is a descendant of an active root.

Let $\widehat{\gamma}$, $\widehat{\delta}$ and $\xi = \mathsf{type}_{\widehat{\gamma}, \widehat{\delta}}$ be defined as in Case 1. In Case 3, F_ξ is a sub-forest of F_ψ, where F_ψ can be obtained from F_ξ by adding a path from an active root r not contained in F_ξ to a candidate node.

Let $\overline{v} = (v_1, \dots, v_k)$ be a set of vertices of F, and let $\rho(F)$ be the number of active roots in F. Similarly, let $\rho(F, \overline{v})$ be the number of active roots in $F[\overline{v}]$. It follows that $(F, \ell) \models \exists y, \psi(\overline{v}, y)$ if and only if $F[\overline{v}]$ is isomorphic to F_ξ, $\ell(v_i) = \gamma(i)$ for each $i \in [k]$, and F contains an active root not contained in $F[\overline{v}]$. The first two conditions can be checked using ξ, and the latter is equivalent to $\rho(F) > \rho(F, \overline{v})$.

Let $\widehat{\Lambda} = \Lambda \cup [0, k+1] \cup \{\mathsf{good}\}$ be a set of labels, and let us consider the $\widehat{\Lambda}^{\psi}$-labeling $\widehat{\ell}^{\psi}$ of F defined as follows. Initially, for every node u in F, let us set $\widehat{\ell}^{\psi}(u) = \ell(u)$. We then update the label $\widehat{\ell}^{\psi}(u)$ of every node u as follows.

- For every node u, the label λ is added to $\widehat{\ell}^{\psi}(u)$, where

$$\lambda = \begin{cases} \rho(F) & \text{if } \rho(F) \leq k \\ k+1 & \text{if } \rho(F) > k. \end{cases}$$

- If u is a good node, then the label good is added $\widehat{\ell}^{\psi}(u)$.

Let us suppose that there exists an lca-reduced $\widehat{\Lambda}^{\psi}$-formula $\alpha(\overline{x})$ that checks whether $\rho(F) > \rho(F, \overline{x})$ where z is the ancestor at level h of x_1. We can then define $\widehat{\psi}$ as the lca-reduced $\widehat{\Lambda}^{\psi}$-formula

$$\widehat{\psi}(\overline{x}) = \mathsf{type}_{\widehat{\gamma}, \widehat{\delta}}(\overline{x}) \ \wedge \ \xi(\overline{x}),$$

where

$$\xi(\overline{x}) = \bigvee_{q \in [p]} \left[\Big(\bigvee_{q \leq m \leq k+1} \mathsf{lab}_m(x_1) \Big) \wedge \bigvee_{S \in K(p, q-1)} \Big(\bigwedge_{i \in S} \bigvee_{j \in X_i} \mathsf{lab}_{\mathsf{good}}(x_j) \wedge \bigwedge_{i \notin S} \bigwedge_{j \in X_i} \neg \mathsf{lab}_{\mathsf{good}}(x_j) \Big) \right] \quad (2)$$

We claim that $\mathsf{true}(\psi, F, \ell) = \mathsf{true}(\widehat{\psi}, F, \widehat{\ell}^{\psi})$ for every Λ-labeled forest $(F, \ell) \in \mathcal{F}_d$. Indeed, observe that F_ξ can contain at most k roots. Therefore α is true if $\rho(F) > k$, which is equivalent the predicate $\mathsf{lab}_{k+1}(x_1)$. If $\rho(F) \leq k$ then the value of $\rho(F)$ is stored in the label of x_1. Let $r_1, \ldots, r_p$ denote the $p \leq k$ roots of F_ξ. For every $i \in [p]$, let X_i be the set of variables in $\overline{x}$ that are descendants of r_i. We obtain that checking whether $q = \rho(F) > \rho(F, \overline{x})$ is equivalent to check the existence of a set $S \in K(p, q-1)$ such that, for every $i \in [p]$, X_i contains a good node if and only if $i \in S$, from which the expression of ξ follows. □

Finally, the following lemma is a refined version of Lemma 4.8 enforcing locality.

Lemma 4.9. *Let $d \in \mathbb{N}$, $r \geq d$, and $\varphi(\overline{x}) = \exists \overline{y} \left(\zeta(\overline{x}, \overline{y}) \wedge \mathrm{ball}(\{\overline{x}, \overline{y}\}, x_1, r)\right)$ be an r-local lca-expressed existential formula in $\mathrm{FO}[\Lambda]$, i.e., ζ is a lca-reduced formula in $\mathrm{FO}[\Lambda]$. It holds that φ is d-local reducible on $\mathcal{F}_d[\Lambda]$.*

Proof. Since all trees in the forests $F \in \mathcal{F}_d$ have diameter at most d, and since $r \geq d$, we get that $\mathrm{ball}(z, x_1, r)$ holds if and only if x_1 and z are in the same tree, which is equivalent to $\neg\mathsf{lca}_{-1}(x_1, z)$. We define:

$$\varphi_1(\overline{x}) = \exists \overline{y}, \zeta(\overline{x}, \overline{y}) \wedge \bigwedge_{z \in \overline{x}, \overline{y}} \neg\mathsf{lca}_{-1}(x_1, z)$$

We have $\mathsf{true}(\varphi, F, \ell) = \mathsf{true}(\varphi_1, F, \ell)$. We can now apply Lemma 4.8 on φ_1. (Note that there were three cases the proof of Lemma 4.8, but the third case cannot happen here as this case corresponds to the case where two variables are not in the same connected components of F, which is forbidden by the terms of the form $\neg\mathsf{lca}_{-1}(x_1, z)$.) By Lemma 4.8, we obtain an lca-reduced formula $\widehat{\varphi}$ and a labeling $\widehat{\ell}$ such that $\mathsf{true}(\varphi_1, F, \ell) = \mathsf{true}(\widehat{\varphi}, F, \widehat{\ell})$. Now, as mentioned above, each x_i is in the same component as x_1. Since each tree has diameter at most d, this mere fact implies that $\mathrm{ball}(x_i, x_1, d)$ holds for every i. As a consequence, $\mathsf{true}(\widehat{\varphi}, F, \widehat{\ell}) = \mathsf{true}(\widehat{\varphi} \wedge \mathrm{ball}(\overline{x}, x_1, d), F, \widehat{\ell})$. □

4.1.3 Quantifier Elimination of lca-Reduced Formulas in CONGEST

We now show that the quantifier elimination on local formulas described in the proof of Lemma 4.9 can be computed in $\mathcal{O}(1)$ rounds in the $\mathsf{CONGEST}$ model.

Lemma 4.10. *Let Λ be a finite set of labels, $d \in \mathbb{N}$ and $r \geq d$. For every r-local lca-expressed existential formula $\varphi(\overline{x}) = \exists \overline{y} \left(\zeta(\overline{x}, \overline{y} \wedge \mathrm{ball}(\{\overline{x}, \overline{y}\}, x_1, r))\right)$ in $\mathrm{FO}[\Lambda]$, there exists a set of labels $\widehat{\Lambda}$, a lca-reduced formula $\widehat{\varphi}$ and a distributed algorithm performing in $\mathcal{O}(1)$ round in the $\mathsf{CONGEST}$ model satisfying the following. For every Λ-labeled graph (G, ℓ), every subgraph $F \in \mathcal{F}_d$ of G:*

- *every node $v \in V(G)$ is given as input: a set of labels $\ell(v) \subseteq \Lambda$, whether it belongs to F, and, if applicable, its depth and parent in F;*
- *every node $v \in V(G)$ outputs a set of labels $\widehat{\ell}(v) \subseteq \widehat{\Lambda}$ such that $\widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell}$ form a d-local reduction of Λ, φ, ℓ.*

Proof. Let $k = |\overline{x}|$. As for the proof of Lemma 4.8, we assume w.l.o.g. that $k \geq 1$, and we treat only the case where $|\overline{y}| = 1$ (as the general case $|\overline{y}| \geq 1$ can be obtained by iteratively eliminating the $|\overline{y}|$ quantifiers one by one). In the same way they are defined in Lemma 4.8, let us define the set $I \subseteq \mathsf{Type}(k+1, D, \Lambda)$, and, for each $\psi \in I$, the label sets $\widehat{\Lambda}^\psi$, the lca-reduced formula $\widehat{\psi}$, and the $\hat{\Lambda}^\psi$-labeling $\widehat{\ell}^\psi$ of F. Furthermore, we define

$$\widehat{\Lambda} = \bigcup_{\psi \in I} \widehat{\Lambda}^\psi,\ \widehat{\varphi} = \bigvee_{\psi \in I} \widehat{\psi},\ \text{and, for every node } u \in F,\ \widehat{\ell}(u) = \bigcup_{\psi \in I} \widehat{\ell}^\psi(u).$$

We then extend $\widehat{\ell}$ to the nodes of $G \setminus F$ by merely defining $\widehat{\ell}(v) = \ell(v)$ for all $v \notin V(F)$. Since $\mathsf{Type}(k+1, d, \Lambda)$ is a finite set, it is enough to show that, for every $\psi \in I$, the label set $\widehat{\Lambda}^{\psi}$, and the formulas $\widehat{\psi}$ and $\widehat{\ell^{\psi}}$ can be computed in $\mathcal{O}(1)$ rounds. In fact, the definitions of the set I, $\widehat{\Lambda}$, and $\widehat{\varphi}$ do not depend in the sub-forest F of G, so I, $\widehat{\Lambda}$, and $\widehat{\varphi}$ can be computed at every node without any communication. It remains to show how to compute, for any given $\psi \in I$, the value of $\widehat{\ell^{\psi}}(v)$ at all nodes v in a constant number of rounds.

Initially the algorithm sets $\widehat{\ell^{\psi}}(v) = \ell(v)$ at every node. Let γ and δ be the functions such that $\psi = \mathsf{type}_{\gamma,\delta}$. Let s, h_s, h_y, and h be as defined in the proof of Lemma 4.8. Observe that every node knows φ, and thus every node can compute ψ, h_s, h_y, and h without any communication. Furthermore, every node knows whether its depth is equal to h, h_s, or h_y, or none of these three values.

We now revisit the three cases treated in the proof of Lemma 4.8.

Case 1: $\delta(s, k+1) \geq 0$ and $h_y = h$. In this case, the algorithm just needs to identify the good nodes in F. To do so, for every $i \in [p]$, every node v at depth h_y in T_i checks whether it is a candidate node. Every such nodes does not need any communication for this, as it knows its depth in T_i, as well as $\ell(v)$ and $\gamma(v)$. Then, the algorithm broadcasts ping messages (on a single bit) downward every tree, from every candidate node. These ping messages are forwarded by their descendants downward every T_i, for $h_s - h$ rounds. More precisely, each candidate node sends a ping as a single bit 1 to all its children, and, for $h_s - h - 1$ rounds, each node that receives that ping from its parent forwards it to all its children. Finally, every node u at depth h_s that receives the ping know that their ancestor at level h_y are candidates, and thus they mark themselves as good nodes. Moreover, every such node u sets $\widehat{\ell^{\psi}}(u) = \ell(u) \cup \{\mathsf{good}\}$. Every other node u in F sets $\widehat{\ell^{\psi}}(u) = \ell(u)$. All these operations consume $h_s - h = \mathcal{O}(d)$ rounds.

Case 2: $\delta(1, k+1) \geq 0$ and $h_y > h$. In this case the algorithm is divided in two phases. Similar to previous case, the candidate nodes can identify themselves without any communication. In the first phase, ping messages are broadcast upward in each tree T_i, starting from the candidate nodes, during $h - h_y$ rounds. In other words, the candidate nodes send a single bit to their parents, and, for $h - h_y - 1$ more rounds, each node that receives a ping in the previous round forwards it to its parent. During this phase, every node u at depth $> h$ that receives a ping message is marked as a good node, and it adds good to $\widehat{\ell^{\psi}}(u)$. This phase takes $h - h_y = \mathcal{O}(d)$ rounds. During the second phase, each pivot node z (i.e., every node at depth h) computes $\kappa(z)$ as the number of children that sent a ping message during the last round of the first phase. Then, each pivot node z broadcasts the value $\ell(\kappa(z))$ downward the trees, for $h - h_s$ rounds. After $h - h_s$ rounds, every node u at depth h_s has received the message $\ell(\kappa(z))$ from its pivot ancestor z, and it adds this label to $\widehat{\ell^{\psi}}(u)$. This phase takes $h - h_s = \mathcal{O}(d)$ rounds.

Case 3: $\delta(1, k+1) = -1$. As mentioned in the proof of Lemma 4.9, this case cannot happen for local formulas.

We conclude that for every $\psi \in I$, every node u can compute $\ell^{\psi}(u)$, in $\mathcal{O}(d)$ rounds. Therefore, the total number of rounds sufficient for computing $\widehat{\ell}$ is $\mathcal{O}(d \cdot |I|) = \mathcal{O}(d \cdot 2^{k|\Lambda|}(d+1)^{k^2})$. $\square$

4.2 Quantifier Elimination on Graphs of Bounded Treedepth

Given a graph $G = (V, E)$, a *decomposition forest* F of G is a rooted spanning forest of G satisfying that for every edge $e \in E$, one of its endpoints is an ancestor of the other in F. Observe that if G is connected then any decomposition forest of G is actually a tree, and every DFS tree of G is a decomposition tree of G. For every $t > 0$, any induced subgraph H of G with treedepth at most t is called *a treedepth-t induced subgraph.*

We stress the fact that a decomposition forest F of G must be a subgraph of G, which is very convenient for distributed algorithms because this enables communication along the edges of F. Note that there exist graphs of treedepth t in which any decomposition forest is of diameter $\Omega(2^t)$. This is for example the case of paths on $2^t - 1$ vertices. Nevertheless, graphs of treedepth at most t have paths of at most $2^t - 1$ vertices. In particular, for any class of graphs of bounded treedepth, all possible decomposition forests also have trees of constant (although exponentially larger) diameter.

Proposition 4.11 ([67])**.** *Any decomposition forest of a graph of treedepth at most t belongs to $\mathcal{F}_{2^t}$.*

In the following lemma, we show that any distributed algorithm aiming at checking a quantifier-free formula in a treedepth-t induced subgraph H of a labeled graph G, may rather check an equivalent formula on a decomposition forest of H.

Lemma 4.12. *Let Λ be a finite set of labels, and let $t > 0$. For every quantifier free formula $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$, there exists a finite set $\widehat{\Lambda}$ depending only on t and Λ, and an lca-reduced formula $\widehat{\varphi}(\overline{x}) \in \mathrm{FO}[\widehat{\Lambda}]$ depending on φ, Λ, and t, such that, for every Λ-labeled treedepth-t induced subgraph (H, ℓ) of a Λ-labeled graph (G, ℓ), and for every decomposition forest F of H, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ of G such that*

$$\mathit{true}(\varphi, H, \ell) = \mathit{true}(\widehat{\varphi}, F, \widehat{\ell}).$$

Moreover, there is a 1-round $\mathsf{CONGEST}$ algorithm that computes $\widehat{\ell}$, assuming each node v is given as input $\ell(v)$, whether it belongs to F and, if applicable, its depth and parent in F.

Proof. Each node u initially knows Λ, φ, $\ell(u)$, t, whether it belongs to H, and, if applicable, its depth and parent in F. Node u returns $\widehat{\Lambda}$, $\widehat{\varphi}$, and $\widehat{\ell}(u)$. Let $d = 2^t$, and let us consider a Λ-labeled graph (G, ℓ), a treedepth-t induced subgraph H of G, and a decomposition forest F of H. The idea of the construction is the following. Each node u of F can be labeled by a subset of $[0, d-1]$ indicating the set of levels of F containing a neighbor of u in H. Then, the $\mathsf{adj}(x, y)$ predicate can be transformed into label and lca predicates on x and y. More precisely, for every node $v \in V(H)$, let $\mathsf{Levels}(v) \subseteq [d]$ be the set of levels of F containing ancestors u of v such that $\{u, v\} \in E(H)$. Then, let $\widehat{\Lambda} = \Lambda \cup [0, d-1]$ (we assume w.l.o.g. that $\Lambda \cap [0, d-1] = \emptyset$), and let us consider the $\widehat{\Lambda}$-labeling $\widehat{\ell}$ of G defined by

$$\widehat{\ell}(v) = \begin{cases} \ell(v) \cup \mathsf{Levels}(v) & \text{if } v \in V(F), \\ \ell(v) & \text{otherwise.} \end{cases}$$

For each $i \in [0, d-1]$, we denote by $\mathsf{level}_i(x)$ the predicate $\mathsf{lab}_i(x)$, which is true when x has a neighbor at depth i. Finally, we define $\widehat{\varphi}$ by replacing each $\mathsf{adj}(x, y)$ predicates by

$$\bigvee_{1 \le i < j \le d-1} \left[\Big(\mathsf{lca}_j(x, x) \wedge \mathsf{lca}_i(y, y) \wedge \mathsf{level}_i(x) \Big) \vee \Big(\mathsf{lca}_i(x, x) \wedge \mathsf{lca}_j(y, y) \wedge \mathsf{level}_i(y) \Big) \right],$$

and every equality predicate by lca predicates as we explained in Lemma 4.2. In this way, we obtain a lca-reduced formula $\widehat{\varphi}$ such that $\mathsf{true}(\varphi, H, \ell) = \mathsf{true}(\widehat{\varphi}, F, \ell)$.

Finally, observe that every node u can compute $\widehat{\Lambda}, \widehat{\varphi}$, and $\widehat{\ell}(u)$ in a single communication round. In fact $\widehat{\Lambda}, \widehat{\varphi}$, and $\widehat{\ell}(u)$ for every $u \in V(G) \smallsetminus V(H)$ are computed without any communication, as these values do not depend on the instance. To compute $\widehat{\ell}(u)$ for $u \in V(F)$, each node broadcasts its depth in F to all its neighbors. After this communication round, $\mathsf{Levels}(u)$ can be set as the set of neighbors with a smaller depth in F, and the computation of $\widehat{\ell}(u) = \ell(u) \cup \mathsf{Levels}(u)$ follows. $\square$

We now prove that existential quantifier in formulas on graphs of bounded treedepth can be eliminated efficiently. The reduction results in a lca-reduced formula to be checked on a decomposition forest of the considered graph.

Lemma 4.13. *Let Λ be a set of labels, $\varphi(\overline{x}) = \exists \overline{y}, \zeta(\overline{x}, \overline{y}) \wedge \text{ball}((\overline{x}, \overline{y}), x_1, r)$ be a r-local existential formula in $\text{FO}[\Lambda]$, where $r \in \mathbb{N}$ and ζ is a quantifier free formula. There exists a set of labels $\widehat{\Lambda}$, and an lca-reduced formula $\widehat{\varphi}(\overline{x})$ such that, for every treedepth-t induced subgraph (H, ℓ) of a Λ-labeled graph (G, ℓ), and for every decomposition forest F of H, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ such that:*

$$\mathsf{true}(\varphi, H, \ell) = \mathsf{true}(\widehat{\varphi}, F, \widehat{\ell}) = \mathsf{true}(\widehat{\varphi} \wedge \text{ball}(\overline{x}, x_1, 2^t), F, \widehat{\ell}).$$

Moreover, there exists an algorithm computing $\widehat{\ell}$ in $\mathcal{O}(1)$ $\mathsf{CONGEST}$ rounds, assuming each node v is given as input $\ell(v)$, whether it belongs to F and, if applicable, its depth and parent in F.

The lemma above considers both a graph H of bounded treedepth, and a decomposition forest F. The distances in both structures are not the same, and two vertices may be further apart in F than they are in H. However, the connected components remain the same in both graphs.

Proof. We are assuming that each node u initially knows $\Lambda, \varphi, \ell(u)$, whether it belongs to H, and, if applicable, its depth and parent in F. We consider graphs of treedepth at most t, thus their connected components have diameter at most 2^t. If $r < 2^t$, then, by Lemma 3.5 we can rewrite $\text{ball}(z, x_1, r)$ as an existential formula of the form $\exists \overline{w}, \psi(x_1, z, \overline{w}) \wedge \text{ball}((z, \overline{w}), x_1, 2^t)$. Thus, without loss of generality, we consider the case $r \geq 2^t$. Therefore, the balls of radius r are then exactly the connected components of H.

By Lemma 4.12, there exists an lca-reduced formula $\zeta_1(\overline{x}, \overline{y})$ such that, for every induced labeled subgraph (H, ℓ) of treedepth at most t, for every decomposition forest F of H, there exists a labeling ℓ_1 such that $\mathsf{true}(\zeta, H, \ell) = \mathsf{true}(\zeta_1, F, \ell_1)$, and ℓ_1 can be computed in $\mathcal{O}(1)$ rounds. Moreover, since connected components are the same in H and F, we also have

$$\mathsf{true}(\zeta \wedge \text{ball}((\overline{x}, \overline{y}), x_1, r), H, \ell) = \mathsf{true}(\zeta_1 \wedge \text{ball}((\overline{x}, \overline{y}), x_1, r), F, \ell_1).$$

The proof is completed by applying Lemma 4.10 to the formula $\exists \overline{y}, \zeta_1(\overline{x}, \overline{y}) \wedge \text{ball}((\overline{x}, \overline{y}), x_1, r)$. $\square$

4.3 Graphs of Bounded Expansion

Let us fix a graph class $\mathcal{G}$ of expansion f. Given a set of labels Λ, let $\mathcal{G}[\Lambda]$ be the set of Λ-labeled graphs (G, ℓ) such that $G \in \mathcal{G}$. Let $(G, \ell) \in \mathcal{G}[\Lambda]$ be the input instance. For every integer p, let $\mathsf{Col}(p)$ be the collection of sets $U \subseteq [f(p)]$ with cardinality at most p.

4.3.1 Skeletons

We introduce a central notion for our proof of Theorem 4.1.

Definition 4.14. *Let $p \in \mathbb{N}$, and let G be a graph of expansion f. A* p-skeleton *of G is a triple $S = (G, c, \delta)$ where c is $(p, f(p))$-treedepth coloring of G, and δ is a function mapping every $U \in \mathsf{Col}(p)$ to a decomposition forest F^U of the subgraph $G[U]$ induced by nodes with colors in U. Labeled p-skeletons are pairs (S, ℓ), where $S = (G, c, \delta)$ is a p-skeleton, and ℓ is a labeling of G.*

Note that, by definition of a $(p, f(p))$-treedepth coloring c, the graph $G[U]$ is of treedepth at most p for every $U \in \mathsf{Col}(p)$. Hence, by Proposition 4.11, we have $F^U \in \mathcal{F}_{2^p}$. The following lemma states that a p-skeleton can be computed in $\mathcal{O}(\log n)$ rounds in the $\mathsf{CONGEST}$ model, where the big-$\mathcal{O}$ notation hides constants depending only of p, and of the expansion of the class.

