What Can Be Certified Compactly?

Compact local certification of MSO properties in tree-like graphs

Nicolas Bousquet
Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205
Villeurbanne, France
nicolas.bousquet@univ-lyon1.fr

Laurent Feuilloley
Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205
Villeurbanne, France
laurent.feuilloley@univ-lyon1.fr

Théo Pierron
Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205
Villeurbanne, France
theo.pierron@univ-lyon1.fr

ABSTRACT

Local certification consists in assigning labels (called *certificates*) to the nodes of a network to certify a property of the network or the correctness of a data structure distributed on the network. The verification of this certification must be local: a node typically sees only its neighbors in the network. The main measure of performance of a certification is the size of its certificates.

In 2011, Göös and Suomela identified $\Theta(\log n)$ as a special certificate size: below this threshold little is possible, and several key properties do have certifications of this type. A certification with such small certificates is now called a *compact local certification*, and it has become the gold standard of the area, similarly to polynomial time for centralized computing. A major question is then to understand which properties have $O(\log n)$ certificates, or in other words: what is the power of compact local certification?

Recently, a series of papers have proved that several well-known network properties have compact local certifications: planarity, bounded-genus, etc. But one would like to have more general results, *i.e.* meta-theorems. In the analogous setting of polynomial-time centralized algorithms, a very fruitful approach has been to prove that restricted types of problems can be solved in polynomial time in graphs with restricted structures. These problems are typically those that can be expressed in some logic, and the graph structures are those with bounded width or depth parameters. We take a similar approach and prove several meta-theorems for local certification.

More precisely, the logic we use is MSO, the most classic fragment for logics on graphs, where one can quantify over vertices and sets of vertices, and consider adjacency between vertices. We prove the relevance of this choice in the context of local certification by first considering properties of trees. On trees, we prove that MSO properties can be certified with labels of constant size, whereas the typical non-MSO property of isomorphism requires $\tilde{\Omega}(n)$ size certificates (where $\tilde{\Omega}$ hides polylogarithmic factors). We then move on to graphs of bounded treedepth, a well-known parameter that basically measures how far a graph is from a star. We first prove that an optimal certification for bounded treedepth uses certificates of size $\Theta(\log n)$, and then prove that in bounded treedepth graphs, every MSO property has a compact certification.

To establish our results, we use a variety of techniques, originating from model checking, tree automata theory, communication complexity, and combinatorics.

PODC '22, July 25–29, 2022, Salerno, Italy

ACM ISBN 978-1-4503-9262-4/22/07…$15.00
https://doi.org/10.1145/3519270.3538416

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**.

KEYWORDS

Local certification, Proof-labeling scheme, Model Checking, Treedepth, distributed decision, MSO logic

ACM Reference Format:
Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. 2022. What Can Be Certified Compactly?: Compact local certification of MSO properties in tree-like graphs. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing (PODC '22), July 25–29, 2022, Salerno, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3519270.3538416

1 INTRODUCTION

1.1 Local certification

In this work, we are interested in the locality of graph properties. For example, consider the property "the graph has maximum degree three". We say that this property can be checked locally, because if every node checks that it has at most three neighbors (which is a local verification), then the graph satisfies the property (which is a global statement). Most graph properties of interest are not local. For example, to decide whether a graph is acyclic, or planar, the vertices would have to look arbitrarily far in the graph. Some properties can be seen as local or not, depending on the exact definition. For example, having a diameter at most 2, is a property that can be checked locally if we consider that looking at distance 3 is local, but not if we insist on inspecting only the neighbors of a vertex.

As distributed computing is subject to faults and changes in the network, it is essential to be able to check properties of the network or of distributed data structures efficiently. Since most properties are not locally checkable, we would like to have a mechanism to circumvent this shortcoming. Local certification is such a mechanism, in the sense that it allows any graph property to be checked locally. For a given property, a local certification is described by a certificate assignment and a verification algorithm: each node receives a certificate, reads the certificates of its neighbors and then runs a verification algorithm. This algorithm decides whether the node accepts or rejects the certification. If the graph satisfies the property, then there should be a certificate assignment such that

all the nodes accept. Otherwise, in each assignment, there must be at least one node that rejects.