Lemma 4.15. *For every $p \in \mathbb{N}$, there exists an algorithm that constructs a p-skeleton of G in $\mathcal{O}(\log n)$ rounds in $\mathsf{CONGEST}$. Specifically, at the end of the algorithm, each node knows its own color, and, for every set $U \in \mathsf{Col}(p)$ containing its color, its parent and depth in the forest F^U corresponding to the skeleton.*

Remark. In the remaining of the paper, when an algorithm takes a skeleton as input, or outputs a skeleton, it is assumed that the skeleton is represented in a distributed manner, in the way the algorithm in Lemma 4.15 computes it.

Proof. The algorithm of Proposition 3.9 enables to compute a $(p, f(p))$-treedepth coloring c of G, in $\mathcal{O}(\log n)$ rounds. After running this algorithm, each node u knows $c(u)$. Let us order the sets in $\mathsf{Col}(p)$ lexicographically. The rest of the algorithm runs in $N = |\mathsf{Col}(p)| \leq 2^{f(p)}$ phases, where each phase takes at most 2^{2p+1} rounds. The i-th phase consists of dealing with the ith set $U \in \mathsf{Col}(p)$. Let us fix one such set U. For every $u \in V(G)$, if $c(u) \notin U$, then u does not participate in that phase (it remains idle, and, in particular, it does not forward any messages). Only the nodes u such that $c(u) \in U$ exchange messages during phase U. The phase is divided in two steps.

- In the first step, each node $u \in G[U]$ performs 2^p rounds during which it broadcasts the value of a variable $\mathsf{min\text{-}id}(u)$. Initially, $\mathsf{min\text{-}id}(u) = \mathsf{id}(u)$. In the subsequent rounds, $\mathsf{min\text{-}id}(u)$ is the minimum identifier of a node in the component of $G[U]$ containing u (i.e., the minimum identifier received by u from a neighbor in U). Observe that $G[U]$ is a graph of diameter at most 2^p, and thus $\mathsf{min\text{-}id}(u)$ is eventually the smallest identifier in the whole component of $G[U]$ containing u. After the first 2^p rounds of the phase, the node with identifier $\mathsf{min\text{-}id}(u)$ is the root of the tree containing u. Let $V_1, \dots, V_h$ be the components of $G[U]$, and, for every $i \in [h]$, let r_i be the node of minimum identifier $\mathsf{id}(r_i)$ among all nodes in V_i. Note that, after step 1 has completed, every node $u \in V_i$ has $\mathsf{min\text{-}id}(u) = \mathsf{id}(r_i)$.
- In the second step, the algorithm runs in parallel in all components V_i, $i \in \{1, \dots, h\}$ for computing a decomposition tree T_i of $G[U]$ rooted at r_i. For this purpose, we use the algorithm from [31] (cf. Lemma 11 therein). This algorithm performs in 2^{2p} rounds.

At the end of these two steps, every node u knows its depth in its tree T_i, as well as the identifier of its parent in T_i. Observe that the collection of trees $\{T_i \mid i \in [h]\}$ is a decomposition forest F^U of $G[U]$. This completes the description of the phase for set U. The whole algorithm runs in $\mathcal{O}(\log n) + |\mathsf{Col}(p)| \cdot 2^{2p+1} = \mathcal{O}(\log n)$ rounds. ☐

4.3.2 Reducibility on Graphs of Bounded Expansion

We now define a specific notion of quantifier elimination, denoted *reducibility*, for graphs of bounded expansion. For this purpose, similarly to how FO formulas on treedepth-t graphs are transformed into FO formulas on their decomposition forests, we will transform FO formulas on graphs of $\mathcal{G}$ into FO formulas on their p-skeletons. For this purpose, we define additional predicates for describing the structure of p-skeletons.

- For each $k \in [f(p)]$, the predicate $\mathsf{color}_k(x)$ checks whether the considered vertex x has color k in the skeleton.
- For every $U \in \mathsf{Col}(p)$, and every $j \in [-1, d-1]$, the predicate $\mathsf{lca}_j^U(x, y)$ is equivalent to the lca_j predicates, but on F^U. That is, $\mathsf{lca}_j^U(x, y)$ is true whenever x and y are colored by colors in U, and $\mathsf{lca}_j(x, y)$ is true on F^U. Such predicates are called lca^U predicates.

A formula $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$ is called *p-lca-reduced* if it is a quantifier-free, and it contains only label, color and lca^U predicates, for some collection of sets $U \in \mathsf{Col}(p)$. For such a formula, one can ask whether a labeled p-skeleton (S, ℓ) satisfies φ on some vertices $\overline{v} \in S$, i.e., one can ask whether

$$(S, \ell) \models \varphi(\overline{v}),$$

where $S = (G, s, \delta)$ is a labeled p-skeleton. The set $\mathsf{true}(\varphi, S, \ell)$ is then defined accordingly. Note that we abusedly write $\overline{v} \in S$ instead of $\overline{v} \in G$, which itself is an abuse of notation for $\overline{v} \in V$, where V is the set of vertices of G.

Definition 4.16. *A formula $\varphi \in \mathrm{FO}[\Lambda]$ is called* reducible on $\mathcal{G}$ *if there exists a positive integer p, a set of labels $\widehat{\Lambda}$, and a p-lca-reduced formula $\widehat{\varphi}$ such that, for every labeled graph $(G, \ell) \in \mathcal{G}[\Lambda]$, and every p-skeleton S of G, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ of S such that*

$$\textit{true}(\varphi, G, \ell) = \textit{true}(\widehat{\varphi}, S, \widehat{\ell}).$$

Given a graph G and a p-skeleton S, the tuple $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ is called the reduction *of (Λ, φ, ℓ) on S.*

Importantly, when considering local formulas, it is also required that the reduction $\widehat{\varphi}$ be a local formula too.

Definition 4.17. *An r-local formula $\varphi(\bar{x}) \in \mathrm{FO}[\Lambda]$ is said* locally reducible on $\mathcal{G}$ *if there exists $r' \in \mathbb{N}$, such that, for every $(G, \ell) \in \mathcal{G}[\Lambda]$, and every p-skeleton S of G, there exists a reduction $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ of (Λ, φ, ℓ) on S such that:*

$$\textit{true}(\varphi, G, \ell) = \textit{true}(\widehat{\varphi}, S, \widehat{\ell}) = \textit{true}(\widehat{\varphi} \wedge \mathrm{ball}(\overline{x}, x_1, r'), S, \widehat{\ell}).$$

Note that, when considering distance predicates on a skeleton $S = (G, c, \delta)$, we use the distances on the underlying graph G. Our goal is now to show that every (local) formula can be efficiently reduced in $\mathcal{G}$ by a distributed algorithm. We first establish this result for existential formulas.

Lemma 4.18. *Let Λ be a set of labels, and let $\varphi(\overline{x}) = \exists \overline{y}, \psi(\overline{x}, \overline{y}) \wedge \mathrm{ball}((\overline{x}, \overline{y}), x_1, r)$ be an r-local existential formula in $\mathrm{FO}[\Lambda]$, where ψ is a quantifier free formula. For every integer $p \geq r \cdot (|\overline{x}| + |\overline{y}|)$, there exists a set of labels $\widehat{\Lambda}$, and a p-lca-reduced formula $\widehat{\varphi}(\overline{x})$ satisfying that, for every $(G, \ell) \in \mathcal{G}[\Lambda]$, and for every p-skeleton S of G, there exists a $\widehat{\Lambda}$-labeling of $\widehat{\ell}$ such that*

$$\textit{true}(\varphi, G, \ell) = \textit{true}(\widehat{\varphi}, S, \widehat{\ell}) = \textit{true}(\widehat{\varphi} \wedge \mathrm{ball}(\overline{x}, x_1, 2^p), S, \widehat{\ell}).$$

Moreover, there exists an algorithm that computes $\widehat{\ell}$ in $\mathcal{O}(1)$ rounds in $\mathsf{CONGEST}$*, assuming S and ℓ are given as input.*

Proof. Let $p \geq r(|\overline{x}| + |\overline{y}|)$. We apply Lemma 4.13 to φ, which says that there exists a set of labels Λ_1, and a *lca*-reduced formula $\varphi_1(\overline{x})$ such that, for every labeled graph (H, ℓ) of treedepth at most p, and for every decomposition forest F of H, there is a Λ_1 labeling ℓ_1 of F such that

$$\mathsf{true}(\varphi, H, \ell) = \mathsf{true}(\varphi_1, F, \ell_1) = \mathsf{true}(\varphi_1 \wedge \mathrm{ball}(\overline{x}, x_1, 2^p), F, \ell_1).$$

Let us now consider now a p-skeleton $S = (G, s, \delta)$. For each $U \in \mathsf{Col}(p)$, the induced subgraph $G[U]$ has treedepth at most p, and $F^U = \delta(U)$ is a decomposition forest of $G[U]$. Thus, we can compute the corresponding labeling ℓ_1^U in $\mathcal{O}(1)$ rounds. Let us denote by $\widehat{\varphi}^U(\overline{x})$ the p-lca-reduced formula $\varphi_1^U(\overline{x}) \wedge \mathsf{color}_U(\overline{x})$. That is, φ_1^U is the formula φ_1 in which all lca predicates of were replaced by lca^U predicates. Finally, let us define

$$\widehat{\varphi}(\overline{x}) = \bigvee_{U \in \mathsf{Col}(p)} \widehat{\varphi}^U(\overline{x}),$$

and, for every node u,

$$\widehat{\ell}(u) = \bigcup_{U \in \mathsf{Col}(p)} \ell_1^U(u),$$

where, w.l.o.g., we assume that the formulas φ^U use disjoint sets of labels. Let us prove that this formula has the desired property. Let $(G, \ell) \in \mathcal{G}[\Lambda]$, and let S be a p-skeleton of G. We prove the equivalence of the following three statements:

(i) $(G, \ell) \models \varphi(\overline{v})$,

(ii) $(S, \widehat{\ell}) \models \widehat{\varphi}(\overline{v})$, and

(iii) $(S, \widehat{\ell}) \models \widehat{\varphi}(\overline{v}) \wedge \mathrm{ball}(\overline{v}, v_1, 2^p)$.

The implication *(iii)* $\Rightarrow$ *(ii)* is obvious. Let us prove *(i)* $\Rightarrow$ *(iii)*. For this purpose, let us assume that

$$(G, \ell) \models \psi(\overline{v}, \overline{w}) \wedge \mathrm{ball}((\overline{v}, \overline{w}), v_1, r)$$

for some $\overline{v}, \overline{w} \subseteq V(G)$. Since $p \geq r \cdot (|v| + |w|)$, there exists $U \in \mathsf{Col}(p)$ such that $G[U]$ contains both $\overline{v}$ and $\overline{w}$, and also, for every $u \in \overline{v}, \overline{w}$, a path of length at most r from v_1 to u. We get that

$$(G[U], \ell) \models \psi(\overline{v}, \overline{w}) \wedge \mathrm{ball}((\overline{v}, \overline{w}), v_1, r).$$

The first of the following series of implications is a direct consequence of the latter, and the second implication follows from Lemma 4.13:

$$\begin{aligned}
(G, \ell) \models \varphi(\overline{v}) &\implies \exists U \in \mathsf{Col}(p), \overline{v} \subseteq G[U] \wedge (G[U], \ell) \models \varphi(\overline{v}) \\
&\implies \exists U \in \mathsf{Col}(p), \overline{v} \subseteq G[U] \wedge (F^U, \ell_1^U) \models \varphi_1(\overline{v}) \wedge \mathrm{ball}(\overline{v}, v_1, 2^p) \\
&\implies \left[\exists U \in \mathsf{Col}(p), (S, \ell_1^U) \models \widehat{\varphi}^U(\overline{v})\right] \wedge \left[\exists U \in \mathsf{Col}(p), F^U \models \mathrm{ball}(\overline{v}, v_1, 2^p)\right] \\
&\implies \left[(S, \widehat{\ell}) \models \widehat{\varphi}(\overline{v})\right] \wedge [S \models \mathrm{ball}(\overline{v}, v_1, 2^p)] \\
&\implies (S, \widehat{\ell}) \models \widehat{\varphi}(\overline{v}) \wedge \mathrm{ball}(\overline{v}, v_1, 2^p)
\end{aligned}$$

Finally we prove $(ii) \Rightarrow (i)$. We have

$$\begin{aligned}
(S, \widehat{\ell}) \models \widehat{\varphi}(\overline{v}) &\implies \exists U \in \mathsf{Col}(p), (S, \ell_1^U) \models \widehat{\varphi}^U(\overline{v}) \\
&\implies \exists U \in \mathsf{Col}(p), \overline{v} \subseteq G[U] \wedge (F^U, \ell_1^U) \models \varphi_1(\overline{v}) \\
&\implies \exists U \in \mathsf{Col}(p), \overline{v} \subseteq G[U] \wedge (G[U], \ell) \models \varphi(\overline{v}) \\
&\implies (G, \ell) \models \varphi(\overline{v})
\end{aligned}$$

where the penultimate implication follows from Lemma 4.13, and the last from the fact that φ is existential (i.e., if the quantified variables exist in a subgraph, then they exist in G). This completes the proof. $\square$

Lemma 4.19. *Let $p \in \mathbb{N}$, let Λ be a set of labels, and let $\varphi(\overline{x})$ be a p-lca-reduced formula. There exists a set of labels $\widehat{\Lambda}$, and there exists a quantifier free formula $\psi(\overline{x}, \overline{y})$ such that, for every $G \in \mathcal{G}$, for every Λ-labeling ℓ of G, and for every p-skeleton S of G, there exists a labeling $\widehat{\ell}$ of G for which*

$$\textsf{\textit{true}}\big((\varphi \wedge \mathrm{ball}(\overline{x}, x_1, r)), S, \ell\big) = \textsf{\textit{true}}(\widehat{\varphi}, G, \widehat{\ell})$$

where

$$\widehat{\varphi}(\overline{x}) = \exists \overline{y}\, \big(\psi(\overline{x}, \overline{y}) \wedge \mathrm{ball}((\overline{x}, \overline{y}), x_1, r + 2^p)\big).$$

Moreover, $\widehat{\ell}$ can be computed without any communication, assuming S and ℓ are given as inputs.

The main idea of the proof is to replace each lca^U-predicate of φ by an existential FO formula on the graph G underlying the (labeled) p-skeleton (S, ℓ). To that end, we augment the labeling ℓ with additional labels, which encode the structure of S. Finally, we add distance predicates to enforce the locality of the resulting formula.

Proof. Let us assume that φ is written in conjunctive normal form (CNF). Let $d = 2^p$. We have that

$$\neg \mathsf{lca}_i(x, y) \equiv \neg \mathsf{color}_U(x) \vee \neg \mathsf{color}_U(y) \vee \bigvee_{j \in [-1, d] \setminus \{i\}} \mathsf{lca}_j(x, y).$$

Therefore, we can assume that φ does not have any predicate of the form $\neg \mathsf{lca}_i(x, y)$, as it suffices to replace each occurrence of such predicates by the corresponding disjunction of predicates. Note that the formula we obtain is still in conjunctive normal form. It only remains to replace each predicate $\mathsf{lca}_i^U(x, y)$ by an equivalent existential formula: since φ is in conjunctive normal form and no lca^U predicate is negated, the formula we obtain is existential. To that end, we add to Λ new labels representing the structure of the p-skeleton: let

$$\widehat{\Lambda} = \Lambda \cup \{\mathsf{color}_k, k \in [f(p)]\} \cup \{\mathsf{depth}_k^U, U \in \mathsf{Col}(p), k \in [2^p]\}.$$

We assume w.l.o.g. that Λ does not already contain these labels. We construct the labeling $\widehat{\ell}$ as follows, without any communication. Each node v adds its own color to its labeling, and, for each $U \in \mathsf{Col}(p)$ that contains v's color, node v adds its depth in F^U to its label.

Consider now a predicate $\mathsf{lca}_i^U(y,z)$. We define a formula $\zeta_i^U \in \mathrm{FO}[\widehat{\Lambda}]$ such that $(S,\ell) \models \mathsf{lca}_i^U(u,v) \iff (G,\widehat{\ell}) \models \zeta_i^U(u,v)$.

$$\zeta_i^U(y,z) = \exists(y_0,\dots,y_d)\,\exists(z_0,\dots,z_d) \bigvee_{(a,b)\in[\max\{0,i\},d-1]^2} \Bigg((y_a = y) \wedge (z_b = z) \wedge \Big(\bigwedge_{s\in[0,i]} y_s = z_s \Big) \wedge \Big(\bigwedge_{s\in[i+1,\min(a,b)]} y_s \neq z_s \Big) \wedge \mathsf{depth}_0^U(y_0) \wedge \mathsf{depth}_0^U(z_0) \wedge \Big(\bigwedge_{s\in[1,a]} \mathsf{depth}_s^U(y_s) \wedge \mathsf{adj}(y_{s-1},y_s) \Big) \wedge \Big(\bigwedge_{s\in[1,b]} \mathsf{depth}_s^U(z_s) \wedge \mathsf{adj}(z_{s-1},z_s) \Big) \Bigg).$$

Indeed, a (resp. b) is the depth of y (resp. z) in F^U, and $y_0,\dots,y_a$ (resp. $z_0,\dots,z_b$) correspond to the path from the root of y's tree to y (resp. from the root of z's tree to z). The equality predicates on the first line, and the depth predicates on the second line guarantee that the endpoints of the path are as expected. The equality predicates on the second line guarantee that both paths share exactly $i+1$ vertices. The depth and adjacency predicates on the third line guarantee that, for every $s \in [1,a]$ (resp., every $s \in [1,b]$), y_{s-1} (resp., z_{s-1}) is the parent of y_s (resp., z_s). By construction, we can assume that each of the y_i (resp., z_i) we introduced are at distance at most d from y (resp., z).

We would like to replace each occurrence of lca_i^U in φ by ζ_i^U, while preserving locality. To that end, we add a distance predicate to the formula ζ_i^U, which enforces locality. Recall that we are interested by the formula $\varphi \wedge \mathrm{ball}(\overline{x},x_1,r)$. Whenever the formula is satisfied, every x_i is at distance at most r from x_1. This implies that every y_s and z_s introduced in $\zeta_i^U(y,z)$ is at distance at most $r+d$ from x_1. We thus replace each $\mathsf{lca}_i^U(y,z)$ predicate in φ by the following formula:

$$\exists(y_0,\dots,y_d)\,\exists(z_0,\dots,z_d) \Bigg(\mathrm{ball}((\overline{y},\overline{z}),x_1,r+d) \wedge \bigvee_{(a,b)\in[\max(0,i),2^p]^2} \Bigg[(y_a = y) \wedge (z_b = z) \wedge \Big(\bigwedge_{s\in[0,i]} y_s = z_s \Big) \wedge \Big(\bigwedge_{s\in[i+1,\min\{a,b\}]} y_s \neq z_s \Big) \wedge \mathsf{depth}_0^U(y_0) \wedge \mathsf{depth}_0^U(z_0) \wedge \Big(\bigwedge_{s\in[1,a]} \big(\mathsf{depth}_s^U(y_s) \wedge \mathsf{adj}(y_{s-1},y_s)\big) \Big) \wedge \Big(\bigwedge_{s\in[1,b]} \big(\mathsf{depth}_s^U(z_s) \wedge \mathsf{adj}(z_{s-1},z_s)\big) \Big) \Bigg] \Bigg)$$

Let us denote by $\tilde{\varphi}(\overline{x})$ this formula. By the definition of $\beta_{r,r'}$ in Proposition 3.5, we can then define

$$\widehat{\varphi}(\overline{x}) = \tilde{\varphi}(\overline{x}) \wedge \beta_{r,r+d}(\overline{x}) \wedge \mathrm{ball}(\overline{x},x_1,r+d).$$

By construction, $\widehat{\varphi}$ is existential, and thus

$$\mathsf{true}(\widehat{\varphi},G,\widehat{\ell}) = \mathsf{true}(\widehat{\varphi},G,\widehat{\ell}) \cap \mathsf{true}(\mathrm{ball}(\overline{x},x_1,r),G,\widehat{\ell}) = \mathsf{true}(\varphi \wedge \mathrm{ball}(\overline{x},x_1,r),S,\ell),$$

which completes the proof. $\square$

We now tackle the general case.

Lemma 4.20. *Let Λ be a set of labels, and let $\varphi(\overline{x})$ be an FO formula in r-local form. There exists $p \in \mathbb{N}$, a set of labels $\widehat{\Lambda}$, and a p-lca-reduced formula $\widehat{\varphi}(\overline{x})$ such that, for every labeled graph $(G, \ell) \in \mathcal{G}[\Lambda]$, and for every p-skeleton S of G, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ for which*

$$\textit{true}(\varphi, G, \ell) = \textit{true}(\widehat{\varphi}, S, \widehat{\ell}) = \textit{true}(\widehat{\varphi} \wedge \mathrm{ball}(\overline{x}, x_1, 2^p), S, \widehat{\ell})$$

Moreover $\widehat{\ell}$ can be computed in $\mathcal{O}(\log n)$ rounds in the CONGEST *model, assuming S and ℓ are given as inputs.*

Proof. The proof proceeds by induction on the structure of the formulas written in local form as defined (in an inductive way) in Definition 3.3. We consider the three cases occurring in Definition 3.3.

Case 1 (base case). $\varphi(\overline{x}) = \psi(\overline{x}) \wedge \mathrm{ball}(\overline{x}, x_1, r)$, where ψ is quantifier free. We can directly apply Lemma 4.18 with $|\overline{y}| = 0$, to compute $\widehat{\ell}$ from the skeleton S and labeling ℓ given as inputs.

Case 2. $\varphi(\overline{x}) = \exists y\ \psi(\overline{x}, y)$, where ψ is in local form. By induction, there exists p', Λ_1, and $\psi_1(\overline{x}, y)$ such that, for every $(G, \ell) \in \mathcal{G}[\Lambda]$, and every p'-skeleton S' of G, there exists a Λ_1-labeling ℓ_1 for which

$$\mathsf{true}(\psi, G, \ell) = \mathsf{true}(\psi_1, S', \ell_1) = \mathsf{true}(\psi_1 \wedge \mathrm{ball}((\overline{x}, y), x_1, 2^{p'}), S', \ell_1)$$

By Lemma 4.15, a p'-skeleton S' of G can be constructed in $\mathcal{O}(\log n)$ rounds in CONGEST. By induction, the corresponding labeling ℓ_1 can be computed in $\mathcal{O}(\log n)$ rounds. For eventually applying Lemma 4.18 on $\exists y\, \psi_1(\overline{x}, y)$, we first apply Lemma 4.19 on ψ_1, with $r = 2^{p'}$. Thanks to this lemma, there exists a set of labels Λ_2, a quantifier-free formula $\rho_2(\overline{x}, y, \overline{z})$, and a labeling ℓ_2 that can be constructed from ℓ_1 such that

$$\mathsf{true}(\psi_1 \wedge \mathrm{ball}((\overline{x}, y), x_1, 2^{p'}), S', \ell_1) = \mathsf{true}(\psi_2(\overline{x}, y), G, \ell_2),$$

where $\psi_2(\overline{x}, y) = \exists \overline{z}\, \rho_2(\overline{x}, y, \overline{z}) \wedge \mathrm{ball}((\overline{x}, y, \overline{z}), x_1, 2^{p'+1})$. Let us then define $\varphi_2(\overline{x}) = \exists y\, \psi_2(\overline{x}, y)$. This is formula is existential since ψ_2 is existential. We can thus apply Lemma 4.18 to φ_2, from which the induction step follows. In particular, we can compute $\widehat{\ell}$ from the skeleton S given as input.