In recent years, the area of local certification has attracted a lot of attention, and we refer to [17] and [16] for respectively a complexity-theory oriented survey, and an introduction to the area.

1.2 Understanding the power of compact local certification

It is known that any property can be certified with $O(n^2)$ bits certificates, where n is the total number of vertices. This is because one can simply give the full description of the graph to every node, which can then check that the property holds in the graph described, and that the graph description is correct locally, and identical between neighbors. This $O(n^2)$ size is extremely large, and the main goal of the study of local certification is to minimize the size of the certificate, expressed as a number of bits per vertex, as a function of n. In addition to the optimization motivation originating from distributed self-stabilizing algorithms, establishing the minimum certificate size also has a more theoretical appeal. Indeed, the optimal certification size of a property can be seen as a measure of its locality: the smaller the labels, the less global information we need to allow local verification, the more local the property.

In [25], Göös and Suomela identified that $\Theta(\log n)$ was a special certificate size. Indeed, on the one hand, for most properties, one cannot hope to go below this size per node. For example, certifying acyclicity requires $\Omega(\log n)$ [25, 28]. On the other hand, once we have a logarithmic number of bits, we can encode identifiers, and distances, which is enough for spanning trees, an essential tool in certification. A certification with $\Theta(\log n)$ bits is now called a *compact local certification*, and it has become the gold standard of the area. Recently, planarity and more generally embeddability on bounded-genus surfaces, and H-minor-freeness for graphs H of size at most 4, have been proved to have such compact certifications [4, 14, 19, 20].

Unfortunately, not every property has a compact certification. For example, having a non-trivial automorphism or not being 3-colorable are properties that cannot be certified with $o(n^2)$ bits [25]. Even surprisingly simple properties, such as having diameter at most 2, cannot be certified with a sublinear number of bits per vertex (up to logarithmic factors), if we only allow the local verification to be at distance one [6]. This raises the following question:

Main question. *What are the graph properties that admit a compact local certification?*

2 OUR APPROACH, RESULTS, AND TECHNIQUES

2.1 A systematic model checking approach

As mentioned above, many specific graph properties such as planarity or small-diameter have been studied in the context of local certification. In this paper, we take a more systematic approach, inspired by model checking, by considering classes of graph properties. We are interested in establishing theorems of the form: "all the properties that can be expressed in some formalism X have a compact certification".

In this paper, we will consider properties that can be expressed by sentences from monadic second order logic (MSO), which is a standard logic for expressing graph properties. These are formed from atomic predicates that test equality or adjacency of vertices and allowing boolean operations and quantifications on vertices, edges, and sets of vertices or edges. Now, certifying a given property consists in certifying that a graph is a positive instance of the so-called graph model checking problem for the corresponding sentence φ:

- Input: A graph G
- Output: Yes, if and only if, G satisfies φ.

2.2 The generic case

Let us first discuss what such a meta-theorem must look like when we do not restrict the class of graphs we consider. As we already mentioned, graphs of diameter at most 2 cannot be certified with sublinear certificates [6]. This can be expressed with the following sentence, where $x - y$ means $(x, y) \in E$:

$$\forall x \forall y (x = y \vee x - y \vee \exists z (x - z \wedge z - y))$$

This sentence is very simple: it is a first order sentence (a special case of MSO), it has quantifier depth three and there is only one quantifier alternation (two standard complexity measures for FO sentences which respectively counts the maximum number of nested quantifiers and the number of alternations between blocks of existential and universal quantifiers). Therefore, there exists very simple first order logic sentences which cannot be certified efficiently, hence there is no room for a generic $O(\log n)$ result.

Note that if we allowed the vertices to see at a larger (but still constant) distance in the graph, then we could verify diameter 2 without certificates. In order to prevent such phenomenon, and because it is more relevant in terms of message complexity, in the whole paper, the radius of the views of the vertices is fixed to 1 (in other words, a node can read the IDs and the certificates of all its neighbors, but cannot see which edges are incident to these vertices). We discuss that aspect in more detail in the full version.

Another example is given by triangle-freeness, which can be expressed by the following sentence:

$$\forall x \forall y \forall z \neg (x - y \wedge y - z \wedge x - z)$$

This sentence also has rank 3 and no quantifier alternation. Proposition 5 of [9] proves that certifying that a graph is triangle-free requires $n/e^{O(\sqrt{n})}$ bits, via reduction to multi-party communication complexity inspired by [10].

The only possible way to simplify the sentences would consist in only having at most two nested quantifiers or not allowing universal quantifiers. In these cases, the following holds:

Lemma 2.1. *FO sentences with quantifier depth at most 2 can be certified with $O(\log n)$ bits. Existential FO sentences (*i.e. *whose prenex normal form has only existential quantifiers) can be certified with $O(\log n)$ bits.*

For the FO sentences with quantifiers of depth at most 2, we can prove that the only non-trivial properties that can be expressed are a vertex being dominant (adjacent to all other vertices) or the graph being a clique. These are easy to certify with $O(\log n)$ bits, *c.f.* the full proofs in the full version.

2.3 The case of trees

In order to go beyond the formulas of Lemma 2.1, we use the approach that is classic in centralized computing: restricting the class of the graphs considered. This approach is relevant in our setting, as exemplified by the certification of diameter, which becomes much easier if we restrict the graphs to *e.g.* trees. Indeed, in this case we can use a spanning tree to point to a central vertex (or edge), that becomes a root (or root-edge), and keep at every vertex both its distance to the root and the depth of its subtree. This certification can be checked by simple distance comparisons, and it uses $O(\log n)$ bits. The first of our main results is that we can actually get a better bound (constant certificates) for all MSO properties on trees.

Theorem 2.2. *Any MSO formula can be certified on trees with certificates of size $O(1)$.*

One can wonder if we can extend this statement to a significantly wider logic. We answer by the negative by proving that some typical non-MSO properties cannot be certified with certificates of sublinear sizes even on trees of bounded depth.

Theorem 2.3. *Certifying the trees that have an automorphism without fixed-point requires certificates of size $\tilde{\Omega}(n)$ (where $\tilde{\Omega}$ hides polylogarithmic factors), even if we restrict to trees of bounded depth.*

2.4 The case of bounded treedepth graphs

In centralized model checking, a classic meta-theorem of Courcelle [8] establishes that all the problems expressible in MSO can be solved is fixed-parameter tractable (FPT) with respect to the treewidth t of the input graph. One of the drawbacks of Courcelle's theorem is that, while FPT, the complexity of the algorithm involves a $2^{2^{\cdots^t}}$ dependency in t, whose height depends on the MSO sentence. Motivated by this unavoidable non-elementary dependence in the formula in Courcelle's theorem [23], Gajarský and Hliněný [24] designed a linear-time FPT algorithm for MSO-model checking with elementary dependency in the sentence (*i.e.* with a power tower of constant height), by paying the price of considering a smaller class of graphs, namely graphs of bounded treedepth. Their result is essentially the best possible, as shown soon after in [29].

One can wonder if some Courcelle-like result holds for certification. Namely, is it possible to certify any MSO-formula on graphs of bounded treewidth with certificates of size $O(\log n)$? A useful step in this direction is to try to understand the complexity of certifying bounded treewidth graphs. Prior to our work, certifying this property was an open problem (see below for some recent development regarding this question). Following the approach of Gajarský and Hliněný, we considered the question of certifying graphs of bounded treedepth. We hoped that these more restricted classes of graphs would be easier to certify, especially since treedepth involves more local insights on the structure of the graphs (compared to treewidth), which is more suited to the distributed setting.