Case 3. $\varphi(\overline{x}) = \mathrm{ball}(\overline{x}, x_1, r) \wedge \forall y \left(\neg \mathrm{ball}(\overline{x}, y), x_1, r) \vee \psi(\overline{x}, y)\right)$ where ψ is in r-local form. We can rewrite φ as

$$\varphi(\overline{x}) = \mathrm{ball}(\overline{x}, x_1, r) \wedge \neg \exists y, \mathrm{ball}((\overline{x}, y), x_1, r) \wedge \neg \psi(\overline{x}, y).$$

By induction on ψ, there exist $p' \in \mathbb{N}$, a set of labels Λ_1, and a p'-lca-reduced formula $\psi_1(\overline{x}, y)$ such that, for every labeled graph $(G, \ell) \in \mathcal{G}[\Lambda]$, and for every p'-skeleton S' of G, there exists a Λ_1-labeling ℓ_1 satisfying

$$\mathsf{true}(\psi, G, \ell) = \mathsf{true}(\psi_1, S', \ell_1).$$

By Lemma 4.15, a p'-skeleton S' of G can be constructed in $\mathcal{O}(\log n)$ rounds in the CONGEST model. By induction, the labeling ℓ_1 can thus be computed in $\mathcal{O}(\log n)$ rounds. Let us then take the negation of ψ_1. Note that the resulting formula $\neg\psi_1$ is still a p'-lca-reduced formula. We then apply Lemma 4.19 to $\neg\psi_1$, which results in a set of labels Λ_2, an existential formula ρ_2 in $(r+2^p)$-local form, and a Λ_2-labeling ℓ_2 of G such that

$$\mathsf{true}(\neg\psi_1 \wedge \text{ball}(\overline{x}, x_1, r), S', \ell_1) = \mathsf{true}(\rho_2, G, \ell_2).$$

As a consequence, we get

$$\begin{aligned}(G,\ell) \models \text{ball}((\overline{v},u), v_1, r) \wedge \neg\psi(\overline{v},u) &\iff \Big(G \models \text{ball}((\overline{v},u),v_1,r)\Big) \wedge \Big((S',\ell_1) \models \neg\psi_1(\overline{v},u)\Big)\\ &\iff (S',\ell_1) \models \text{ball}((\overline{v},u),v_1,r) \wedge \neg\psi_1(\overline{v},u)\\ &\iff (G,\ell_2) \models \rho_2(\overline{v},u)\end{aligned}$$

where the first equivalence is by induction, and the last by application of Lemma 4.19.

Let us now focus on the formula

$$\eta_2(\overline{x}) = \exists y\ \rho_2(\overline{x}, y).$$

Since ρ_2 is existential, and since it is in $(r+2^{p'})$-local form, the formula η_2 is also existential and in $(r+2^{p'})$-local form. By Lemma 4.18, there exists an integer $p \geq r \cdot |\overline{x}|$, a set of labels Λ_3, and a p-*lca*-reduced-formula $\eta_3(\overline{x})$ such that, for every p-skeleton S of G (and in particular the skeleton S given as input), there exists a labeling ℓ_3 for which

$$\mathsf{true}(\eta_2, G, \ell_2) = \mathsf{true}(\eta_3, S, \ell_3).$$

Once again, the negation of η_3 is still p-*lca*-reduced.

We also need to transform the first half of φ, namely ball$(\overline{x}, x_1, r)$, into a p-*lca*-reduced formula. The formula ball$(\overline{x}, x_1, r)$ is in fact an existential formula in r-local form (with zero existential quantifiers). Therefore, we can apply Lemma 4.18. We can even apply this lemma with the same p since p satisfies the required condition $p \geq r|\overline{x}|$. Thus, there exists a set of label Λ_γ, and a p-lca-reduced formula $\gamma(\overline{x})$ such that, for every $G \in \mathcal{G}$, and every p-skeleton S of G, there exists a Λ_γ-labeling ℓ_γ for which

$$\mathsf{true}(\text{ball}(\overline{x}, x_1, r)), G) = \mathsf{true}(\gamma, S, \ell_\gamma) = \mathsf{true}(\gamma \wedge \text{ball}(\overline{x}, x_1, 2^p), S, \ell_\gamma).$$

Finally, let $\widehat{\Lambda} = \Lambda_\gamma \cup \Lambda_3$ where, w.l.o.g., we assume that Λ_γ and Λ_3 are disjoint, let

$$\widehat{\varphi}(\overline{x}) = \gamma(\overline{x}) \wedge \neg\eta_3(\overline{x}),$$

and for every node v let $\widehat{\ell}(v) = \ell_\gamma(v) \cup \ell_3(v)$. We get

$$\begin{aligned}\mathsf{true}(\varphi, G, \ell) &= \mathsf{true}\big(\text{ball}(\overline{x}, x_1, r), G, \ell\big) \cap \mathsf{true}\Big(\neg\exists y\ \big(\text{ball}((\overline{x},y), x_1, r) \wedge \neg\psi(\overline{x},y)\big), G, \ell\Big)\\ &= \mathsf{true}\big(\text{ball}(\overline{x}, x_1, r), G, \ell_2\big) \cap \mathsf{true}(\neg\eta_2, G, \ell_2)\\ &= \mathsf{true}\big(\gamma(\overline{x}), S, \ell_\gamma\big) \cap \mathsf{true}\big(\neg\eta_3(\overline{x}), S, \ell_3\big)\\ &= \mathsf{true}\big(\widehat{\varphi}, S, \widehat{\ell}\big).\end{aligned}$$

Moreover, since $\mathsf{true}(\gamma, S, \ell_\gamma) = \mathsf{true}(\gamma \wedge \text{ball}(\overline{x}, x_1, 2^p), S, \ell)$, we also have

$$\mathsf{true}\big(\widehat{\varphi}, S, \widehat{\ell}\big) = \mathsf{true}\big(\widehat{\varphi} \wedge \text{ball}(\overline{x}, x_1, 2^p), S, \widehat{\ell}\big).$$

We complete the proof by studying the round-complexity of computing the labeling $\widehat{\ell}$ in CONGEST. For each of the lemmas used in the proof, the construction of the labeling is performed in $\mathcal{O}(1)$ rounds. For cases 2 and 3, we also need to construct a skeleton, which can be done in $\mathcal{O}(\log n)$ rounds. Thus, the overall round complexity is $\mathcal{O}(\log n)$, as claimed. □

4.3.3 Proof of Theorem 4.1

Let $\varphi(x)$ be a r-local formula. By Lemma 3.4, it can be rewritten in r-local form. Then, by Lemma 4.20, for some $p \in \mathbb{N}$ φ can be reduced in $\mathcal{O}(\log n)$ rounds to a p-lca-reduced formula $\widehat{\varphi}(x)$. Using Lemma 4.15, we can build a p-skeleton S in $\mathcal{O}(\log n)$ rounds. Then, from an input labeling ℓ, the reduced labeling $\widehat{\ell}$ can be computed in $\mathcal{O}(\log n)$ rounds. It satisfies:

$$(G, \ell) \models \varphi(v) \iff (S, \widehat{\ell}) \models \widehat{\varphi}(v)$$

Since $\widehat{\varphi}(x)$ is quantifier free, each node v can check locally without additional communication whether it satisfies $\widehat{\varphi}$, which completes the proof.

5 Distributed Model Checking of General Formulas

In this section, we establish Theorem 1.5, generalized to labeled graphs, as formaly stated below.

Theorem 5.1. *Let Λ be a finite set. For every FO formula φ on Λ-labeled graphs, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a distributed algorithm that, for every n-node network $G = (V, E) \in \mathcal{G}$ of diameter D, and every $\ell : V \to 2^{\Lambda}$, decides whether $(G, \ell) \models \varphi$ in $\mathcal{O}(D + \log n)$ rounds under the* $\mathsf{CONGEST}$ *model.*

Actually, we show a result stronger than Theorem 5.1. Given a formula $\varphi(x)$ with one free variable, we provide a CONGEST algorithm that marks all nodes of n-node graphs with bounded expansion that satisfy the formula, in $\mathcal{O}(D + \log n)$ rounds, where D is the diameter of the input graph[7]. To prove the result, we need to revisit the proof of Theorem 4.1, which involves quantifier elimination of existential formulas first over forests of bounded-depth, then over graphs of bounded tree-depth, and finally over graphs with bounded expansion. Fortunately, once we have resolved quantifier elimination over forests bounded-depth, the extensions to the other two cases will follow from the arguments already developed in the previous section for local formulas. Therefore, it will be sufficient to revisit quantifier elimination for existential formulas on forests of bounded-depth, and then proceed by induction to cover the general case.

5.1 Rooted Forests of Bounded Depth

As established in the proof of Lemma 4.8, an existential formula over bounded-depth forests corresponds to checking the presence of "subpatterns" (representing subforests) in which the free variables of the formula can be mapped onto. One of these free variables, quantified existentially, is to be eliminated. Depending on the position of this existentially quantified variable y, we distinguish three cases. In the first two case, y shares common ancestors with the other free variables — these cases are the "connected cases". In the third case, y belongs to a connected component that is different from all the other variables in the formula. When dealing with local formulas, we only needed to handle the connected cases. However, for general formulas, we must also address the remaining case. This is for this "disconnected case" that our algorithm requires $\mathcal{O}(D)$ additional rounds.

[7]This problem is more challenging than simply checking whether the graph satisfies a formula φ, i.e., stronger than the model checking problem as, given an FO formula φ without free variables, one can define a logically equivalent formula $\varphi'(x)$ by adding a dummy free variable, so that the graph satisfies φ if and only if it satisfies $\varphi'(v)$ at each of its nodes v.

Lemma 5.2. *Let $\varphi(\overline{x}) = \exists\overline{y}\ \zeta(\overline{x},\overline{y})$ be a formula in* FO$[\Lambda]$*. There exists $\widehat{\Lambda}$, a* lca*-reduced formula $\widehat{\varphi}(\overline{x})$ and a distributed algorithm performing in $\mathcal{O}(D)$ rounds in the* CONGEST *model satisfying the following. For every Λ-labeled graph (G,ℓ) of diameter D, and for every subgraph $F \in \mathcal{F}_d$ of G,*

- *every node $v \in V(G)$ is given as input its labeling $\ell(v)$, whether it belongs to F, and whether it is a root of a tree in F (if not, it also receives as input its depth and parent in F);*
- *every node $v \in V(G)$ outputs a set of labels $\widehat{\ell}(v)$, such that $\widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell}$ form a reduction of Λ, φ, ℓ.*

Proof. We follow the same guidelines as in the proof of Lemma 4.10 for Cases 1 and 2. It is thus sufficient to consider Case 3, as described in Lemma 4.8. We use the same notations as in these lemmas. Recall that Case 3 assumes

$$\delta(1, k+1) = -1.$$

The algorithm consists in four phases.

1. During the first phase, the algorithm aims at identifying the active roots. First, the candidate nodes are identified in the same way as in the previous cases 1 and 2. By broadcasting a ping message upward each tree from the candidate nodes for h_y rounds, each root becomes aware of the presence of a candidate node in its tree. The roots that receive a ping message mark themselves as active roots. This phase takes $\mathcal{O}(d)$ rounds.

2. During the second phase, the algorithm aims at identifying the good children, by broadcasting a ping message downward from every active root. Every node u that receives such ping message on this phase marks itself as good, and adds the label good to $\widehat{\ell^{\psi}}(u)$. This phase takes $\mathcal{O}(d)$ rounds. At this point, the algorithm constructs a BFS tree T of G, rooted at node r_1. This procedure takes $\mathcal{O}(D)$ rounds.

3. During the third phase, the root r_1 gets the number $\rho(F)$ of active roots in F, by counting upward T. More specifically, let us denote by $d_T = \mathcal{O}(D)$ the depth of T. The counting proceeds in $d_T + 1$ round. For each node u in T, let $a(u)$ be 1 if u is an active root, and 0 otherwise. In the first round, each node u at depth d_T set $\mathsf{val}(u) = a(u)$, and sends $\mathsf{val}(u)$ to its parent in T. Then, at round $i+1$ with $i \in [d_t]$, every node u of T at depth $d_T - i$ computes $\mathsf{val}(u) = a(u) + \sum_{v \in C(u)} \mathsf{val}(u)$, where $C(u)$ is the set of children of u in T, and sends $\mathsf{val}(u)$ to its parent. At the end of this phase, we have $\mathsf{val}(r_1) = \rho(F)$. Observe that, for every node u, $\mathsf{val}(u) \le n$, and thus this value can be communicated in one round under the CONGEST model. Therefore, the third phase runs in $\mathcal{O}(D) + d_T + 1 = \mathcal{O}(D)$ rounds.

4. Finally, during the fourth phase, the root r_1 broadcast $\rho(F)$ downward the tree T, to all nodes in G. Then, each node u in F adds the label $\min(\rho(F), k+1)$ to $\widehat{F^{\psi}}(u)$. This phase takes $d_T = O(D)$ rounds.

We conplete the proof by noticing that, for every $\psi \in I$, every node u can compute $\ell^{\psi}(u)$ in $\mathcal{O}(D)$ rounds. Therefore, the total number of rounds sufficient for computing $\widehat{\ell}$ is

$$|I| \cdot D \le c \cdot 2^{k|\Lambda|}(d+1)^{k^2} \cdot D,$$

where c is a constant not depending on either d, Λ, ψ, or D. □

5.2 Quantifier Elimination on Graphs of Bounded Treedepth

We now deal with the quantifier elimination on graphs of bounded treedepth.

Lemma 5.3. *Let Λ be a set of labels, and let $\varphi(\overline{x}) = \exists\overline{y}, \zeta(\overline{x}, \overline{y})$ be an existential formula in* $\mathrm{FO}[\Lambda]$*, where ζ is a quantifier free formula. There exists a set of labels $\widehat{\Lambda}$, and a* lca*-reduced formula $\widehat{\varphi}(\overline{x})$ such that, for every induced subgraph (H, ℓ) of treedepth t of a Λ-labeled graph (G, ℓ), and for every decomposition forest F of H, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ such that:*

$$\mathsf{true}(\varphi, H, \ell) = \mathsf{true}(\widehat{\varphi}, F, \widehat{\ell})$$

Moreover, there exists an algorithm computing $\widehat{\ell}$ in $\mathcal{O}(D)$ rounds in the $\mathsf{CONGEST}$ *model, each node u is given as input $\ell(u)$, whether it belongs to H and, if applicable, its depth and parent in F.*

Proof. By Lemma 4.12, there exists an lca-reduced formula $\zeta_1(\overline{x}, \overline{y})$ such that, for every induced labeled subgraph (H, ℓ) of treedepth at most t, for every decomposition forest F of H, there exists a labeling ℓ_1 such that $\mathsf{true}(\zeta, H, \ell) = \mathsf{true}(\zeta_1, F, \ell_1)$, and ℓ_1 can be computed in $\mathcal{O}(1)$ rounds in the $\mathsf{CONGEST}$ model. It is thus sufficient to merely apply Lemma 5.2 to the formula $\exists\overline{y}, \zeta_1(\overline{x}, \overline{y})$, which takes $\mathcal{O}(D)$ $\mathsf{CONGEST}$ rounds. □

5.3 Graphs of Bounded Expansion

We can now directly treat the case of graphs of bounded expansion. As we did in the local case, we consider first the case of existential formulas. Let $\mathcal{G}$ be a class of graphs, of bounded expansion.

Lemma 5.4. *Let Λ be a set of labels, and let $\varphi(\overline{x}) = \exists\overline{y}\, \psi(\overline{x}, \overline{y})$ be an existential formula in* $\mathrm{FO}[\Lambda]$*, where ψ is quantifier-free. For every $p \geq |\overline{x}| + |\overline{y}|$, there exists a set $\widehat{\Lambda}$ of labels, and a p-*lca*-reduced formula $\widehat{\varphi}$ such that, for every $(G, \ell) \in \mathcal{G}[\Lambda]$, and for every p-skeleton S of G, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ such that*

$$\mathsf{true}(\varphi, G, \ell) = \mathsf{true}(\widehat{\varphi}, S, \widehat{\ell}).$$

Moreover, there exists a distributed algorithm that computes $\widehat{\ell}$ in $\mathcal{O}(D)$ rounds in $\mathsf{CONGEST}$*, assuming ℓ and S are given as input.*

Proof. Let us fix an integer $p \geq |\overline{x}| + |\overline{y}|$. By applying Lemma 5.3 to φ, we get that there exist a set of labels Λ_1, and an *lca*-reduced formula $\varphi_1(\overline{x})$ such that, for every labeled graph (H, ℓ) of treedepth at most p, and for every decomposition forest F of H, there exists a Λ_1-labeling ℓ_1 of F such that

$$\mathsf{true}(\varphi, H, \ell) = \mathsf{true}(\varphi_1, F, \ell_1).$$

Let us now consider a p-skeleton S of (G, ℓ). For each $U \in \mathsf{Col}(p)$, the induced subgraph $G[U]$ has treedepth at most p, and F^U is a decomposition forest of $G[U]$. Thus, the corresponding labeling ℓ_1^U can be computed in $\mathcal{O}(D)$ rounds. Let us consider the p-lca-reduced formula

$$\widehat{\varphi}^U(\overline{x}) = \varphi_1^U(\overline{x}) \wedge \mathsf{color}_U(\overline{x}).$$

Note that φ_1^U is the formula φ_1 in which all lca predicates were replaced by lca^U predicates. We also define

$$\widehat{\varphi}(\overline{x}) = \bigvee_{U \in \mathsf{Col}(p)} \widehat{\varphi}^U(\overline{x}).$$

Finally, for each node u, we set

$$\widehat{\ell}(u) = \bigcup_{U \in \mathsf{Col}(p)} \ell_1^U(u),$$

where it is assumed, w.l.o.g., that the formulas φ^U use disjoint sets of labels. Let us now prove that the formula $\widehat{\varphi}(\overline{x})$ has the desired property. Let $(G,\ell) \in \mathcal{G}[\Lambda]$, let S be a p-skeleton of G, and let us prove that

$$(G,\ell) \models \varphi(\overline{v}) \iff (S,\widehat{\ell}) \models \widehat{\varphi}(\overline{v}).$$

For this purpose, let us assume that $(G,\ell) \models \psi(\overline{v},\overline{w})$ for some $\overline{v},\overline{w} \subseteq V(G)$. Since $p \geq |v| + |w|$, there exists $U \in \mathsf{Col}(p)$ such that $G[U]$ contains both $\overline{v}$ and $\overline{w}$. Therefore, $(G[U],\ell) \models \psi(\overline{v},\overline{w})$. Furthermore,

$$\begin{aligned}
(G,\ell) \models \varphi(\overline{v}) &\iff \exists U \in \mathsf{Col}(p), \overline{v} \subseteq G[U], (G[U],\ell) \models \varphi(\overline{v}) \\
&\iff \exists U \in \mathsf{Col}(p), \overline{v} \subseteq G[U], (F^U,\ell_1^U) \models \varphi_1(\overline{v}) \\
&\iff \exists U \in \mathsf{Col}(p), (S,\ell_1^U) \models \widehat{\varphi}^U(\overline{v}) \\
&\iff (S,\widehat{\ell}) \models \widehat{\varphi}(\overline{v})
\end{aligned}$$

where the second equivalence is thanks to Lemma 5.3. $\square$

Next, we adapt Lemma 4.19 to the case of general (i.e., non necessarily local) formulas.

Lemma 5.5. *Let $p \in \mathbb{N}$, let Λ be a set of labels, and let $\varphi(\overline{x})$ be a p-lca-reduced formula. There exists a set of labels $\widehat{\Lambda}$, and a quantifier free formula $\psi(\overline{x},\overline{y})$ such that, for every $G \in \mathcal{G}$, every p-skeleton S of G, and every Λ-labeling ℓ of G, there exists a labeling $\widehat{\ell}$ of G for which*

$$\mathit{true}(\varphi,S,\ell) = \mathit{true}(\widehat{\varphi},G,\widehat{\ell}).$$

Moreover, for every p, Λ and φ, there exists a distributed algorithm that, given S and ℓ as inputs, computes $\widehat{\ell}$ without any communication.

Proof. This lemma is a direct consequence of the arguments in the proof of Lemma 4.19. Indeed, we follow the same steps as in that proof, until the predicates $\mathsf{lca}_i^U(y,z)$ has to be replace by the similar existential formula

$$\begin{aligned}
\exists(y_0,\dots,y_d)\,\exists(z_0,\dots,z_d) \bigvee_{(a,b)\in[i,2^p-1]^2} \Bigg(& (y_a = y) \wedge (z_b = z) \\
& \wedge \Big(\bigwedge_{s\in[0,i]} y_s = z_s\Big) \wedge \Big(\bigwedge_{s\in[i+1,\min(a,b)]} y_s \neq z_s\Big) \wedge \mathsf{depth}_0^U(y_0) \wedge \mathsf{depth}_0^U(z_0) \\
& \wedge \Big(\bigwedge_{s\in[1,a]} \mathsf{depth}_s^U(y_s) \wedge \mathsf{adj}(y_{s-1},y_s)\Big) \wedge \Big(\bigwedge_{s\in[1,b]} \mathsf{depth}_s^U(z_s) \wedge \mathsf{adj}(z_{s-1},z_s)\Big)\Bigg)
\end{aligned}$$

This completes the proof. $\square$

We are now ready to deal with quantifier elimination of general first-order formulas.