On this question, we prove that one can locally check that a graph has treedepth at most t with logarithmic-size certificates.

Theorem 2.4. *We can certify that a graph has treedepth at most t with $O(t \log n)$ bits.*

We also show that Theorem 2.4 is optimal in n, in the sense that certifying treedepth at most t requires $\Omega(\log n)$ bits, even for small t.

Theorem 2.5. *Certifying that the treedepth of the graph is at most t requires $\Omega(\log n)$ bits, for any $t \geq 5$.*

This result contrasts with the fact that certifying trees of depth t can be done with $O(\log t)$ bits (thus independent of n), by simply encoding distances to the root.

The next problem in line is then MSO-model checking for graphs of bounded treedepth. In such classes, it happens that MSO and FO have the same expressive power [11]: for every t and every MSO sentence, there exists a FO sentence satisfied by the same graphs of treedepth at most t.

Theorem 2.6. *Every FO (and hence MSO) sentence φ can be locally certified with $O(t \log n + f(t, \varphi))$-bit certificates on graphs of treedepth at most t (for some function f).*

This result, as well as Theorem 2.2, holds for MSO properties about the structure of the graphs, but our techniques also work for graphs with constant-size inputs, in the spirit of locally checkable labelings [30].

Inspired by our results and techniques, Fraigniaud, Montealegre, Rapaport, and Todinca, very recently proved that it is possible to certify MSO properties in bounded treewidth graphs, with certificates of size $\Theta(\log^2 n)$ [22]. Replacing treedepth by treewidth is very interesting, as the second parameter is more general and well-known, but it comes at the cost of certificates of size $\Theta(\log^2 n)$, hence not a compact certification *per se*. It is a fascinating question whether this is optimal or can be reduced down to $O(\log n)$.

Theorem 2.6 has an interesting corollary for the certification of graphs with forbidden minors. An important open question in the field of local certification is to establish whether all the graph classes defined by a set of forbidden minors have a compact certification (e.g. Open problem 4 in [16]). Note that this question generalizes the results about planarity and bounded-genus graphs of [14, 19, 20]. Very recently, Bousquet, Feuilloley and Pierron proved that the answer is positive for all minors of size at most 4 [4], but the question is still wide open for general minors. Theorem 2.6 leads to the following result, where P_t and C_t are respectively the path and the cycle of length t.

Corollary 2.7. *For all t, P_t-minor-free graphs and C_t-minor-free graphs can be certified with $O(\log n)$-bit certificates.*

Still related to the certification of minors, Esperet and Norin [15] (generalizing a result by Elek [12]) proved very recently that certifying that a graph belongs to a minor-closed class or is far from it (in the sense of the edit distance, as in property testing) can be done with constant size certificate. Using our certification of bounded treedepth, they generalize this result to all monotone properties of minor-closed classes, with $O(\log n)$-size certificates.

Let us finish this overview, by mentioning a related line of research. A recent series of papers have characterized diverse logics on graphs by various models of distributed local computation, in a similar way as descriptive complexity in centralized computing [27]. In this area, a paper that is especially relevant to us is [34], which proves that MSO logic on graphs is equivalent to a model called

alternating distributed graph automata. These are actually quite different from our model, with several provers, more constrained local computation, and more general output functions. We describe this model and discuss the differences in more details in the full version.

2.5 A glimpse of our techniques and the organization of the paper

We use a variety of techniques to prove our results, and except for a section of preliminaries (Section 3), each upcoming section of this paper corresponds to one technique. First, we show how to prove the constant size MSO certification in trees (Theorem 2.2) by seeing the certificates as a state labeling by the right type of tree automata, and then using the known logic-automata correspondence to derive our result. We will discuss in the appendix how this automata view can be an inspiration to generalize locally checkable languages (LCLs) [30] beyond bounded degree graphs.