Lemma 5.6. *Let Λ be a set of labels, and let $\varphi(\overline{x})$ be a formula in $\mathrm{FO}[\Lambda]$. There exists $p \in \mathbb{N}$, a set of labels $\widehat{\ell}$, and a p-lca-reduced formula $\widehat{\varphi}(\overline{x})$ such that, for every $(G, \ell) \in \mathcal{G}[\Lambda]$, and every p-skeleton S of G, there exists a $\widehat{\ell}$-labeling $\widehat{\ell}$ of G such that*

$$\textit{true}(\varphi, G, \ell) = \textit{true}(\widehat{\varphi}, S, \widehat{\ell}).$$

Moreover, there exists a distributed algorithm that computes the labeling $\widehat{\ell}$ in $\mathcal{O}(D + \log n)$ rounds, assuming ℓ and S are given as input.

Proof. We prove the lemma by designing the algorithm by induction on the quantifier depth t of the formula. For the base case, $\varphi(\overline{x})$ is quantifier-free. Let $p = |\overline{x}|$. Thanks to Lemma 4.15, the p-skeleton S can be computed in $\mathcal{O}(\log n)$ rounds. We can then apply Lemma 5.4 with $|\overline{y}| = 0$, and return the reduction of (Λ, φ, ℓ) on S.

Now let us suppose that the statement of the lemma holds for all formulas with quantifier depth $t \geq 0$. Let $\varphi(\overline{x}) = Qy, \psi(\overline{x}, y)$, where $Q \in \{\exists, \forall\}$, and ψ is a formula of quantifier depth t. We study $Q = \exists$ and $Q = \forall$ separately.

Existential quantifier. By induction hypothesis, an integer p_1, a set Λ_1, a p_1-lca-reduced formula $\psi_1(\overline{x}, y)$, a set of labels ℓ_2, and a p_1-skeleton S such that

$$\mathsf{true}(\psi, G, \ell) = \mathsf{true}(\psi_1, S, \ell_1)$$

can be computed in $\mathcal{O}(D + \log n)$ rounds. One would then like to apply Lemma 5.4 on $\exists y\, \psi_1(\overline{x}, y)$. However, ψ_1 is not defined over graphs of bounded expansion, but over p_1-skeletons. Therefore, we first apply Lemma 5.5 on $p_1, \Lambda_1, \psi_1, G, S, \ell_1$, to compute $\Lambda_2, \psi_2(\overline{x}, y)$, and ℓ_2 satisfying

$$\mathsf{true}(\psi_1, S, \ell_1) = \mathsf{true}(\psi_2, G, \ell_2)$$

in $\mathcal{O}(\log n)$ rounds, where ψ_2 is existential, of the form $\psi_2(\overline{x}, y) = \exists \overline{z}, \rho(\overline{x}, y, \overline{z})$, where $\rho \in \mathrm{FO}[\Lambda_2]$ is quantifier-free. Now we can define $\varphi_2(\overline{x}) = \exists y\, \psi_2(\overline{x}, y)$. Note that φ_2 is also existential, and

$$\mathsf{true}(\varphi_2, G, \ell_2) = \mathsf{true}(\varphi, G, \ell).$$

We can finally apply Lemma 5.4 to compute a reduction $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ of $(\Lambda_2, \varphi_2, \ell_2)$.

Universal quantifier. In this case, we merely rewrite φ as:

$$\varphi(\overline{x}) = \neg \exists y\, \neg \psi(\overline{x}, y).$$

By induction hypothesis, an integer p_1, a set Λ_1, a p_1-lca-reduced formula $\psi_1(\overline{x}, y)$, a Λ_1-labeling ℓ_1 of G, and a p_1-skeleton S_1 of G can be computed in $\mathcal{O}(D + \log n)$ rounds, such that

$$\mathsf{true}(\psi, G, \ell) = \mathsf{true}(\psi_1, S_1, \ell_1).$$

The negation of ψ_1 is still a p_1-lca-reduced formula. Let us apply Lemma 5.5 on p_1, Λ_1, $\neg\psi_1$, G, S_1, and ℓ_1 to compute a set Λ_2, an existential formula ψ_2, and a Λ_2-labeling ℓ_2 of G in $\mathcal{O}(\log n)$ rounds, such that

$$\mathsf{true}(\neg\psi_1, S_1, \ell_1) = \mathsf{true}(\psi_2, G, \ell_2).$$

Now we can define $\rho(\overline{x}) = \exists y\, \psi_2(\overline{x}, y)$. Observe that this formula is existential, and it satisfies

$$\mathsf{true}(\varphi, G, \ell) = \mathsf{true}(\neg\rho, G, \ell_2).$$

We can eventually apply Lemma 5.4 to compute the reduction $(p, \widehat{\Lambda}, \widehat{\rho}, \widehat{\ell})$ of $(\Lambda_2, \rho, \ell_2)$ in $\mathcal{O}(D + \log n)$ additional rounds. In particular, $\widehat{\rho}$ is p-lca-reduced, as well as $\neg\widehat{\rho}$. Therefore $(p, \widehat{\Lambda}, \neg\widehat{\rho}, \widehat{\ell})$ is the reduction of (Λ, φ, ℓ).

We obtain that eliminating each quantifier of φ can be done in $\mathcal{O}(D + \log n)$ rounds. $\square$

5.4 Proof of Theorem 5.1

Let φ be a formula in FO[Λ]. We define the formula $\varphi'(x)$ with one (dummy) free variable by

$$\varphi'(x) = \varphi \wedge (x = x).$$

We do have that, for every labeled graph (G, ℓ), and for every node $v \in V(G)$,

$$(G, \ell) \models \varphi'(v) \iff (G, \ell) \models \varphi.$$

By Lemma 5.6, $\varphi'(x)$ can be reduced to a p-lca-reduced formula $\widehat{\varphi}(x)$ in $\mathcal{O}(D+\log n)$ rounds. Since $\widehat{\varphi}(x)$ is quantifier free, each node v can check locally (without additional communication) whether it satisfies $\widehat{\varphi}$. By definition of the reduction, the truth of $\widehat{\varphi}$ is equivalent to that of φ', which completes the proof. $\square$

6 Counting and Optimization Problems

In this section, we prove Theorems 1.6 and 1.7. We restate these two theorems here, in their full generality of labeled graphs.

Theorem 6.1. *Let Λ be a finite set. For every FO formula $\varphi(x_1, \ldots, x_k)$ on Λ-labeled graphs with $k \geq 1$ free variables $x_1, \ldots, x_k$, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a distributed algorithm that, for every n-node network $G = (V, E) \in \mathcal{G}$ of diameter D and every $\ell : V \to 2^\Lambda$, performs global counting in $O(D + \log n)$ rounds in $\mathsf{CONGEST}$. If the formula $\varphi(x_1, \ldots, x_k)$ is local then there exists a distributed algorithm that performs local counting in $\mathcal{O}(\log n)$ rounds in $\mathsf{CONGEST}$.*

Theorem 6.2. *Let Λ be a finite set. For every FO formula $\varphi(x_1, \ldots, x_k)$ on Λ-labeled graphs with $k \geq 1$ free variables $x_1, \ldots, x_k$, and for every class graphs $\mathcal{G}$ of bounded expansion, there exists a distributed algorithm that, for every n-node network $G = (V, E) \in \mathcal{G}$ of diameter D, every $\ell : V \to 2^\Lambda$, and every weight function $\omega : V \to \mathbb{N}$, computes a k-tuple of vertices $(v_1, \ldots, v_k)$ such that $(G, \ell) \models \varphi(v_1, \ldots, v_k)$ and $\sum_{i=1}^{k} \omega(v_i)$ is maximum, in $\mathcal{O}(D + \log n)$ rounds under the $\mathsf{CONGEST}$ model. (The same holds if replacing maximum by minimum.) If no tuples $(v_1, \ldots, v_k)$ exist such that $(G, \ell) \models \varphi(v_1, \ldots, v_k)$, then all nodes reject.*

To ease the notations, let us adopt the following notations.

- Let $\mathsf{OPT}_\varphi(G, L)$ denote the optimum value for φ, i.e., the maximum value of $\sum_{i=1}^{k} \omega(v_i)$ over all k-uples of vertices $(v_1, \ldots, v_k)$ such that $G \models \varphi(v_1, \ldots, v_k)$.

- Let $\mathsf{count}_\varphi(G, L)$ denote the number of k-uples satisfying the formula.
- Given some $(p, f(p))$-treedepth coloring of G, and a set of colors U, let $\mathsf{OPT}_\varphi(G[U], L)$ and $\mathsf{count}_\varphi(G[U], L)$ be the same quantities as above, but assuming that the formula is restricted to the subgraph $G[U]$ of G induced by the colors in U.

Let us prove the following lemma, which is restricted to quantifier-free formulas.

Lemma 6.3. *Let Λ be a finite set. For every* quantifier-free *FO formula $\varphi(x_1, \dots, x_k)$ on Λ-labeled graphs with $k \geq 1$ free variables $x_1, \dots, x_k$, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a distributed algorithm that, for every n-node network $G = (V, E) \in \mathcal{G}$ of diameter D, and every $\ell : V \to 2^\Lambda$, performs global counting in $O(D + \log n)$ rounds under the CONGEST model. If the formula is local, then there exists an algorithm that performs local counting in $\mathcal{O}(\log n)$ rounds under the CONGEST model.*

6.1 Proof of Lemma 6.3 and Theorem 6.2

We first consider Lemma 6.3, on counting. In this case, the formula φ has exactly k variables, namely $x_1, \dots, x_k$. We now apply Proposition 3.9 to compute, in $\mathcal{O}(\log n)$ rounds, a $(k, f(k))$-treedepth coloring of G. Since formula φ has exactly k variables, a k-uple of vertices $(v_1 \dots, v_k)$ satisfies the formula in (G, ℓ) if and only if it satisfies it in some set $G[U]$. By definition, $\mathsf{count}_\varphi(G[U], \ell)$ is the size of the set $\mathsf{true}(\varphi, G[U], \ell)$ of k-uples $(v_1, \dots, v_k)$ such that $(G[U], \ell) \models \varphi(v_1, \dots, v_k)$. An easy but crucial observation is that, for any two sets of colors U_1, U_2, we have that

$$\mathsf{true}(\varphi, G[U_1], \ell) \cap \mathsf{true}(\varphi, G[U_2], \ell) = \mathsf{true}(\varphi, G[U_1 \cap U_2], \ell),$$

because $\varphi(x_1, \dots, x_k)$ has no other variables than the free ones. The condition fails if one allows other variables, which explains that the technique does not extend directly to arbitrary FO formulas; this is the subject of Section 6.2. The equality above allows us to use the inclusion-exclusion principle in order to count the solutions of φ, assuming that we were able to count the partial solutions over subgraphs $G[U]$, where U are sets of at most k colors.

Lemma 6.4. *Let $\varphi(x_1, \dots, x_k) \in FO(\Lambda)$ be a quantifier-free formula. Let us consider a $(k, f(k))$-treedepth coloring of $(G, \ell) \in \mathcal{G}[\Lambda]$. Let $\kappa = \binom{f(k)}{k}$, and let $U_1 \dots U_\kappa$ be all subsets of $[f(k)]$ of size k. Then*

$$\mathsf{count}_\varphi(G, \ell) = \sum_{h=1}^{\kappa} (-1)^{h-1} \sum_{1 \leq i_1 \leq \dots \leq i_h \leq \kappa} \mathsf{count}_\varphi(G[U_{i_1} \cap \dots \cap U_{i_h}], \ell).$$

Proof. Recall that for any $i \in [\kappa]$, $\mathsf{count}_\varphi(G[U_i], \ell)$ is the size of the set $\mathsf{true}(\varphi, G[U_i], \ell)$ of k-uples $(v_1, \dots, v_k)$ such that $(G[U_i], \ell) \models \varphi(v_1, \dots, v_k)$. Also, $\mathsf{count}_\varphi(G[U_{i_1} \cap \dots \cap U_{i_h}], \ell)$ is the size of the set $\mathsf{true}(\varphi, G[U_{i_1} \cap \dots \cap U_{i_h}], \ell)$, which is itself equal to $\mathsf{true}(\varphi, G[U_{i_1}]) \cap \dots \cap \mathsf{true}(\varphi, G[U_{i_h}], \ell)$. Eventually, $\mathsf{true}(\varphi, G, \ell) = \mathsf{true}(\varphi, G[U_1], \ell) \cup \dots \cup \mathsf{true}(\varphi, G[U_\kappa], \ell)$ and $\mathsf{count}_\varphi(G, \ell) = |\mathsf{true}(\varphi, G, \ell)|$, so the formula is a straightforward application of the inclusion-exclusion principle. □

Before explaining (in Lemma 6.7) how to compute the values $\mathsf{count}_\varphi(G[U], \ell)$, we address optimization as stated in Theorem 6.2. Recall that, in general, the formula φ may have other (quantified) variables than its free valiables. We first transform $\varphi(\overline{x})$ into an equivalent existential formula $\tilde{\varphi}(\overline{x})$ at the cost of enriching the set of labels, and the labeling of the input graph.

Lemma 6.5. *Given a formula $\varphi(\overline{x}) \in FO[\Lambda]$, and a labeled graph $(G, \ell) \in \mathcal{G}[\Lambda]$, a tuple $(\tilde{\Lambda}, \tilde{\varphi}, \tilde{\ell})$ such that*

1. *$\tilde{\ell}$ is a $\tilde{\Lambda}$-labeling of G,*
2. *$\tilde{\varphi}(\overline{x})$ is an existential formula in $FO[\tilde{\Lambda}]$, i.e., $\tilde{\varphi}(\overline{x}) = \exists \overline{y}\, \psi(\overline{x}, \overline{y})$ where $\psi \in \tilde{\Lambda}$ is quantifier-free, and*
3. *$\mathsf{true}(\varphi, G, \ell) = \mathsf{true}(\tilde{\varphi}, G, \tilde{\ell})$,*

can be computed in $\mathcal{O}(D + \log n)$ rounds in $\mathsf{CONGEST}$.

Proof. Thanks to Lemma 5.6 (see also Definition 4.16), there exists some constant p such that, for any p-skeleton S of G (obtained by Lemma 4.15), the tuple $(\Lambda, \varphi, \tau_p, \ell)$ can be reduced to a tuple $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ in $\mathcal{O}(D + \log n)$ rounds, where $\widehat{\varphi}(\overline{x})$ is a p-lca-reduced (thus quantifier-free) formula in $FO(\widehat{\Lambda})$, $\widehat{\ell}$ is a $\widehat{\Lambda}$ labeling of S, and $\mathsf{true}(\varphi, G, \ell) = \mathsf{true}(\widehat{\varphi}, S, \widehat{\ell})$. Now, given the tuple $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$, Lemma 5.5 states that the desired tuple $(\tilde{\Lambda}, \tilde{\varphi}, \tilde{\ell})$ can be computed with no extra communication. $\square$

Thanks to Lemma 6.5, we may now assume, w.l.o.g., that, in the proof of Theorem 6.2, the formula $\varphi(\overline{x}) = \exists \overline{y} \psi(\overline{x}, \overline{y})$ is given in an existential form, where $\psi \in FO[\Lambda]$ is quantifier-free.

Let $p = |\overline{x}| + |\overline{y}|$. We use again Proposition 3.9 to compute a $(p, f(p))$-treedepth coloring of G in $\mathcal{O}(\log n)$ rounds. Since φ has exactly p variables, a k-uple of vertices $(v_1 \ldots, v_k)$ satisfies the formula in (G, ℓ) if and only if it satisfies it in some subgraph $G[U]$, where U can be viewed as obtained by "guessing" the colors of $(v_1, \ldots, v_k)$ as well as the colors of the quantified variables $(y_1, \ldots, y_{p-k}) = \overline{y}$. This is formally stated as follows, for further references.

Lemma 6.6. *Let $\varphi \in FO(\Lambda)$ be an existential formula, i.e., $\varphi(\overline{x}) = \exists \overline{y}, \psi(\overline{x}, \overline{y})$, where $\psi \in FO[\Lambda]$ is quantifier-free. Let $p = |\overline{x}| + |\overline{y}|$, and let us assume a given a $(p, f(p))$ coloring of $(G, \ell) \in \mathcal{G}[\Lambda]$. We have*

$$\mathsf{OPT}_\varphi(G, \ell) = \max_{U \subseteq [f(p)],\, |U| \leq p} \mathsf{OPT}_\varphi(G[U], \ell).$$

It remains to compute $\mathsf{OPT}_\varphi(G[U], \ell)$ and $\mathsf{count}_\varphi(G[U], \ell)$ for every set U of at most p colors, given some $(p, f(p))$-treedepth coloring of G. This is partially done in Proposition 3.7 thanks a previous result in [31] in which it is shown that $\mathsf{OPT}_\varphi(G, \ell)$ and $\mathsf{count}_\varphi(G, \ell)$ can be computed in a constant number of rounds in connected graphs of bounded treedepth. We however need to extend their techniques to fit our purposes, i.e., to deal with *disconnected* graphs of bounded treedepth (the non-necessarily connected subgraphs of G induced by a set of colors U). For global counting, the cost of communication among different components adds an additive term $\mathcal{O}(D)$ to the round complexity.

Lemma 6.7. *Let $\mathcal{G}[\Lambda]$ be a class of labeled graphs of bounded expansion, let $\varphi(\overline{x})$ be a $FO(\Lambda)$ formula, and $p \in \mathbb{N}$. Let us assume given a $(p, f(p))$-treedepth coloring of $(G, \ell) \in \mathcal{G}[\Lambda]$, and let $U \subseteq [f(p)]$ be any set of at most p colors. Then $\mathsf{OPT}_\varphi(G[U], \ell)$ and $\mathsf{count}_\varphi(G[U], \ell)$ can be computed in $\mathcal{O}(D)$ rounds in $\mathsf{CONGEST}$. Moreover, if φ is local, then local counting in $G[U]$ can be solved in $\mathcal{O}(1)$ rounds.*

Proof. Let us first consider the case of local counting, that is, we assume φ is local. In this case, $\mathsf{count}_\varphi(G[U], \ell)$ is equal to the sum of count_φ over all connected components of $G[U]$, i.e.,

$$\mathsf{count}_\varphi(G[U], \ell) = \sum_{H \in \mathrm{CC}(G[U])} \mathsf{count}_\varphi(H, \ell),$$

where $\mathrm{CC}(G[U])$ denotes the set of connected components of $G[U]$. Each component $H \in CC(G[U])$ has treedepth at most $|U|$, and thus, by Proposition 3.7, $\mathsf{count}_\varphi(H, \ell)$ can be computed in $\mathcal{O}(1)$ rounds. One arbitrary node $v_H \in H$ outputs $\nu(v_H) = \mathsf{count}_\varphi(H, \ell)$, whereas every other node u outputs $\nu(u) = 0$. Since H has bounded treedepth and thus bounded diameter, we can choose v_H to be for example the node in H with maximal identifier. It follows that

$$\sum_{v \in V} \nu(v) = \sum_{H \in \mathrm{CC}(G[U])} \nu(v_H) = \sum_{H \in \mathrm{CC}(G[U])} \mathsf{count}_\varphi(H, \ell) = \mathsf{count}_\varphi(G[U], \ell)$$

Let us now focus on counting and optimization for an arbitrary (non-necessarily local) formula.

To prove the lemma, we generalize the techniques of [31] for proving Proposition 3.7. We will only restate here the main features of the proof in [31], based on tree decomposition and treewidth-related techniques. Recall that the tree decomposition of a graph $H = (V, E)$ is a pair

$$(T = (I, F), \{B_i\}_{i \in I})$$

where T is a tree, and, for every $i \in I$, B_i is a subset of vertices of V, referred to as the *bag* of node i in T, or merely "bag i", such that

- every vertex of H is contained in at least one bag,
- for every edge of H, at least one bag contains its two endpoints, and
- for every vertex v of H, the nodes of T whose bags contain v form a connected subgraph of T.

The width of the decomposition is the maximum size of a bag, minus one. The treewidth of H is the minimum width over all its tree decompositions. Note that graphs of treedepth at most p have treewidth at most 2^p [67]. The proof of Proposition 3.7 is based on three ingredients.

1. The first ingredient is a dynamic programming centralized algorithm for decision, optimization and counting solutions of a given formula $\varphi(x) \in FO(\Lambda)$, over labeled graphs (H, ℓ) of bounded treewidth. Here we will assume that a tree decomposition of constant width is part of the input, and moreover the tree is rooted. The distributed algorithm used in [31] is based on the centralized algorithm by Borie, Parker and Tovey [6]. The nodes of the tree decomposition $(T = (I, F), \{B_i\}_{i \in I})$ are processed bottom-up, from the leaves to the root of T. When processing node i, let $j_1, \dots, j_q$ be its children in T. The essential observation due to [6] is that the dynamic programming tables at node i are of constant size for decision, and of $\mathcal{O}(\log n)$ size for counting and optimization. Moreover, for computing the information (encoding partial solutions) at node i, one only needs to know the subgraphs $H[B_i]$ and $H[B_{j_1}], \dots, H[B_{j_q}]$ induced by the bags at node i and its children, together with the information already computed at the children nodes. In particular, when i is a leaf, $H[B_i]$ suffices for computing the dynamic programming table at node i.

2. The second ingredient of [31] is an implementation of the above centralized algorithm in the $\mathsf{CONGEST}$ model, in a number of rounds equal to the depth of the tree decomposition T, under the condition that one can associate to each node i of T a vertex $h(i)$ of H, such that vertices $h(i)$ satisfy several properties in order to "simulate the behavior" of node i in the dynamic programming. Vertex $h(i)$ must know the depth of tree T and the depth of node i in

T, the bag B_i and the subgraph $H[B_i]$, and moreover $h(i)$ knows the node $h(p_T(i))$ associated to the parent $p_T(i)$ of i in T, and vertices $h(i), h(p_T(i))$ are equal or adjacent in H. Informally, the distributed algorithm implements the centralized one by making vertex $h(i)$ to help it computing the right moment (i.e., at the round corresponding to the depth of T minus the depth of i in T, plus one) the dynamic programming table at node i is processed, and the result transmitted to $h(p_T(i))$. Observe that $h(i)$ has collected all the information from the children $j_1, \ldots, j_q$ of i, thanks to the fact that $h(j_1), \ldots, h(j_q)$ have already computed and communicated it at the previous round. (To be complete, in [31], one actually has $h(i) = i$ and T is a subgraph of H, but our generalization is a simple observation.) This explains that the number of rounds of the distributed algorithm is the depth of the tree decomposition T.

3. The third ingredient in [31] is an algorithm computing the desired tree decomposition $(T = (I, F), \{B_i\}_{i \in I})$ of graph H of treedepth at most p in a constant number of rounds, together with the information requested by the previous item. The width of the tree-decomposition is at most p, and the depth of T is at most 2^p.

Let us now come back to our framework, that is when $H = G[U]$ is the subgraph of G induced by some set U of at most p colors of some $(p, f(p))$-treedepth decomposition of G. In particular $H = G[U]$ has treedepth at most p. Our goal is to replace the third item above by a distributed algorithm running in $\mathcal{O}(D)$ rounds in CONGEST for computing a suitable tree decomposition of $G[U]$. We compute a p-skeleton using Lemma 4.15. Note that this can be done in $\mathcal{O}(2^{2p})$ rounds, hence in a constant number of rounds whenever the coloring is provided. Let F^U be the resulting decomposition forest spanning $G[U]$, of depth at most 2^d. Recall that each node knows its parent and depth in the forest.

Generalizing [31], we transform the decomposition forest F^U into a tree decomposition of $G[U]$. This tree decomposition $(T = (I, F), \{B_i\}_{i \in I})$ has to be computed in time $\mathcal{O}(D)$ and to satisfy all the conditions of the second ingredient above. The only difference is that, for each node i of T, its "leader" $h(i)$ has to be a vertex of G (not necessarily of $G[U]$), and, for each pair of nodes i, j of T such that $j = p_T(i)$, vertices $h(i)$ and $h(j)$ must be adjacent in G. In our construction, there may be up to two different nodes of T having the same leader in G.

Let us choose an arbitrary vertex r of G, and compute a breadth-first search tree of G. From this rooted tree, remove every node v that does not have a descendant which is a root in F^U. Note that the construction of this tree can be performed in time $\mathcal{O}(D)$ rounds. We call TS the resulting tree, rooted at r. TS can be viewed as a Steiner tree in G spanning r and all the vertices labeled root^U. The depth of TS is, by construction, at most D, the diameter of G.

From now on, let us focus on TS alone, independently from the the fact that it is a subgraph of G. Let us then construct a "virtual" tree T by augmenting TS a follows. For every vertex labeled root^U in TS, we root the corresponding tree T^U to that vertex. So, such a vertex has two types of children in T, those originally in TS, and those in the tree T^U rooted at that vertex. Note that two different vertices i and j in T may correspond to the same node v in G, as node i may be v as a node in TS, and j may be v again, but as a node of T^U. For every node i of T, let $h(i)$ be the corresponding vertex of G. Note that, by a simple distributed algorithm working in $\mathcal{O}(D)$ rounds, each node $h(i)$ of G can learn the depth of T, the depth of i in T, and the identifier of $h(p_T(i))$. The depth of T is at most $D + 2^p$ because the forest F has depth at most 2^p. By construction, $h(i)$ and $h(p_T(i))$ are either adjacent in G or equal, whenever i is labeled root^U.

It remains to construct the bag B_i for each node i of T, and to ensure that the node $h(i)$ of G knows the subgraph induced by B_i in $H = G[U]$.

- For every node i of T belonging to TS, we set $B_i = \emptyset$.
- For every node i belonging to F^U, we act as in [31], i.e., we set $h(i) = i$, and B_i is the set of ancestors of i in F^U. As observed in [31], each node i can learn the subgraph induced by bag B_i in a constant number of rounds. Moreover, for each tree T^U of the forest F^U, the restriction of the decomposition $(T = (I, F), \{B_i\}_{i \in F})$ to T^U is a tree decomposition of the subgraph of G induced by the vertices of T^U, of width at most 2^p.

It is a matter of exercise to check that, by attaching to TS the tree-decompositions corresponding to the trees T^U in the forest F^U, the resulting tree is a tree decomposition of $G[U]$ with width at most 2^p. Therefore we can apply the same dynamic programming scheme as above for computing values $\mathsf{OPT}_\varphi(G[U], \ell)$ and $\mathsf{count}_\varphi(G[U], \ell)$ in $D + 2^p$ rounds. Together with the construction of the tree decomposition, the overall algorithm takes $\mathcal{O}(D)$ rounds. $\square$

By plugging the result of Lemma 6.7 into Lemma 6.6 and Lemma 6.4 over all sets U of at most p colors for a suitable p depending on φ and a $(p, f(p))$-treedepth coloring of G, we achieve the proof of Lemma 6.3 and Theorem 6.2. Note that in each case, we need to build a $(p, f(p))$-treedepth coloring, which can be done in $\mathcal{O}(\log n)$ $\mathsf{CONGEST}$ rounds by Proposition 3.9. $\square$

6.2 Proof of Theorem 6.1

Lemma 6.3 shows how to solve the counting problem for *quantifier-free* FO formulas on bounded expansion classes. In order to prove the more general Theorem 6.1, we apply the quantifier elimination techniques developed in Sections 4 and 5. Lemma 5.6 transforms a general FO formula into a p-lca-reduced formula. However, Lemma 6.3 does not handle such formulas, as it only deals with usual adjacency and equality predicates. In order to solve that issue, it is tempting to use Lemma 5.5, which replaces lca predicates by FO formulas. However, this would result in an existential formula rather than a quantifier free formula as needed for applying Lemma 6.3. The main idea of the proof of Theorem 6.1 is to maintain that the quantified variables introduced by Lemma 5.5 are quantified *uniquely*, in the following sense.

Definition 6.8. *Let $p \in \mathbb{N}$, let Λ a set of labels, and let $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$ be a p-lca-reduced formula. φ is* lca-freeable *on $\mathcal{G}$ if there exists $\widehat{\Lambda}$, and a quantifier free formula $\psi(\overline{x}, \overline{y}) \in \mathrm{FO}[\widehat{\Lambda}]$ such that, for every $G = (V, E) \in \mathcal{G}$, for every Λ-labeled p-skeleton (S, ℓ) of G, there exists a $\widehat{\Lambda}$-labeling $\widehat{\ell}$ of G satisfying*

1. *$\mathsf{true}(\varphi, S, \ell) = \mathsf{true}\Big(\big(\exists \overline{y}\, \psi(\overline{x}, \overline{y})\big), G, \widehat{\ell}\Big)$, and*
2. *for every $\overline{v} \subseteq V$ satisfying $(S, \ell) \models \varphi(\overline{v})$, there exists a unique $\overline{w} \subseteq V$ such that $(G, \widehat{\ell}) \models \psi(\overline{v}, \overline{w})$.*

The following lemma outlines the usefulness of lca-freeability for counting.

Lemma 6.9. *Let $p \in \mathbb{N}$, and let $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$ be a p-lca-reduced, and lca-freeable formula. For every $\widehat{\Lambda}, \psi(\overline{x}, \overline{y}), G, S, \ell, \widehat{\ell}$ as in Definition 6.8,*

$$\mathsf{count}_\varphi(S, \ell) = \mathsf{count}_\psi(G, \widehat{\ell}).$$

Proof. Definition 6.8 specifies the existence of a one-to-one correspondence between $\mathsf{true}(\varphi, S, \ell)$ and $\mathsf{true}(\psi, G, \widehat{\ell})$. These two set have thus the same cardinality. □

The following lemma is an analog of Lemma 5.5.

Lemma 6.10. *Every formula $\varphi(y,z) = \mathsf{lca}_i^U(y,z)$ is lca-freeable on $\mathcal{G}$. Moreover, assuming (S,ℓ) is given as input, every node u can compute $\widehat{\Lambda}, \psi, \widehat{\ell}(u)$ without any communication.*

Proof. The proof follows the same guidelines as the one of Lemma 5.5. Recall labels color_k were added for each color k, and the label depth_j^U was added to mark the depth of a node in each forest F^U. Each lca predicates was then replaced by an existential formula. The following formula is very similar, except that we removed the quantifiers on $y_0, \ldots, y_{d-1}, z_0, \ldots, z_{d-1}$, which are now free variables, and we added another constraint, specified on the last line of the formula below.

$$\begin{aligned}
\psi(y,z,y_0,\ldots,y_{d-1},z_0,\ldots,z_{d-1}) = \bigvee_{(a,b)\in[\max(0,i),d-1]^2} \Bigg(& (y_a = y) \wedge (z_b = z) \\
& \wedge \Big(\bigwedge_{s\in[0,i]} y_s = z_s\Big) \wedge \Big(\bigwedge_{s\in[i+1,\min(a,b)]} y_s \neq z_s\Big) \wedge \mathsf{depth}_0^U(y_0) \wedge \mathsf{depth}_0^U(z_0) \\
& \wedge \Big(\bigwedge_{s\in[1,a]} \mathsf{depth}_s^U(y_s) \wedge \mathsf{adj}(y_{s-1},y_s)\Big) \wedge \Big(\bigwedge_{s\in[1,b]} \mathsf{depth}_s^U(z_s) \wedge \mathsf{adj}(z_{s-1},z_s)\Big) \\
& \wedge \Big(\bigwedge_{s\in[a+1,d-1]} y_s = y\Big) \wedge \Big(\bigwedge_{s\in[b+1,d-1]} z_s = z\Big)\Bigg).
\end{aligned}$$

Recall that a (resp., b) represents here the depth of y (resp., z) in F^U, and $y_0, \ldots, y_a$ (resp., $z_0, \ldots, z_b$) represent the path from the root of y's tree to y (resp., from the root of z's tree to z). Since, in general, variables $y_{a+1}, \ldots, y_{d-1}, z_{b+1}, \ldots, z_{d-1}$ can take any value, the (new) last line of the formula above is needed to enforce their uniqueness. For the same reasons as in Lemma 5.5, ψ satisfies

$$\mathsf{true}(\varphi, S, \ell) = \mathsf{true}(\exists \overline{y}, \exists \overline{z}, \psi(y,z,\overline{y},\overline{z}), G, \widehat{\ell}).$$

Let us now prove that the quantified variables are unique. Fix $(y,z) \in \mathsf{true}(\varphi, S, \ell)$ and let $\overline{y}, \overline{z}$ such that $(G, \widehat{\ell}) \models \psi(y,z,\overline{y},\overline{z})$. For some $(a,b) \in [i, d-1]^2$, the inside of the disjunction of ψ is true. This implies that:

- y (resp. z) has depth a (resp. b),
- for $s \leq a$ (resp. $s \leq b$), y_s (resp. z_s) is the (unique) ancestor of y (resp. z) at depth s, and
- for $s > a$ (resp. $s > b$), $y_s = y$ (resp. $z_s = z$), thus it is also unique.

This completes the proof. □

Lemma 6.11. *A conjunction of lca-freeable formulas is lca-freeable.*

Proof. Let $\varphi_1(\overline{x}), \varphi_2(\overline{x})$ be two lca-freeable formulas and $\varphi(\overline{x}) = \varphi_1(\overline{x}) \wedge \varphi_2(\overline{x})$. Let $\widehat{\Lambda}_1, \widehat{\Lambda}_2$, $\psi_1(\overline{x}, \overline{y}), \psi_2(\overline{x}, \overline{z})$ as in Definition 6.8. We set $\widehat{\Lambda} = \widehat{\Lambda}_1 \cup \widehat{\Lambda}_2$, where we assume, w.l.o.g., that the sets of labels $\widehat{\Lambda}_1, \widehat{\Lambda}_2$ are disjoint, and we set $\psi(\overline{x}, \overline{y}, \overline{z}) = \psi_1(\overline{x}, \overline{y}) \wedge \psi_2(\overline{x}, \overline{z})$. For $G, S, \ell_1, \ell_2, \widehat{\ell}_1, \widehat{\ell}_2$ as in Definition 6.8, we define $\widehat{\ell}(u) = \widehat{\ell}_1(u) \cup \widehat{\ell}_2(u)$. Let $\overline{v} \in \mathsf{true}(\varphi, S, \ell)$. Since ψ_1 (resp. ψ_2) is lca-freeable, there exists a unique $\overline{w}$ (resp. $\overline{u}$) such that $(\overline{v}, \overline{w}) \in \mathsf{true}(\psi_1, G, \widehat{\ell}_1)$ (resp. $(\overline{v}, \overline{u}) \in \mathsf{true}(\psi_2, G, \widehat{\ell}_2)$). Then, $(G, \widehat{\ell}) \models \psi(\overline{v}, \overline{w}, \overline{u})$. Conversely, let $\overline{w'}, \overline{u'}$ such that $(G, \widehat{\ell}) \models \psi(\overline{v}, \overline{w'}, \overline{u'})$. Then, $(G, \widehat{\ell}) \models \psi_1(\overline{v}, \overline{w'}$ (resp. $\psi_2(\overline{v}, \overline{u'})$), which implies $\overline{w'} = \overline{w}$ (resp. $\overline{u'} = \overline{u}$). $\square$

At this stage, we can now prove the global counting part of Theorem 6.1.

Proof of Theorem 6.1 (global counting). Let Λ be a (finite) set of labels and $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$. By Lemma 5.6, there exists $p \in \mathbb{N}$, Λ_1 and $\varphi_1(\overline{x}) \in \mathrm{FO}[\Lambda_1]$ p-lca-reduced such that, for every $(G, \ell) \in \mathcal{G}[\Lambda]$, and for every S p-skeleton of G, there exists a Λ_1-labeling ℓ_1 of S such that $\mathsf{true}(\varphi, G, \ell) = \mathsf{true}(\varphi_1, S, \ell_1)$ (and thus $\mathsf{count}_\varphi(G, \ell) = \mathsf{count}_{\varphi_1}(S, \ell_1)$. We compute a p-skeleton using Lemma 4.15, and the corresponding labeling ℓ_1, in $\mathcal{O}(D + \log n)$ rounds. Let us write φ_1 in disjunctive normal form:

$$\varphi_1(\overline{x}) = \bigvee_{j=1}^{k} \psi_j(\overline{x})$$

where ψ_j is a conjunction of lca_i^U predicates, label predicates, color predicates, or their negations. As in Lemma 4.19, note that $\neg\mathsf{lca}_i^U(x, y)$ is equivalent to:

$$\neg\mathsf{color}_U(x) \vee \neg\mathsf{color}_U(y) \vee \bigvee_{j \in [-1,d] \setminus \{i\}} \mathsf{lca}_j^U(x, y).$$

We may thus assume that φ_1 does not contain any negated lca predicate. We apply the inclusion/exclusion principle to φ_1, that is,

$$\mathsf{count}_{\varphi_1}(S, \ell_1) = \sum_{h=1}^{k} (-1)^{h-1} \sum_{1 \le j_1 < \cdots < j_h \le k} \mathsf{count}_{\bigwedge_{m=1}^{h} \psi_{j_m}}(S, \ell_1).$$

It only remains to compute the values of

$$\mathsf{count}_{\bigwedge_{m=1}^{h} \psi_{j_m}}(S, \ell_1),$$

for some fixed indexes $j_1, \ldots, j_h$. Since each ψ_{j_m} are conjunctions of lca^U-predicates and (possibly negated) label and color predicates, it is also true of $\bigwedge_{m=1}^{h} \psi_{j_m}$. By Lemmas 6.10 and 6.11, this formula is lca-freeable. By Lemma 6.9 there exists $\widehat{\Lambda}$, $\widehat{\psi} \in \mathrm{FO}[\Lambda]$ quantifier-free, and $\widehat{\ell}$ such that

$$\mathsf{count}_{\bigwedge_{m=1}^{h} \psi_{j_m}}(S, \ell_1) = \mathsf{count}_{\widehat{\psi}}(G, \widehat{\ell}).$$

Moreover, $\widehat{\Lambda}, \widehat{\psi}, \widehat{\ell}(u)$ can be computed by any node u without communication (cf. Lemmas 6.10 and 6.11, and their proofs). Finally, since $\widehat{\psi}$ is quantifier free, it only remains to apply Lemma 6.3 to complete the proof. $\square$

In order to prove our result on *local* counting, we need to adapt the notion of *lca-freeable formulas* so that it preserves locality.

Definition 6.12. *Let $p \in \mathbb{N}$, let Λ be a (finite) set of labels, and let $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$ be a* local *p-lca-reduced formula. φ is* locally lca-freeable on $\mathcal{G}$ *if the following two conditions hold:*

- *φ is lca-freeable, and*
- *the quantifier-free formula ψ from Definition 6.8 can be chosen to be local.*

The following lemma is the "locality preserving version" of Lemmas 6.10 and 6.11.

Lemma 6.13. *Let $\varphi(\overline{x}) \in \mathrm{FO}[\Lambda]$ be a p-lca-reduced formula that is supposed to be a conjunction of lca^U predicates, and (possibly negated) label and color predicates. If φ is local then it is locally lca-freeable. Moreover, assuming a p-skeleton (S, ℓ) is given as input, each node u can compute $\widehat{\Lambda}, \psi, \widehat{\ell}(u)$ matching the requirement of Definition 6.8 without any communication.*

Before proving Lemma 6.13, we first prove a technical lemma.

Lemma 6.14. *Let $\varphi(\overline{x})$ be a local p-lca-reduced formula that is assumed to be a conjunction of lca^U predicates, and (possibly negated) label and color predicates. Let $k = |x|$, and let $H(\varphi)$ be the graph with vertex set equal to the set of variables $x_1, \ldots, x_k$, and where there is an edge between x_i and x_j if and only if the formula φ contains a predicate of the form $\mathsf{lca}^U_m(x_i, x_j)$ with $m \geq 0$. If there exists a labeled skeleton (S, ℓ), and a set of vertices $\overline{v}$ such that $(S, \ell) \models \varphi(\overline{v})$, then, $H(\varphi)$ is connected.*

Proof. Let us fix a formula φ, a labeled skeleton (S, ℓ), and a set of vertices $\overline{v}$ as in the statement of the lemma. We proceed by contradiction. Let us assume that $H(\varphi)$ is not connected. Up to a reordering of the variables $x_2, \ldots, x_k$, we may assume that $x_1, \ldots, x_s$ form a single connected component, $1 \leq s < k$, and that the vertices $x_{s+1}, \ldots, x_k$ are in other connected components. Let us consider the labeled skeleton (S', ℓ') consisting of two connected components S^1 and S^2, each of which being a copy of (S, ℓ). For any vertex $u \in S$, let us denote by u^i the copy of u in S^i. Let $\overline{w}$ where, for every $i = 1, \ldots, k$,

$$w_i = \begin{cases} v_i^1 & \text{if } i \leq s \\ v_i^2 & \text{if } i > s \end{cases}$$

We have that $(S', \ell') \models \varphi(\overline{w})$. However $w_{s+1}, \ldots, w_k$ are not in the connected component of w_1. In particular, they are not at bounded distance from w_1. This contradicts the fact that φ is local (see Definition 3.2). $\square$

When $H(\varphi)$ is connected, we say that φ is *lca-connected*.

Proof of Lemma 6.13. Since φ is local, we get from Lemma 6.14 that it is either lca-connected, or always false. Let us assume that it is not lca-connected. Then one can trivially output the formula $\perp$ that is always false. Thus, we now assume that φ is lca-connected. We then merely adapt the proofs of Lemmas 6.10 and 6.11. Let us consider a labeled p-skeleton (S, ℓ). As in the proof of Lemma 6.10, we add new labels depth^U_i, and color_k, and we denote by $\widehat{\ell}$ the resulting labeling. We then replace each predicate $\mathsf{lca}_i^U(y,z)$ of φ by

$$\begin{aligned}\bigvee_{(a,b)\in[\max(0,i),d-1]^2} \Bigg(&(y_a = y) \wedge (z_b = z) \wedge \Big(\bigwedge_{s\in[0,i]} y_s = z_s \Big) \wedge \Big(\bigwedge_{s\in[i+1,\min(a,b)]} y_s \neq z_s \Big) \\ &\wedge \mathsf{depth}_0^U(y_0) \wedge \mathsf{depth}_0^U(z_0) \\ &\wedge \Big(\bigwedge_{s\in[1,a]} \mathsf{depth}_s^U(y_s) \wedge \mathsf{adj}(y_{s-1},y_s) \Big) \wedge \Big(\bigwedge_{s\in[1,b]} \mathsf{depth}_s^U(z_s) \wedge \mathsf{adj}(z_{s-1},z_s) \Big) \\ &\wedge \Big(\bigwedge_{s\in[a+1,d-1]} y_s = y \Big) \wedge \Big(\bigwedge_{s\in[b+1,d-1]} z_s = z \Big) \Bigg),\end{aligned}$$

where $y_1,\ldots,y_d,z_1,\ldots,z_d$ are new free variables (which are *unique*, see the proof of Lemma 6.10). Note that if the formula above is satisfied, then each newly introduced variable y_i (resp. z_i) is at distance at most d from y (resp. z). Moreover, if $i \geq 0$, then y and z are at distance at most $2d$, as they are both at distance at most d from $y_0 = z_0$. Let us denote by $\psi(\overline{x},\bar{t})$ the formula obtained by replacing each lca^U-predicate by the formula above ($\bar{t}$ is the collection of newly introduced variables). By the same argument as in Lemma 6.10, we have

$$\mathsf{true}(\varphi,S,\ell) = \mathsf{true}(\exists \bar{t}\ \psi(\overline{x},\bar{t}),G,\widehat{\ell}),$$

and

$$\mathsf{count}_\varphi(S,\ell) = \mathsf{count}_\psi(G,\widehat{\ell}).$$

It only remains to show that ψ is local. Since it is quantifier-free, this is equivalent to showing that, for some r', every labeled graph (G',ℓ') satisfies

$$(\overline{v},\overline{w}) \in \mathsf{true}(\psi,G',\ell') \implies \overline{v},\overline{w} \subseteq B_{G'}(v_1,r').$$

Since, are underlined above, each newly introduced variable in $t_i \in \bar{t}$ is at distance at most d from at least one of the variables x_j, it is sufficient to show that for some r'

$$(\overline{v},\overline{w}) \in \mathsf{true}(\psi,G',\ell') \implies \overline{v} \subseteq B_{G'}(v_1,r'),$$

as $\overline{v} \subseteq B_{G'}(v_1,r')$ implies that $\overline{w} \subseteq B_{G'}(v_1,r'+d)$. Recall that $\varphi(\overline{x})$ is lca-connected, and so let $H(\varphi)$ be defined as in Lemma 6.14. For each index $1 \leq j \leq |x|$, since $H(\varphi)$ is connected there is a path $x_{i_1},\ldots,x_{i_k}$ from x_1 to x_j in $H(\varphi)$. Since $H(\varphi)$ has $|x|$ nodes, we may assume this path has length at most $|x|-1$. The formula ψ guarantees that, for every $l<k$, x_{i_l} and $x_{i_{l+1}}$ are at distance at most $2d$ in G'. Finally, x_1 and x_j are at distance at most $2d(|x|-1)$. This concludes the proof that φ is local. □

We are now ready to establish Theorem 6.1.

Proof of Theorem 6.1 (local counting). We follow the same guidelines as in the proof of the global counting result, but we make sure that locality is preserved throughout the process. Let us consider a labeled graph (G,ℓ). Let us apply the *locality preserving* elimination technique (cf. Lemma 4.20) for reducing φ to a local p-lca-reduced formula φ_1. We then construct a p-skeleton S

(cf. Lemma 4.15), and a new labeling ℓ_1, in $\mathcal{O}(\log n)$ rounds. Let us rewrite φ_1 in disjunctive normal form:

$$\varphi_1(\overline{x}) = \bigvee_{j=1}^{k} \psi_j(\overline{x})$$

where ψ_j is a conjunction of lca^U-predicates, and (possibly negated) label and color predicates — recall that we can assume that there are no negated lca^U-predicates. The formula φ_1 is quantifier-free so, thanks to Lemma 3.6, each term of the disjunction is local. Let us apply the inclusion/exclusion principle to φ_1, i.e.,

$$\mathsf{count}_{\varphi_1}(S, \ell_1) = \sum_{h=1}^{k} (-1)^{h-1} \sum_{1 \le j_1 < \cdots < j_h \le k} \mathsf{count}_{\bigwedge_{m=1}^{h} \psi_{j_m}}(S, \ell_1).$$

It thus remains solely to perform the local counting of

$$\mathsf{count}_{\bigwedge_{m=1}^{h} \psi_{j_m}}(S, \ell_1)$$

for fixed indexes $j_1, \dots, j_h$. Since each formula ψ_{j_m} is a *local* conjunction of lca-predicates (as well as possibly negated label and color predicates), then same holds for $\bigwedge_{m=1}^{h} \psi_{j_m}$. By Lemma 6.13, this latter formula formula is locally lca-freeable, and one can compute a local, quantifier free formula $\widehat{\psi}$, and a labeling $\widehat{\ell}$ of G, without any communication, such that

$$\mathsf{count}_{\bigwedge_{m=1}^{h} \psi_{j_m}}(S, \ell_1) = \mathsf{count}_{\widehat{\psi}}(G, \widehat{\ell}).$$

Since $\widehat{\psi}$ is local and quantifier-free, it suffices to apply (the local case of) Lemma 6.3 for completing the proof. □

7 Distributed Certification

In this section we prove that every graph property expressible in FO can be certified with certificates on $\mathcal{O}(\log n)$ bits in graphs of bounded expansion.

Theorem 7.1. *Let Λ be a finite set. For every FO formula φ on Λ-labeled graphs, and for every class of graphs $\mathcal{G}$ of bounded expansion, there exists a distributed certification scheme that, for every n-node network $G = (V, E) \in \mathcal{G}$ and every $\ell : V \to 2^{\Lambda}$, certifies $(G, \ell) \models \varphi$ using certificates on $\mathcal{O}(\log n)$ bits.*

For establishing Theorem 7.1, we revisit the logical structures introduced and analyzed in the previous section, by mainly focusing on the points where theses structures must be certified instead of being computed.

7.1 Certifying Reduction of Trees with Bounded Depth

We begin with a basic primitive, that we use to count values, to identify connected components, and for several other applications.

Proposition 7.2 ([56]). *Let us suppose that each node u in a graph $G = (V, E)$ is given as an input a triple $(\textsf{root}(u), \textsf{parent}(u), \textsf{depth}(u))$, where $\textsf{root}(u)$ and $\textsf{parent}(u)$ are node identifiers, and $\textsf{depth}(u)$ is a positive integer. Let us consider the algorithm where each node u checks:*

- *For each $v \in N(u)$, $\textsf{root}(v) = \textsf{root}(u) = \rho$.*
- *If $\textsf{id}(u) = \rho$ then $\textsf{parent}(u) = \textsf{id}(u)$ and $\textsf{depth}(u) = 0$.*
- *If $\textsf{id}(u) \neq \rho$, then there exists $v \in N(u)$ s.t. $\textsf{id}(v) = \textsf{parent}(u)$ and $\textsf{depth}(v) = \textsf{depth}(u) - 1$.*

If all checks are passed at every node, then there exists a spanning tree T of G rooted at ρ such that the function $\textsf{parent}$ defines the parent relation in T, and $\textsf{depth}(u)$ is the depth of u in T.

In the above proposition, if all checks are passed at every node, then the set

$$\{(\textsf{root}(u), \textsf{parent}(u), \textsf{depth}(u)) \mid u \in V\}$$

is called the *encoding* of the tree T. Given a graph G and a forest $F \subseteq G$ with components $T_1, \ldots, T_k$, we say that the nodes of G are given the *encoding* of F if, for every $i \in [k]$, each node $u \in T_i$ is given the encoding of T_i.

Lemma 7.3. *Let $\mathcal{A}$ be an algorithm such that, in every n-node network $G = (V, E)$, every node $u \in V$ is given an $\mathcal{O}(\log n)$-bit input $\textsf{In}(u)$, and must produce an $\mathcal{O}(\log n)$-bit output $\textsf{Out}(u)$. Let $R(n)$ be the round-complexity of $\mathcal{A}$ in the BROADCAST CONGEST model. Then, there is a certification scheme using certificates on $\mathcal{O}(R(n) \log n)$ bits that certifies that a pair of functions $f : V \to \{0,1\}^*$ and $g : V \to \{0,1\}^*$ form an input-output pair for $\mathcal{A}$ (i.e., if, for every node $u \in V$, $\textsf{In}(u) = f(u)$, then, for every node $u \in V$, $\textsf{Out}(u) = g(u)$).*

Proof. On legal instances (f, g) for network $G = (V, E)$, the prover provides each vertex $u \in V$ with the sequence $M(u) = (m_1(u), \ldots, m_{R(n)}(u))$ where $m_i(u)$ is the message that u broadcasts to its neighbors on input $f(u)$ during the i-th round of $\mathcal{A}$.

The verification algorithm proceeds as follows. Every node collects the certificates of its neighbors, and simulates the $R(n)$ rounds of $\mathcal{A}$. That is, u checks first whether $m_1(u)$ is indeed the message that it would send in the first round of $\mathcal{A}$. For every $i \in \{2, \ldots, R(n)\}$, node u assumes that the messages $m_1(v), \ldots, m_{i-1}(v)$ of each of its neighbors $v \in N(u)$ were correct, and checks whether $m_i(u)$ is indeed the message that it would broadcast at round i. If some of these tests are not passed then u rejects. If all tests are passed, then u finally checks whether $g(u)$ is the output that u would produce by executing $\mathcal{A}$, and accepts or rejects accordingly.

The completeness and soundness of the algorithm follow by construction. □

Let us now prove a result analogous to Lemma 5.2, but for distributed certification.

Lemma 7.4. *Let (G, ℓ) be a connected Λ-labeled graph, and let $F \in \mathcal{F}_d$ be a subgraph of G. Let $\varphi(\overline{x}) = \exists \overline{y}\ \zeta(\overline{x}, \overline{y})$ be a formula in $\mathrm{FO}[\Lambda]$, where ζ is an $\textsf{lca}$-reduced formula. Let us assume that every node u of G is provided with the following values:*

- *Two sets Λ and $\widehat{\Lambda}$;*
- *The formula φ, and a $\textsf{lca}$-reduced formula $\widehat{\varphi}$;*
- *A set of labels $\ell(u) \subseteq \Lambda$, and a set of labels $\widehat{\ell}(u) \subseteq \hat{\Lambda}$;*

- *The encoding* $(\textsf{\textit{root}}(u), \textsf{\textit{parent}}(u), \textsf{\textit{depth}}(u))$ *of* F*;*

- *The encoding* $(\textsf{\textit{root}}^T(u), \textsf{\textit{parent}}^T(u), \textsf{\textit{depth}}^T(u))$ *of a spanning tree* T *of* G*.*

There is a certification scheme using $\mathcal{O}(\log n)$*-bit certificates that certifies that* $\textsf{\textit{true}}(\varphi, F, \ell) = \textsf{\textit{true}}(\widehat{\varphi}, F, \tilde{\ell})$*.*

Proof. Let $k = |\overline{x}|$. As in the proof of Lemma 4.8, let us assume w.l.o.g. that $k \geq 1$ (as otherwise we can create a dummy free variable x for φ). Moreover, let us assume that $\widehat{\Lambda}$ and $\widehat{\varphi}$ are the set and the formula defined in the proof of Lemma 4.8. Since $\widehat{\Lambda}$ and $\widehat{\varphi}$ can be computed by every node without any communication, each node can reject in case they are not appropriately defined. Similarly, we can assume that ℓ and $\hat{\ell}$ are, respectively, a Λ-labeling and a $\hat{\Lambda}$-labeling. Thanks to Proposition 7.2, $\mathcal{O}(\log n)$-bits certificates are sufficient to certify that the encodings of F and T are correctly. The remaining of the proof consists in checking that $\widehat{\ell}$ is correctly set as specified the proof of Lemma 4.8.

Let us first suppose that $|\overline{y}| = 1$. We shall deal with the case when $|y| > 1$ later in the proof. We start by proving the result for the case where ζ is a $\textsf{lca}$-type $\textsf{type}_{\gamma,\delta} \in \textsf{Type}(k+1, d, \Lambda)$, where the variable y is identified with variable $k+1$ in the definition of γ and δ. In a second stage, we'll define s, h, h_s and h_y in the same way as we did in Lemma 4.8, and revisit the cases of Lemma 4.8.

In Case 1 and 2, observe that the algorithms provided in Lemma 4.10 are algorithms that fit with the broadcast constraint of the BROADCAST CONGEST model, running in $\mathcal{O}(d)$ rounds. On input ℓ, these algorithms output $\widehat{\ell}$. By Lemma 7.3, we can certify whether $\widehat{\ell}$ is well defined in $d \cdot \mathcal{O}(\log n) = \mathcal{O}(\log n)$ rounds.

Now let us deal with Case 3. The algorithm $\mathcal{A}$ given in Lemma 5.2 for this case has three phases. The first two phases can be implemented by an algorithm $\mathcal{B}$ under the BROADCAST CONGEST model, running in $\mathcal{O}(d)$ rounds. The output of $\mathcal{B}$ consists in a marking of all active roots, as well as a marking of all good nodes. We can simulate these two first phases by giving a certificate to each node, interpreted as the markings given by $\mathcal{B}$, as well as a certificate of the execution of $\mathcal{B}$. According to Lemma 7.3, this can be implemented using certificates on $\mathcal{O}(\log n)$ bits. The last phase of Algorithm $\mathcal{A}$ aims at providing each node with the number of active roots. Each node u receives a certificate consisting of:

- an integer $\rho(u)$ equal the number of active roots in F;

- a bit $a(u)$ indicating whether or not u is an active root;

- an integer $\textsf{val}(u)$ equal to the sum of all bits $a(v)$ in the subtree of T rooted at u.

Observe that these certificates can be encoded in $\mathcal{O}(\log n)$ bits. For a non-negative integer ν, let

$$h(\nu) = \begin{cases} \nu & \textsf{if } \nu \leq k, \\ \infty & \textsf{otherwise.} \end{cases}$$

The verifier checks the following conditions at every node u:

(1) For every $v \in N(u)$, $\rho(u) = \rho(v)$;

(2) $a(u) = 1$ if and only if u is marked as active by $\mathcal{B}$;

(3) The following condition holds

$$\mathsf{val}(u) = a(u) + \sum_{v \in \mathsf{Children}(u)} \mathsf{val}(v),$$

where $\mathsf{Children}(u)$ denotes the set of nodes v such that $\mathsf{parent}^T(v) = \mathsf{id}(u)$;

(4) If $\mathsf{id}(u) = \mathsf{root}^T(u)$, then $\mathsf{val}(u) = \rho(u)$;

(5) If $u \in F$ is marked as good by $\mathcal{B}$, then $\widehat{\ell}(u) = \ell(u) \cup \{\mathsf{good}, h(\rho(u))\}$;

(6) If $u \in F$ is not marked as good by $\mathcal{B}$, then $\widehat{\ell}(u) = \ell(u) \cup \{h(\rho(u))\}$;

(7) If $u \notin F$ then $\widehat{\ell}(u) = \ell(u)$.

We now check completeness and soundness of this certification scheme. Condition (1) is satisfied if and only if every node received the same value ρ for the number of active roots. Conditions (2) and (3) are both satisfied if and only if $\mathsf{val}(u)$ correspond to the number of active roots in the subgraph of T rooted at u. In particular $\mathsf{val}(r)$ is equal to the number of active roots. Then, conditions (1) to (4) are satisfied if and only if ρ corresponds to the number of active roots in F. Finally, conditions (1) to (7) are satisfied if and only if $\widehat{\ell}$ is well defined according to the proof of Lemma 4.8.

Let us denote by $\mathcal{A}^{\mathsf{type}}$ the verification algorithm, and by Cert^{ζ} the certificates described for each of the three cases when ζ is an lca-type formula. Let us now assume that ζ is an arbitrary lca-reduced formula. From Lemma 4.5, there exists a set $I \subseteq \mathsf{Type}(k+1, D, \Lambda)$ such that,

$$\zeta(\overline{x}, y) = \bigvee_{\psi \in I} \psi(\overline{x}, y).$$

For each $\psi \in I$, let us denote by $\widehat{\Lambda}^{\psi}$, $\widehat{\psi}$ and $\widehat{\ell^{\psi}}$ the reduction on $\mathcal{F}_d$ of Λ, ψ and ℓ, respectively. The prover provides each node u of G with the certificate $c^{\psi}(u) = \widehat{\ell^{\psi}}(u)$ for each ψ, in addition to the certificate Cert^{ψ}.

The verifier proceeds as follows. Each node u checks that it accepts when running algorithm $\mathcal{A}^{\mathsf{type}}$ with certificates $c^{\psi}(u)$, for every $\psi \in I$. Then, each node u relabels the sets $\widehat{\ell^{\psi}}$ to make them disjoint, as we did in the proof of Lemma 4.8), and checks whether $\widehat{\ell}(u) = \bigcup_{\psi \in I} \widehat{\ell^{\psi}}(u)$. The soundness and completeness of the whole scheme follows from the soundness and completeness of $\mathcal{A}^{\mathsf{type}}$.

It remains to deal with the case where $q = |\overline{y}| > 1$. Let us denote by $\mathcal{A}$ the verification algorithm that we described for the case when $|\overline{y}| = 1$. The prover provides the nodes with a set of triplets $(\zeta_i, \ell_i, \mathsf{Cert}_i)_{i \in [q]}$ such that, for each $i \in [q]$, Cert_i is a certificate used to check whether

$$\mathsf{true}(\exists y_i, \zeta_i(\overline{x}, y_1, \ldots, y_i), G, \ell_i) = \mathsf{true}(\zeta_{i-1}(\overline{x}, y_1, \ldots, y_{i-1}), G, \ell_{i-1}),$$

Where $\zeta_q = \zeta$, $\ell_q = \ell$, $\zeta_0 = \widehat{\varphi}$, and $\ell_0 = \widehat{\ell}$. Every node u accepts if $\mathcal{A}$ accept all certificates at u.

From the soundness and completeness of $\mathcal{A}$, we obtain that every node accepts if and only if

$$\begin{aligned}\mathsf{true}(\varphi(\overline{x}), G, \ell) &= \mathsf{true}(\exists(y_1,\ldots,y_q)\,\zeta(\overline{x},(y_1,\ldots,y_q)), G, \ell)\\ &= \mathsf{true}(\exists(y_1,\ldots,y_q)\,\zeta_q(\overline{x},(y_1,\ldots,y_q)), G, \ell_q)\\ &= \mathsf{true}(\exists(y_1,\ldots,y_{q-1})\,\zeta_{q-1}(\overline{x},(y_1,\ldots,y_{q-1})), G, \ell_{q-1})\\ &\ldots\\ &= \mathsf{true}(\exists y_1\,\zeta_1(\overline{x},y_1), G, \ell_1)\\ &= \mathsf{true}(\zeta_0(\overline{x}), G, \ell_0)\\ &= \mathsf{true}(\widehat{\varphi}(\overline{x}), G, \widehat{\ell}).\end{aligned}$$

Which completes the proof. □

7.2 Certifying Low-Treedepth Colorings and Skeletons

Bousquet et al. [27] have shown that, for every MSO property $\mathcal{P}$, there is a certification scheme using $\mathcal{O}(\log n)$-bit certificates for certifying $\mathcal{P}$ on graphs of bounded treedepth. Since the mere property "treedepth at most k" is expressible in MSO, we obtain the following.

Proposition 7.5 ([27])**.** *For every integer $p \geq 1$, there exists a certification scheme using $\mathcal{O}(\log n)$-bit certificates for certifying "treedepth at most p".*

We show how to use this result for certifying a $(p, f(p))$-treedepth coloring. Moreover, we also extend this certification scheme to an encoding of a p-skeleton. Given a $(p, f(p))$-treedepth coloring C_p, and p-skeleton S, the *encoding* of S is defined as the encoding of all decomposition forests F^U of $G[U]$, for all $U \in \mathsf{Col}(p)$. More precisely, in the encoding of S, for every $U \in \mathsf{Col}(p)$, every node u has a triple $\{(\mathsf{root}^U(u), \mathsf{parent}^U(u), \mathsf{depth}^U(u))\}$ corresponding to the subtree T^U_u containing u of the decomposition forest F^U of $G[U]$.

Lemma 7.6. *Let $f : \mathbb{N} \to \mathbb{N}$, and let $\mathcal{G}$ be a class of graphs of expansion f. Let $p > 0$, and let us suppose that each node u is given (1) a value $C_p(u) \in [f(p)]$, and (2) an encoding of S. There is a certification scheme using $\mathcal{O}(\log n)$-bit certificates for certifying that C_p is a $(p, f(p))$-treedepth coloring of G, and that the encoding of S is indeed defining a p-skeleton of G.*

Proof. Let us first certify that C_p is a $(p, f(p))$-treedepth coloring of G. For every set $U \in \mathsf{Col}(p)$, every node u receives as a certificate: (1) The encoding of a rooted spanning tree T of the component of $G[U]$ containing u (where each component is identified with the node identifier of the root of T); (2) The certification of the property "treedepth at most p" for all the nodes in the component of $G[U]$ containing u. The soundness and completeness of (1) and (2) follow from Proposition 7.2 and Proposition 7.5.

Next, let us certify that S is a p-skeleton of G. First, we use the verifier of Proposition 7.2 to check that the encoding of each tree T^U_u is correct, for all $u \in V$ and all $U \in \mathsf{Col}(p)$. If no nodes reject at that point, we can safely deduce that if u and v are in the same connected component of $G[U]$, then the encoding of T^U_u and of T^U_v must induce the same tree.

It remains to certify that each T^U_u forms a decomposition tree of the component of $G[U]$ containing u. More precisely, we must certify that all neighbors of u in $G[U]$ are either descendants or ancestors of u in T^U_u. Let us fix an arbitrary set $U \in \mathsf{Col}(p)$. Let $(\mathsf{root}^U(u), \mathsf{parent}^U(u), \mathsf{depth}^U(u))$

be the encoding of T_u^U that has u in its input (given in the encoding of S). Then node u receives the set $P_u^U = \{w_1^u, \ldots, w_d^u\}$ as certificate, corresponding to a sequence of at most $d \leq 2^p$ node identifiers representing the path from $\mathsf{root}(u)$ to u. Then node u accepts P_u^U if the following holds:

(i) $d = \mathsf{depth}^U(u)$;

(ii) $w_1^u = \mathsf{root}^U(u)$ and $w_d^u = \mathsf{id}(u)$;

(iii) if $\mathsf{id}(u) \neq \mathsf{root}^U(u)$ and $v \in N(u)$ is such that $\mathsf{id}(v) = \mathsf{parent}^U(u)$, then P_v^U is a sequence of length $d-1$ satisfying $w_v^i = w_u^i$ for every $i \in [d-1]$;

(iv) if every $v \in N(u)$ is such that $C_p(v) \in U$, then either $\mathsf{id}(u) \in P_v^U$ or $\mathsf{id}(v) \in P_u^U$.

Conditions (i) to (iii) are satisfied if and only if P_u^U represents the path from the root of T_u^U to u, for all u. Assuming that (i) to (iii) are satisfied, we have that (iv) is satisfied if and only if T_u^U is a decomposition tree of $G[U]$, for all nodes u of $G[U]$. Indeed, for every u in $G[U]$, and for every $v \in N(u)$ that is also in $G[U]$, we have that either u is an ancestor of v (i.e. $\mathsf{id}(u)$ belongs to P_v^U), or v is an ancestor of u (i.e. $\mathsf{id}(v)$ belongs to P_u^U).

Then, for every $U \in \mathsf{Col}(p)$, one can certify that T_u^U is an decomposition tree of $G[U]$ with a certificate containing at most 2^p node identifiers, which can be encoded on $\mathcal{O}(2^p \cdot \log n) = \mathcal{O}(\log n)$ bits. Furthermore, one can certify that the union over all nodes u and all $U \in \mathsf{Col}(p)$, of T_u^U is a p-skeleton of G using certificates of size $\mathcal{O}(|\mathsf{Col}(p)| \cdot 2^p \cdot \log n) = \mathcal{O}(\log n)$ bits. □

7.3 Certifying Reductions on Graphs of Bounded Expansion

Let $f : \mathbb{N} \to \mathbb{N}$, and let $\mathcal{G}$ be a class of connected graphs with expansion f. We first define a notion of reducibility that admits an efficient distributed certification scheme.

Definition 7.7. *Let $\varphi \in \mathrm{FO}[\Lambda]$. For every labeled graph (G, ℓ) with $G \in \mathcal{G}$, let us assume that every node u of G is provided with the following values:*

1. *The set Λ, the formula φ, and the label $\ell(u) \subseteq \Lambda$;*
2. *A positive integer p;*
3. *A set $\widehat{\Lambda}$, a formula $\widehat{\varphi}$, and a label $\widehat{\ell}(u) \subseteq \widehat{\Lambda}$;*
4. *The encoding of a p-skeleton S of G.*

We say that φ admits a distributedly certifiable reduction on $\mathcal{G}$ *if there is a certification scheme using $\mathcal{O}(\log n)$-bits certificates for certifying that $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ is the reduction of $(\Lambda, \varphi, S, \ell)$ on G.*

In a way similar to what was done in Section 5, we show that every first-order formula over labeled graphs admits a certifiable reduction. We fist establish this result for existential formulas.

Lemma 7.8. *Every existential formula in* $\mathrm{FO}[\Lambda]$ *admits a distributed certifiable reduction on $\mathcal{G}$.*

Proof. Let $\varphi \in \mathrm{FO}[\Lambda]$ be an existential formula. Consider that each node is provided with the values listed in Definition 7.7. Let $\varphi(\overline{x}) = \exists \overline{y}\, \psi(\overline{x}, \overline{y})$, where ψ is quantifier free. Every node checks that $p = |\overline{x}| + |\overline{y}|$. The scheme of Lemma 7.6 enables to certify that the encoding S correctly defines a p-skeleton of G. Let C_p be the $(p, f(p))$-treedepth coloring that defines S. The same construction as in Lemma 5.4 can then be used. More precisely, for every $U \in \mathsf{Col}(p)$, let $\widetilde{\Lambda}^U$, $\widetilde{\psi}^U$, $\widetilde{\ell}^U$, Λ^U, ψ^U, and ℓ^U be defined as in the proof of Lemma 5.4. Instead of computing these values, each node receives them from the prover as certificates. The values of $\widehat{\Lambda}$, $\widehat{\varphi}$, $\widetilde{\Lambda}^U$, $\widetilde{\psi}^U$, Λ^U, and ψ^U do not depend on the input graph, and can be computed by each node. The labeling $\widetilde{\ell}^U$ can be computed from ℓ by a 1-round algorithm in the $\mathsf{BROADCAST\ CONGEST}$ model. Hence, using Lemma 7.3, the correctness of each labeling $\widetilde{\ell}^U$ can be certified with $\mathcal{O}(\log n)$-bit certificates. The correctness of each labeling $\widehat{\ell}^U$ can be certified using Lemma 7.4, using $\mathcal{O}(\log n)$-bit certificates. Finally, each node can check the correctness of $\widehat{\ell}$, and accept or reject accordingly. □

We can now proceed with the general case.

Lemma 7.9. *Every formula in* FO[Λ] *admits a distributed certifiable reduction on $\mathcal{G}$.*

Proof. Let $\varphi \in \mathrm{FO}[\Lambda]$ be an existential formula. Let us assume that each node is provided with the values described in Definition 7.7, i.e. the tuples (Λ, φ, ℓ) and $(p, \hat{\Lambda}, \hat{\varphi}, \hat{\ell})$, and the encoding of a p-skeleton S. We follow the same induction as in Lemma 5.4, over the quantifier depth of φ. The base case is trivial as the construction requires no communication. Furthermore, if $\varphi = \neg\psi$ where ψ admits a distributed certifiable reduction, then φ admits a distributed certifiable reduction as well, as we described in Lemma 5.4.

It remains to consieder the case $\varphi(\overline{x}) = \exists y\, \psi(\overline{x}, y)$. Let $(p^\psi, \widehat{\Lambda}^\psi, \widehat{\psi}, \widehat{\ell}^\psi(u))$, $(\Lambda^\xi, \xi, \ell^\xi)$ as it was done in Case 1 in the proof of Lemma 5.6. Again, instead of computing these values, each node receives them as certificates, and checks them. The values of $p, p^\psi, \widehat{\Lambda}^\psi$, $\widehat{\psi}$, Λ^ξ, and ξ do not depend on the input graph, and can be computed by each node. Then, each node can construct the input values (1) to (4) given in Definition 7.7 with respect to to ξ and ψ. By the induction hypothesis, the correctness of $\widehat{\ell}^\psi(u)$ at each node u can be certified with $\mathcal{O}(\log n)$-bit certificates.

Recall that, in Lemma 5.6, the value of $\ell^\xi(u)$ on each node u was computed using Lemma 5.5, without any communication, under the assumption that each node knows a encoding a p^ψ-skeleton S of G. The prover provides the encoding of a p^ψ-skeleton to each node as a $\mathcal{O}(\log n)$-bit certificate, which can be checked using Lemma 7.6. Finally, the correctness of $\widehat{\ell}$ can be certified with $\mathcal{O}(\log n)$-bit certificates, using the scheme of Lemma 7.8 with given values $(\Lambda^\xi, (\exists y, \xi(\overline{x}, y)), \ell^\xi)$ and $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ and S. □

7.4 Proof of Theorem 7.1

We are now ready to prove Theorem 7.1.

Proof. Let $\varphi \in \mathrm{FO}[\Lambda]$. We proceed the same way as in Theorem 5.1. Specifically, let us redefine $\varphi(x)$, as φ with a dummy free-variable x. The prover provides each node with the following certificate:

(i) a positive integer p, a set $\widehat{\Lambda}$, a p-lca-reduced formula $\widehat{\varphi}$, and a label set $\widehat{\ell}(u)$;

(ii) the certificate of u used to certify the p-skeleton S of G;

(iii) the certificate $\mathsf{Cert}(u)$ for certifying that $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ is the reduction of $(\Lambda, \varphi, S, \ell)$.

The part (i) of the certificate can be encoded in $\mathcal{O}(1)$ bits, the part (ii) can be encoded in $\mathcal{O}(\log n)$ bits, and Lemma 7.9 asserts that a certificate for (iii) exists and, furthermore, it can be encoded in $\mathcal{O}(\log n)$ bits.

The verifier proceeds as follows. Each node checks that $(p, \widehat{\Lambda}, \widehat{\varphi}, \widehat{\ell})$ is the reduction of $(\Lambda, \varphi, S, \ell)$ using the certificates given in (iii) using the verifier $\mathcal{A}$ in Lemma 7.9, with certificates (i) and (ii) provided to the nodes. If a node rejects in $\mathcal{A}$, then the node rejects, otherwise it carries on the verification as described hereafter.

Assuming that no nodes have rejected so far, we have that $\mathsf{true}(\varphi, G, \ell) = \mathsf{true}(\widehat{\varphi}, G, \widehat{\ell})$. Since φ has only one dummy variable, the truth value of $\varphi(x)$, as well as the one of $\widehat{\varphi}(x)$, is the same for every node in G. Moreover, the formula $\widehat{\varphi}(x)$ is a Boolean combination of label predicates of x. Therefore, every node u can check $\widehat{\varphi}(u)$ using $\widehat{\ell}(u)$, and it accepts if $\widehat{\varphi}(u)$ holds, and rejects otherwise.

It remains to analyze soundness and completeness.

For establishing completeness of the certification scheme, let us suppose that $G \models \varphi$. Then, by Lemma 5.6, we have that φ is reducible in $\mathcal{G}$. Then there exist certificates for (i) and (ii), and, by Lemma 7.9, there exist certificates for (iii) such that every node accepts when running verifier $\mathcal{A}$. Since $G \models \varphi$ we have that $\mathsf{true}(\varphi, G) = \mathsf{true}(\widehat{\varphi}, G, \widehat{\ell}) = V(G)$. It follows that $\widehat{\varphi}(u)$ is true at every node u, and thus every node accepts.

For establishing soundness, let us suppose that $G \not\models \varphi$, and let us suppose that the nodes received certificates (i)-(iii) such that every node accepts when renning the verifier $\mathcal{A}$. Then $\mathsf{true}(\varphi, G, \ell) = \mathsf{true}(\widehat{\varphi}, G, \widehat{\ell})$. However, in this case $\mathsf{true}(\varphi, G, \ell) = \emptyset$. Therefore $\widehat{\varphi}(u)$ does not hold at any node u, and thus all nodes reject. □

8 Lower Bounds and Impossibility Results

In this section we give evidence that our results somehow represent the limit of tractability of distributed model checking and distributed certification of properties on graphs of bounded expansion.

All our reductions are based on classical results in communication complexity. Let $f : X \times Y \to \{0, 1\}$ be a function. The *deterministic communication complexity of* f $D(f)$ corresponds to the minimum number of bits that must be exchanged in any deterministic protocol between a player Alice holding an input $x \in X$ and another player Bob $y \in Y$ in order to output the value $f(x, y)$. The *non-deterministic communication complexity of* f, $N(f)$ is defined similarly, but for non-deterministic communication protocols. We refer to the book of Kushilevitz and Nissan for more detailed definitions [59].

In important problem in communication complexity is *set disjointedness*. It is defined by function Disj_n. For each n the inputs of Disj_n are subsets of $[n]$. Given $A, B \subseteq [n]$ we have that $\mathsf{Disj}(A, B) = 1$ if and only if $A \cap B = \emptyset$.

Proposition 8.1 ([59])**.** *$D(\mathsf{Disj}_n) = \Omega(n)$ and $N(\mathsf{Disj}_n) = \Omega(n)$.*

8.1 No Extension to Monadic Second-Order Logic

We first show that it is not possible to extend our results to the whole set of MSO properties on graphs of bounded expansion. We illustrate this fact by considering the property of being non-three-colorable. We define Non-3-Coloring the set of graphs that do not admit a proper three-coloring (their chromatic number is at least 4). The membership in Non-3-Coloring can be represented by the following MSO formula:

$$\forall C_1, C_2, C_3, \exists x, y,\ \mathsf{adj}(x, y) \wedge \big((x \in C_1 \wedge y \in C_1) \vee (x \in C_2 \wedge y \in C_2) \vee (x \in C_3 \wedge y \in C_3)\big)$$

In the following theorem, we show that there are no efficient CONGEST algorithms for checking nor compact certification scheme for certifying the membership in Non-3-Coloring, even under the promise that the input graph belongs to a class of graphs of bounded expansion and of logarithmic diameter.

Theorem 8.2. *Let $\mathcal{G}$ the class of graphs of maximum degree 5 and diameter $\mathcal{O}(\log n)$. Every CONGEST algorithm deciding the membership in Non-3-Coloring of a n-node input graph in $\mathcal{G}$ requires $\Omega(n/\log^2 n)$ rounds of communication.*

Every certification scheme that certifies membership to Non-3-Coloring of a n-node input graph in $\mathcal{G}$ requires certificates of size $\Omega(n/\log n)$.

Proof. [44] show that every locally checkable proof, and therefore every proof-labeling scheme for Non-3-Coloring requires certificates of size $\Omega(n^2/\log n)$. The same proof directly implies the same lowerbound for the number of rounds of a CONGEST algorithm deciding the membership in Non-3-Coloring. We give an sketch of the proof of Göös and Suomella and then we show how to adapt it to the case when the input graph has maximum degree 4.

The proof of [44] is consists in a reduction to Disj_{n^2}. In the reduction each instance (A, B) of Disj_{n^2} is associated with a $\mathcal{O}(n)$-node graph $G_{A,B}$. The set of vertices of $G_{A,B}$ is divided in two sets of vertices, denoted V_A and V_B, and satisfies the following conditions:

(C1) The graph induced by V_A only depends on A.

(C2) The graph induced by V_B only depends on B.

(C3) The set of edges $E_{A,B}$ with one endpoint in V_A and the other in V_B contains at most $\mathcal{O}(\log n)$ edges, and does not depend on A and B, only on n. The nodes with endpoints in $E_{A,B}$ are denoted $V_{A,B}$.

(C4) $A \cap B = \emptyset$ if and only if $G_{A,B} \in$ Non-3-Coloring.

Then, given a certification scheme $\mathcal{P}$ for Non-3-Coloring with certificates of size $c(n)$ on n-node graphs, we build the following non-deterministic protocol for Disj_{n^2}. First, Alice and Bob build graph $G_{A,B}$, where Alice builds the subgraph depending on A, i.e., V_A and all edges incident to V_A, and Bob the part depending on B. Then, non-deterministically each player generate the certificates of all the nodes of $G_{A,B}$, and communicate to the other player the certificates generated for the nodes in the cut. Then, the players run the verification algorithm of $\mathcal{P}$ and accept if all the nodes on their side accept. We have that both players accept if and only if $G_{A,B} \in$ Non-3-Coloring, which holds if and only if $A \cap B = \emptyset$. The total communication is $\mathcal{O}(\log n \cdot c(n))$. Using Proposition 8.1 we obtain that $\mathcal{O}(\log n \cdot c(n)) = N(\mathsf{Disj}_{n^2}) = \Omega(n^2)$. Therefore, $c(n) = \Omega(n^2/\log n)$.

The construction is also useful for a lowerbound in the CONGEST model. Let $\mathcal{A}$ be an algorithm deciding the membership in Non-3-Coloring in $\mathcal{R}(n)$ rounds. Then Alice and Bob construct $G_{A,B}$ as above, and simulate the $\mathcal{R}(n)$ rounds of $\mathcal{A}$ on it, communicating on each round to other player the messages transmitted over the edges $E_{A,B}$. The players accept if all the nodes of their sides accept. On each communication round the players send $\mathcal{O}(\log n)$ bits per edge of $E_{A,B}$. Hence, the nodes interchange $\mathcal{O}(R(n) \cdot |E_{A,B}| \cdot \log n) = \mathcal{O}(R(n) \cdot \log^2 n)$ bits in total. From Proposition 8.1 we obtain that $\mathcal{O}(R(n) \cdot \log^2 n) = D(\mathsf{Disj}_{n^2}) = \Omega(n^2)$. Therefore, $R(n) = \Omega(n^2/\log^2 n)$.

We now use the classical reduction from Garey, Johnson and Stockmeyer [37], used to show that 3-colorability on planar graphs of maximum degree 4 is NP-Complete. Given an n-node graph G, the reduction picks an arbitrary graph G and constructs a graph $\widetilde{G}$ of maximum degree 4, such that $G \in \textsf{Non-3-Coloring}$ if and only if $\widetilde{G} \in \textsf{Non-3-Coloring}$. Roughly, the we can construct $\widetilde{G}$ replacing each node v of degree greater than 4 by a node gadget, as the one represented in Figure 3. Let us denote d_v the degree of v. The gadget consists in a set S_v of $7(d_v-2)+1$ nodes of degree at most 4. In S_v there are d_v nodes $v_1, \dots, v_{d_v}$ satisfying that on every proper three-coloring of $\widetilde{G}$, these nodes must be colored the same. Furthermore, every coloring that assigns $v_1, \dots, v_{d_v}$ the same color can be extended to a proper 3-coloring of the whole gadget. For each node in G, consider an arbitrary ordering of its neighbors. Let us pick an edge $\{u, v\}$ of G, such that v is the i-th neighbor of u and v is the j-th neighbor of v. Then, in $\widetilde{G}$ we add the edge $\{u_i, v_j\}$. By construction satisfies that $\widetilde{G}$ has the desired properties (c.f. [37] for more details). Observe that in the worst case $\widetilde{G}$ has $\sum_{v \in V(G)} 7(d_v-2)+1 = \Theta(n^2)$ nodes.

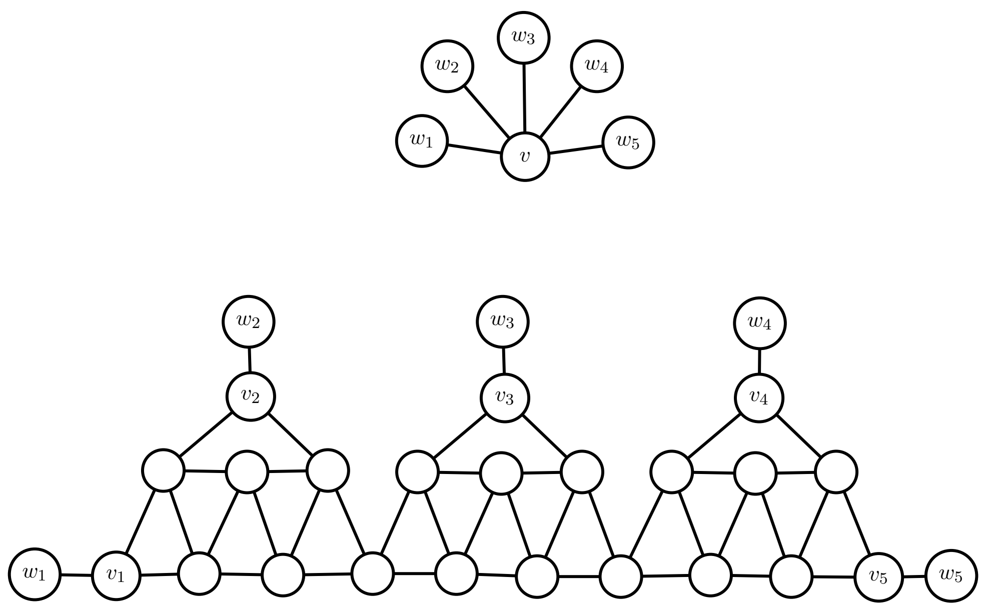

Figure 3: Node gadget. In this figure is depicted the gadget for a node of degree 5. Notice that in every proper three-coloring of the gadget the nodes $v_1, \dots, v_5$ receive the same colors. Conversely, if we color the nodes $v_1, \dots, v_5$ with the same color, we can extend the coloring into a three-coloring of all the nodes of the gadget.

The statement of the theorem is obtained by plugging the reduction construction of Garey, Johnson and Stockmeyer to the construction of Göös and Suomella. More precisely, given an instance (A, B) of $\textsf{Disj}_n$, we build graph $G_{A,B}$. Then, we apply the construction of Garey, Johnson and Stockmeyer on $G_{A,B}$ to obtain a $\widetilde{G}_{A,B}$ of maximum degree 4. Observe that $\widetilde{G}_{A,B} \in \textsf{Non-3-Coloring}$ if and only if $A \cap B = \emptyset$. We partition the nodes of $\widetilde{G}_{A,B}$ into $\widetilde{V_A}$ and $\widetilde{V_B}$. All the nodes inside gadgets created for nodes in V_A (respectively V_B) belong to $\widetilde{V_A}$ (respectively $\widetilde{V_B}$). We obtain that $\widetilde{G}_{A,B}$ satisfies conditions (C1)-(C4) and has maximum degree 4. However, $\widetilde{G}_{A,B}$ might have a large diameter.

We now reduce the diameter constructing a graph $\widehat{G}_{A,B}$ obtained from $\widetilde{G}_{A,B}$ by plugging to it two rooted binary trees T_A and T_B, where each edge of T_A and T_B is subdivided. Tree T_A has depth $\mathcal{O}(\log |\widetilde{V_A}|)$, and each leaf corresponds to a node of $\widetilde{V_A}$. The tree T_B is defined analogously for $\widetilde{V_B}$.

Finally, add an edge between the root of T_A and the root of T_B. Let us define $\widehat{V}_A = \widetilde{V}_A \cup V(T_A)$ and $\widetilde{V}_B = \widetilde{V}_B \cup V(T_B)$. We obtain that $\widehat{G}_{A,B}$ also satisfies conditions (C1)-(C4) and belongs to $\mathcal{G}$. Conditions C1 and C2 follow from the construction of $\widehat{V}_A$ and $\widehat{V}_B$. Condition C3 holds because in $\widehat{G}_{A,B}$ there is only one more edge in the cut than $\widetilde{G}_{A,B}$ consisting in the edge between the roots. Finally, for condition (C4) we observe that any 3-coloring of the nodes in $\widetilde{G}_{A,B}$ can be extended to a three coloring of the trees T_A and T_B. First, pick two different colors for the roots of T_A and T_B. Then, color arbitrarily the other nodes of degree 3 in T_A and T_B. Then, color the nodes of degree two (the subdivided edges) with a color different than their endpoints. We deduce that $\widehat{G}_{A,B}$ satisfies conditions (C1)-(C4) and belongs to $\mathcal{G}$.

Let $\mathcal{P}$ be a certification scheme (resp. let $\mathcal{A}$ be a $\mathsf{CONGEST}$ algorithm) for $\mathsf{Non\text{-}3\text{-}Coloring}$ with certificates of size $c(n)$ (resp. running in $R(n)$ rounds) under the promise that the input graph is a n-node graph contained in $\mathcal{G}$. Let $A, B \subseteq [n]$ be an instance of Disj_{n^2}. Consider the communication protocol where Alice and Bob construct $\widehat{G}_{A,B}$ and simulate $\mathcal{P}$ (resp. $\mathcal{A}$) in the same way that we explained for $G_{A,B}$. Since $\widehat{G}_{A,B}$ is a graph with $\Theta(n^2)$ nodes, we obtain that $\mathcal{O}(c(n^2) \cdot \log n) = N(\mathsf{Disj}_{n^2}) = \Omega(n^2)$, hence $c(n^2) = \Omega(n^2/\log n)$, and then $c(n) = \Omega(n/\log n)$. Similarly, we obtain that $R(n) = \Omega(\sqrt{n}/\log^2(n))$. □

Remark. Theorem 8.2 implies that the difficulty of certifying MSO properties over graphs of max degree 5 is (up to logarithmic factors) as hard as the hardest problem over that family. Indeed, a graph of max degree 5 can be encoded using $\mathcal{O}(n \log n)$ bits, e.g. by adjacency lists of all nodes. We can use such encoding to design a certification scheme that certifies any property on graphs of maximum degree 5: every node interprets its certificate as an encoding of a graph of maximum degree 5, and checks that every neighbor received the same encoding, and that its local view corresponds with the one in the encoding. If the conditions are verified, the nodes can compute any property of the input graph without any further communication.

We can also show that we cannot extend the certification of first-order properties to monadic second-order properties for planar graphs. However, in this case we obtain a weaker lowerbound.

Theorem 8.3. *Every certification scheme for membership to* $\mathsf{Non\text{-}3\text{-}Coloring}$ *under the promise that the input graph is planar requires certificates of size* $\Omega(\sqrt{n}/\log n)$ *bits.*

Similar to the proof of Theorem 8.2 we use the construction of Göös and Suomella and the reduction of Garey, Johnson and Stockmeyer for planar graphs. Given an n-node graph G, the reduction of [37] for planar graphs first pick an arbitrary embedding of G in the plane. In this embedding, there might be a number $\mathsf{cr}(G)$ of edge crossings (they must exist if G is non-planar). Then, a planar graph $\widetilde{G}$ is constructed from G by replacing on point where a pair of edges cross, by the *crossing gadget* shown in Figure 4. This gadget allows to transmit the coloring of one of the endpoints of the edge in a planar way. For each crossing, we add 13 new vertices. In the worst case, $\mathsf{cr}(G) = \Theta(n^4)$. Hence, the obtained graph has $\widetilde{G}$ has $\Theta(n^4)$ nodes.

Proof. Let $\mathcal{G}$ be the class of planar graphs. As we did for Theorem 8.2, we plug the reduction construction of Garey, Johnson and Stockmeyer to the construction of Göös and Suomella. More precisely, given an instance (A, B) of Disj_n, we build graph $G_{A,B}$ (c.f. proof of Theorem 8.2). Then, using the crossing gadget we obtain a planar graph $\widetilde{G}_{A,B}$. We partition the nodes of $\widetilde{G}_{A,B}$ into $\widetilde{V}_A$ and $\widetilde{V}_B$. All the nodes inside gadgets created for edges with both endpoint in V_A (respectively V_B)

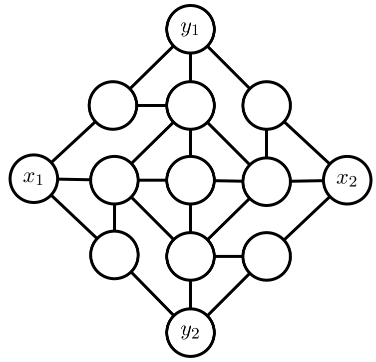

Figure 4: Crossing gadget. In every proper three-coloring of the gadget node label x_1 has the same color of node labeled x_2, as well as node labeled y_1 has the same color of y_2. Conversely, every three-coloring of x_1, x_2, y_1 and y_2 that assigns the same color to x_1 and x_2, and the same color to y_1 and y_2 can be extended into a three coloring of the gadget.

belong to $\widetilde{V_A}$ (respectively $\widetilde{V_B}$). We also consider that the nodes inside the gadgets used for the edges in the set $E_{A,B}$ of $G_{A,B}$ belong to V_A. We obtain that $\widetilde{G}_{A,B}$ satisfies conditions (C1)-(C4).

The remaining of the proof is analogous to Theorem 8.2. Let $\mathcal{P}$ be a certification scheme for Non-3-Coloring with certificates of size $c(n)$ under the promise that the input graph is planar. On input $A, B \subseteq [n]$, Alice and Bob construct $\widetilde{G}_{A,B}$ and simulate $\mathcal{P}$ in the same way that we explained for $G_{A,B}$. Since $\widetilde{G}_{A,B}$ is a graph with $\Theta(n^4)$ nodes, we obtain that $\mathcal{O}(c(n^4) \cdot \log n) = N(\mathsf{Disj}_{n^2}) = \Omega(n^2)$, hence $c(n^4) = \Omega(n^2/\log n)$, and then $c(n) = \Omega(\sqrt{n}/\log n)$. □

8.2 No Extension to First Order Logic on Graphs of Bounded Degeneracy

We now show that we cannot extend the efficient checking nor the certification of first order properties into graphs of degeneracy 2 and diameter 7. Consider the set of graphs C_6-free containing all graphs not having a cycle of six nodes as a subgraph.

Theorem 8.4. *Let $\mathcal{G}$ the class of degeneracy 2 and diameter at most 6. Every CONGEST algorithm deciding the membership in C_6-free under the promise that the input graph belongs to $\mathcal{G}$ requires $\tilde{\Omega}(\sqrt{n})$ rounds of communication.*

Every certification scheme that certifies membership to C_6-free under the promise that the input graph belongs to $\mathcal{G}$ requires certificates of size $\tilde{\Omega}(\sqrt{n})$.

Proof. The result for CONGEST algorithms and graph of degeneracy 2 was already shown in Korhonen et al. [55], which in turn use a construction of Druker et al. [20]. Roughly, the lower-bound is based on a reduction to Disj_m, for $m = \Theta(n^{1+1/2})$. Each instance $A, B \in [m]$ is associated with a $\mathcal{O}(n)$-node graph $G_{A,B}$ of degeneracy 2. The set of vertices of $G_{A,B}$ are divided in two sets of vertices, denoted V_A (and called *the nodes owned by Alice*) and V_B (*the nodes owned by Bob*), satisfying the following conditions:

(B1) The graph induced by V_A only depends on A.

(B2) The graph induced by V_B only depends on B.

(B3) The set of edges $E_{A,B}$ with one endpoint in V_A and the other in V_B contains at most $\mathcal{O}(n)$ edges, and does not depend on A and B.

(B4) $A \cap B = \emptyset$ if and only if $G_{A,B} \in C_6\text{-}\mathsf{free}$.

We need to deal with one detail. Graph $G_{A,B}$ is not necessarily of bounded diameter. To fix this issue, we build graph $\widehat{G}_{A,B}$ from $G_{A,B}$ by adding a node r, connecting it to every node in $V_A \cup V_B$, and then subdividing all the new edges two times. We obtain that every node in $V_A \cup V_B$ is at a distance of 3 from r, and then $\widehat{G}_{A,B}$ has a diameter of at most 6 and still degeneracy 2. Observe also that r cannot belong to any C_6, hence $\widehat{G}_{A,B}$ belongs to C_6-free if and only if $G_{A,B}$ does. We define $\widehat{V}_A$ as V_A plus all the new nodes of $\widehat{G}_{A,B}$, and $\widehat{V}_B$ simply as V_B. We obtain that $\widehat{G}_{A,B} \in \mathcal{G}$ and satisfies conditions (B1)-(B4).

Then, analogously to the proof of Theorem 8.2, we can use $\widehat{G}_{A,B}$ to transform any $\mathsf{CONGEST}$ algorithm into a protocol for Disj_m. More precisely, let $\mathcal{A}$ be an $R(n)$-round algorithm that decides the membership in C_6-free of a graph in $\mathcal{G}$. Given an instance $A, B \in [m]$, the players construct $G_{A,B}$, simulate $\mathcal{A}$ in the nodes they own, and in each round communicate to the other player the messages sent from nodes on their side through $E_{A,B}$; in particular, the information communicated in one round is of size $\mathcal{O}(n \log n)$. We deduce that $\mathcal{O}(n \log n \cdot R(n)) = D(\mathsf{Disj}_m) = \Omega(n^{1+1/2})$. Therefore, $R(n) = \tilde{\Omega}(\sqrt{n})$. The same construction directly implies a lower bound of $\tilde{\Omega}(n^{1/2})$ for the size of a certificate of any certification scheme for C_6-free. □

9 Conclusion

In centralized model checking, the fixed-parameter tractability of FO was extended from graphs of bounded expansion to the more general class of *nowhere dense* graphs in [47]. Notably, nowhere dense graphs represent the ultimate frontier in the program of designing fixed-parameter tractable algorithms for model checking FO formulas. Indeed, as shown in [21], for any graph class $\mathcal{G}$ that is not nowhere dense (i.e., $\mathcal{G}$ is *somewhere dense*) and closed under taking subgraphs, FO model checking on $\mathcal{G}$ is not fixed-parameter tractable (unless FPT = W[1]). The tractability frontier for FO model checking in the $\mathsf{CONGEST}$ model remains unknown. Moreover, there is no reason to expect that the tractability landscape in the distributed setting should mirror the one of the centralized setting. This raises several concrete and intriguing questions, discussed hereafter.

9.1 Distributed Computing in Nowhere Dense Graphs

Is it possible to design model-checking algorithms for FO formulas running in $\mathcal{O}(D + \mathrm{polylog}(n))$ rounds in $\mathsf{CONGEST}$ on nowhere dense graphs of diameter D? From a theoretical perspective, a negative answer to this question would be even more insightful than a positive one. More broadly, what is the precise boundary for $\mathcal{O}(D + \mathrm{polylog}(n))$-round FO model checking in the $\mathsf{CONGEST}$ model? Does it coincide with the class of bounded expansion graphs, extend to nowhere dense graphs, or even include a larger family of graphs? There are several difficulties in extending our approach to nowhere dense graphs. First, it is unclear how to implement the combinatorial arguments from [47], particularly the constructions of sparse neighborhood covers, in a distributed model. Second, the proof of [47] builds on a weak variant of Gaifman's local form, and not a quantifier elimination procedure. While this is sufficient for fixed-parameter tractability of FO model-checking, we do not know whether it could be adapted in the distributed setting.

9.2 Distributed Computing for Extensions of First-Order Logic

Another question is whether a meta-theorem with $\mathcal{O}(D + \text{polylog}(n))$ round complexity can be obtained for some extensions of FO logic on planar graphs. Theorem 8.2 shows that we cannot hope to extend Theorem 5.1 to MSO, even on planar graphs. On the other hand, in centralized computing, the fixed-parameter tractability of logics lying strictly between FO and MSO is an active area of research. For example, the extension of FO known as *separator logic*, denoted $\mathsf{FOL}+\mathsf{conn}$, was introduced independently by Schirrmacher, Siebertz, and Vigny [74], and by Bojańczyk and Pilipczuk [3]. This logic extends FO, for every $k \geq 1$, with the general predicate $\mathsf{conn}_k(x, y, z_1, \ldots, z_k)$, which evaluates to true on a graph G if the nodes corresponding to x and y are connected by a path that avoids the nodes corresponding to $\{z_1, \ldots, z_k\}$. Another example is *disjoint paths logic*, denoted $\mathsf{FOL}+\mathsf{DP}$, which extends FO with the atomic predicate $\mathsf{dp}_k(x_1, y_1, \ldots, x_k, y_k)$, expressing the existence of k pairwise internally vertex-disjoint paths between x_i and y_i for $i \in \{1, \ldots, k\}$. The main obstacle to developing distributed meta-theorems for these logics lies in the absence of distributed counterparts of the core algorithmic tools used in the centralized setting—such as tree decompositions which underlies tractability results for $\mathsf{FOL}+\mathsf{conn}$ [72], or the irrelevant vertex technique used for $\mathsf{FOL}+\mathsf{DP}$ [43].

References

[1] S. A. Amiri, S. Schmid, and S. Siebertz, *Distributed dominating set approximations beyond planar graphs*, ACM Transactions on Algorithms, 15 (2019), pp. Article No. 39, 1–18.

[2] D. A. Baterisna and Y. Chang, *Optimal local certification on graphs of bounded pathwidth*, in Proceedings of the 44th Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM Press, 2025, pp. 99–109.

[3] M. Bojanczyk and M. Pilipczuk, *Definability equals recognizability for graphs of bounded treewidth*, in Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), ACM, 2016, pp. 407–416.

[4] M. Bonamy, C. Gavoille, T. Picavet, and A. Wesolek, *Local constant approximation for dominating set on graphs excluding large minors*, in Proceedings of the 44th Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM Press, 2025, pp. 77–87.

[5] É. Bonnet, E. J. Kim, S. Thomassé, and R. Watrigant, *Twin-width I: tractable FO model checking*, in 61st IEEE Symposium on Foundations of Computer Science (FOCS), 2020, pp. 601–612.

[6] R. B. Borie, R. G. Parker, and C. A. Tovey, *Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families*, Algorithmica, 7 (1992), pp. 555–581.

[7] N. Bousquet, L. Feuilloley, and T. Pierron, *Local certification of graph decompositions and applications to minor-free classes*, in 25th International Conference on Principles of Distributed Systems (OPODIS), vol. 217 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 22:1–22:17.

[8] K. Censor-Hillel, *Distributed subgraph finding: Progress and challenges (invited talk)*, in 48th International Colloquium on Automata, Languages, and Programming (ICALP), vol. 198 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 3:1–3:14.

[9] Y.-J. Chang, *Efficient distributed decomposition and routing algorithms in minor-free networks and their applications*, in 42nd ACM Symposium on Principles of Distributed Computing (PODC), 2023, pp. 55–66.

[10] Y.-J. Chang, S. Pettie, T. Saranurak, and H. Zhang, *Near-optimal distributed triangle enumeration via expander decompositions*, Journal of the ACM (JACM), 68 (2021), pp. 1–36.

[11] L. Cook, E. J. Kim, and T. Masařík, *A tight meta-theorem for local certification of* mso_2 *properties within bounded treewidth graphs*, in Proceedings of the 44th Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM Press, 2025, pp. 110–120.

[12] B. Courcelle, *The monadic second-order logic of graphs. I. Recognizable sets of finite graphs*, Inf. Comput., 85 (1990), pp. 12–75.

[13] A. Czygrinow, M. Hańćkowiak, and W. Wawrzyniak, *Fast distributed approximations in planar graphs*, in 22nd International Symposium on Distributed Computing (DISC), vol. 5218 of Lecture Notes in Computer Science, Springer, 2008, pp. 78–92.

[14] ——, *Distributed approximation algorithms for the minimum dominating set in* k_h*-minor-free graphs*, in 29th International Symposium on Algorithms and Computation (ISAAC), vol. 123 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018, pp. 22:1–22:12.

[15] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer, *Distributed verification and hardness of distributed approximation*, SIAM J. Comput., 41 (2012), pp. 1235–1265.

[16] A. Dawar, M. Grohe, and S. Kreutzer, *Locally excluding a minor*, in 22nd IEEE Symposium on Logic in Computer Science (LICS), 2007, pp. 270–279.

[17] E. D. Demaine, F. Reidl, P. Rossmanith, F. S. Villaamil, S. Sikdar, and B. D. Sullivan, *Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs*, J. Comput. Syst. Sci., 105 (2019), pp. 199–241.

[18] M. Dory, Y. Efron, S. Mukhopadhyay, and D. Nanongkai, *Distributed weighted min-cut in nearly-optimal time*, in 53rd ACM Symposium on Theory of Computing (STOC), 2021, pp. 1144–1153.

[19] P. G. Drange, M. S. Dregi, F. V. Fomin, S. Kreutzer, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, F. Reidl, F. S. Villaamil, S. Saurabh, S. Siebertz, and S. Sikdar, *Kernelization and sparseness: The case of dominating set*, in Proceedings of the 33rd International Symposium on Theoretical Aspects of Computer Science (STACS), vol. 47 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016, pp. 31:1–31:14.

[20] A. Drucker, F. Kuhn, and R. Oshman, *On the power of the congested clique model*, in 33rd ACM Symposium on Principles of Distributed Computing (PODC), 2014, pp. 367–376.

[21] Z. Dvořák, D. Král', and R. Thomas, *Testing first-order properties for subclasses of sparse graphs*, J. ACM, 60 (2013), p. 36.

[22] H.-D. Ebbinghaus and J. Flum, *Finite Model Theory: First Edition*, Springer Berlin Heidelberg, 1995.

[23] 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 Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP), vol. 80 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017, pp. 63:1–63:14.

[24] R. L. Epstein, *Classical mathematical logic: The semantic foundations of logic*, Princeton University Press, 2011.

[25] G. Even, O. Fischer, P. Fraigniaud, T. Gonen, R. Levi, M. Medina, P. Montealegre, D. Olivetti, R. Oshman, I. Rapaport, and I. Todinca, *Three notes on distributed property testing*, in 31st International Symposium on Distributed Computing (DISC), vol. 91 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 15:1–15:30.

[26] S. Faour, M. Ghaffari, C. Grunau, F. Kuhn, and V. Rozhon, *Local distributed rounding: Generalized to mis, matching, set cover, and beyond*, in ACM-SIAM Symposium on Discrete Algorithms (SODA), 2023, pp. 4409–4447.

[27] L. Feuilloley, N. Bousquet, and T. Pierron, *What can be certified compactly? compact local certification of MSO properties in tree-like graphs*, in 41st ACM Symposium on Principles of Distributed Computing (PODC), 2022, pp. 131–140.

[28] L. Feuilloley and P. Fraigniaud, *Survey of distributed decision*, Bull. EATCS, 119 (2016).

[29] M. Fischer, M. Ghaffari, and F. Kuhn, *Deterministic distributed edge-coloring via hypergraph maximal matching*, in 58th IEEE Symposium on Foundations of Computer Science (FOCS), 2017, pp. 180–191.

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

[31] F. V. Fomin, P. Fraigniaud, P. Montealegre, I. Rapaport, and I. Todinca, *Distributed model checking on graphs of bounded treedepth*, in 38th International Symposium on Distributed Computing (DISC), vol. 319 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 25:1–25:20.

[32] P. Fraigniaud, F. Mazoit, P. Montealegre, I. Rapaport, and I. Todinca, *Distributed certification for classes of dense graphs*, in 37th International Symposium on Distributed Computing (DISC), vol. 281 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 20:1–20:17.

[33] P. Fraigniaud, P. Montealegre, I. Rapaport, and I. Todinca, *A meta-theorem for distributed certification*, Algorithmica, 86 (2024), pp. 585–612.

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

[35] H. Gaifman, *On local and non-local properties*, in Studies in Logic and the Foundations of Mathematics, vol. 107, Elsevier, 1982, pp. 105–135.

[36] J. A. Garay, S. Kutten, and D. Peleg, *A sub-linear time distributed algorithm for minimum-weight spanning trees*, in 34th IEEE Symposium on the Foundations of Computer Science (FOCS), 1993, pp. 659–668.

[37] M. R. Garey, D. S. Johnson, and L. Stockmeyer, *Some simplified np-complete problems*, in 6th ACM symposium on Theory of computing, 1974, pp. 47–63.

[38] M. Ghaffari and B. Haeupler, *Distributed algorithms for planar networks I: planar embedding*, in 35th ACM Symposium on Principles of Distributed Computing (PODC), 2016, pp. 29–38.

[39] ——, *Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut*, in 27th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2016, pp. 202–219.

[40] ——, *Low-congestion shortcuts for graphs excluding dense minors*, in 40th ACM Symposium on Principles of Distributed Computing (PODC), 2021, pp. 213–221.

[41] M. Ghaffari and F. Kuhn, *Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition*, in 62nd IEEE Symposium on Foundations of Computer Science (FOCS), 2021, pp. 1009–1020.

[42] M. Ghaffari and M. Parter, *Near-optimal distributed dfs in planar graphs*, in 31st International Symposium on Distributed Computing (DISC), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017, pp. 21:1–21:16.

[43] P. A. Golovach, G. Stamoulis, and D. M. Thilikos, *Model-checking for first-order logic with disjoint paths predicates in proper minor-closed graph classes*, in Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2023, pp. 3684–3699.

[44] M. Göös and J. Suomela, *Locally checkable proofs in distributed computing*, Theory Comput., 12 (2016), pp. 1–33.

[45] M. Grohe, *Logic, graphs, and algorithms*, in Logic and Automata: History and Perspectives, in Honor of Wolfgang Thomas, vol. 2 of Texts in Logic and Games, Amsterdam University Press, 2008, pp. 357–422.

[46] M. Grohe and S. Kreutzer, *Methods for algorithmic meta theorems*, in Model Theoretic Methods in Finite Combinatorics - AMS-ASL Joint Special Session, vol. 558, AMS, 2009, pp. 181–206.

[47] M. Grohe, S. Kreutzer, and S. Siebertz, *Deciding first-order properties of nowhere dense graphs*, in 46th ACM Symposium on Theory of Computing (STOC), 2014, pp. 89–98.

[48] B. Haeupler, T. Izumi, and G. Zuzic, *Low-congestion shortcuts without embedding*, in 2016 ACM Symposium on Principles of Distributed Computing, 2016, pp. 451–460.

[49] B. Haeupler, T. Izumi, and G. Zuzic, *Near-optimal low-congestion shortcuts on bounded parameter graphs*, in 30th International Symposium on Distributed Computing (DISC), vol. 9888 of LNCS, Springer, 2016, pp. 158–172.

[50] B. Haeupler, J. Li, and G. Zuzic, *Minor excluded network families admit fast distributed algorithms*, in ACM Symposium on Principles of Distributed Computing (PODC), ACM, 2018, pp. 465–474.

[51] S. Har-Peled and K. Quanrud, *Approximation algorithms for polynomial-expansion and low-density graphs*, SIAM J. Comput., 46 (2017), pp. 1712–1744.

[52] O. Heydt, S. Kublenz, P. Ossona de Mendez, S. Siebertz, and A. Vigny, *Distributed domination on sparse graph classes*, European Journal of Combinatorics, 123 (2025), p. 103773.

[53] W. Hodges, *Model theory*, Cambridge university press, 1993.

[54] T. Izumi, N. Kitamura, T. Naruse, and G. Schwartzman, *Fully polynomial-time distributed computation in low-treewidth graphs*, in 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, 2022, pp. 11–22.

[55] J. H. Korhonen and J. Rybicki, *Deterministic Subgraph Detection in Broadcast CONGEST*, in 21st International Conference on Principles of Distributed Systems (OPODIS 2017), vol. 95 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 4:1–4:16.

[56] A. Korman, S. Kutten, and D. Peleg, *Proof labeling schemes*, Distributed Comput., 22 (2010), pp. 215–233.

[57] S. Kreutzer, *Algorithmic meta-theorems*, in Finite and Algorithmic Model Theory, J. Esparza, C. Michaux, and C. Steinhorn, eds., vol. 379 of London Mathematical Society Lecture Note Series, Cambridge University Press, 2011, pp. 177–270.

[58] S. Kublenz, S. Siebertz, and A. Vigny, *Constant round distributed domination on graph classes with bounded expansion*, in 28th International Colloquium on Structural Information and Communication Complexity (SIROCCO), vol. 12810 of Lecture Notes in Computer Science, Springer, Cham, 2021, pp. 334–351.

[59] E. Kushilevitz and N. Nisan, *Communication Complexity*, Cambridge University Press, 1997.

[60] C. Lenzen and B. Patt-Shamir, *Fast routing table construction using small messages*, in ACM Symposium on Theory of Computing Conference (STOC), 2013, pp. 381–390.

[61] C. Lenzen, Y.-A. Pignolet, and R. Wattenhofer, *Distributed minimum dominating set approximations in restricted families of graphs*, Distributed Computing, 26 (2013), pp. 119–137.

[62] N. Linial, *Locality in distributed graph algorithms*, SIAM J. Comput., 21 (1992), pp. 193–201.

[63] V. Manoharan and V. Ramachandran, *Computing minimum weight cycle in the CONGEST model*, in 43rd ACM Symposium on Principles of Distributed Computing (PODC), 2024, pp. 182–193.

[64] D. Nanongkai, *Distributed approximation algorithms for weighted shortest paths*, in ACM Symposium on Theory of Computing (STOC), 2014, pp. 565–573.

[65] M. Naor and L. J. Stockmeyer, *What can be computed locally?*, SIAM J. Comput., 24 (1995), pp. 1259–1277.

[66] J. Nešetřil and P. Ossona de Mendez, *Linear time low tree-width partitions and algorithmic consequences*, in 38th Annual ACM Symposium on Theory of Computing (STOC), 2006, pp. 391–400.

[67] J. Nešetřil and P. Ossona de Mendez, *Sparsity – Graphs, Structures, and Algorithms*, vol. 28 of Algorithms and combinatorics, Springer, 2012.

[68] ——, *A distributed low tree-depth decomposition algorithm for bounded expansion classes*, Distributed Comput., 29 (2016), pp. 39–49.

[69] A. Panconesi and R. Rizzi, *Some simple distributed algorithms for sparse networks*, Distributed Computing, 14 (2001), pp. 97–100.

[70] D. Peleg, *Distributed computing: a locality-sensitive approach*, SIAM, 2000.

[71] D. Peleg and V. Rubinovich, *A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction*, SIAM J. Comput., 30 (2000), pp. 1427–1442.

[72] M. Pilipczuk, N. Schirrmacher, S. Siebertz, S. Toruńczyk, and A. Vigny, *Algorithms and data structures for first-order logic with connectivity under vertex failures*, in Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP), vol. 229 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022, pp. 102:1–102:18.

[73] M. Pilipczuk, S. Siebertz, and S. Torunczyk, *Parameterized circuit complexity of model-checking on sparse structures*, in 33rd ACM/IEEE Symposium on Logic in Computer Science (LICS), 2018, pp. 789–798.

[74] N. Schirrmacher, S. Siebertz, and A. Vigny, *First-order logic with connectivity operators*, in Proceedings of the 30th EACSL Annual Conference on Computer Science Logic (CSL), vol. 216 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022, pp. 34:1–34:17.

[75] D. Seese, *Linear time computable problems and first-order descriptions*, Mathematical Structures in Computer Science, 6 (1996), pp. 505–526.

[76] S. Siebertz and A. Vigny, *Parameterized distributed complexity theory: A logical approach*, CoRR, abs/1903.00505 (2019).