The proof of the certification of bounded treedepth (Theorem 2.4) is in Section 5, and uses spanning tree certification along with an analysis of interplay between ancestors in the decompositions and the separators in the graph. Given this certification, we certify MSO properties (Theorem 2.6) via kernelization. In more details, we show that for any graph there exists a kernel, that is, a graph that satisfies the exact same set of MSO properties, whose size only depends on the formula and on the treedepth (and in particular not in the size of the original graph). We show that this kernel can be certified locally, which is enough for our purpose, as we can finish by describing the full kernel to all nodes, and let them check the MSO property at hand.

Finally, in Section 7, we describe the proofs of our two lower bounds (Theorem 2.3 and 2.5) by reduction from two-party nondeterministic communication complexity.

To our knowledge, it is the first time that automata tools, kernelization, and reductions from communication complexity for the $\Theta(\log n)$ regimes, are used in local certification.

3 PRELIMINARIES

All the graphs considered in this paper are connected, loopless (no edge with equal endpoints) and have at least one vertex.

3.1 Treedepth

Treedepth was introduced by Nešetřil and Ossona de Mendez in [31] as a graph parameter inducing a class where model checking is more efficient. In the last ten years, this graph parameter received considerable attention (see [32] for a book chapter about this parameter). Treedepth is related to other important width parameters in graphs. In particular, it is an upper bound on the pathwidth, which is essential in the study of minors [35] and interval graphs [2].

Let T be a rooted tree. A vertex u is an *ancestor* of v in T, if u is on the path between v and the root. We say that v is a *descendant* of u if u is an ancestor of v.

Definition 3.1 ([31]). The *treedepth* of a graph G is the minimum height of a forest F on the same vertex set as G, such that for every edge (u, v) of the graph G, u is an ancestor or a descendant of v in the forest.

Since in our setting G is connected, F is necessarily a tree, called an *elimination tree*. In a more logic-oriented perspective, it is called a *model* of the graph. If the tree has depth at most k, it is a *k-model of G* (see Figure 1). Note that there might be several elimination trees.

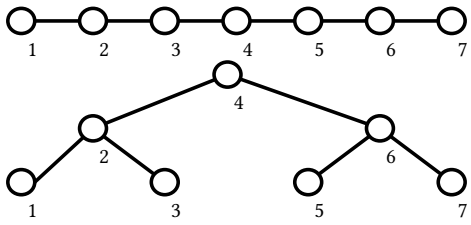

Figure 1: An example of an elimination tree. On top, the graph G, that is a path on seven vertices, and on the bottom an elimination tree T of this graph. Since this tree has depth 2, the path has treedepth at most 2, and this is actually optimal.

Let us fix an elimination tree. A *vertex of G has depth d*, if it has depth d in the elimination tree. For any vertex v, let G_v be the subgraph of G induced by the vertices in the subtree of T rooted in v. Note that, for the root r, $G_r = G$. Now, a model T of G is *coherent* if, for every vertex v, the vertices of the subforest rooted in v form a connected component in G. In other words, for every child w of v, there exists a vertex x of the subtree rooted in w that is connected to v.

We have the following simple result, that we prove in the full version for completeness, as well as the fact that any graph has a coherent model.

Remark 1. *Let T be a coherent d-model of a connected graph G and u be a vertex of G. Then G_u induces a connected subgraph.*

3.2 FO and MSO logics

Graphs can be seen as relational structures on which properties can be expressed using logical sentences. The most natural formalism considers a binary predicate that tests the adjacency between two vertices. Allowing standard boolean operations and *quantification on vertices*, we obtain the *first-order logic* (FO for short) on graphs. Formally, a FO formula is defined by the following grammar:

$$x = y \mid x - y \mid \neg F \mid F \wedge F \mid F \vee F \mid \forall x F \mid \exists x F$$

where x, y lie in a fixed set of variables. Except for $x - y$, which denotes the fact that x and y are adjacent, the semantic is the classic one. Given a FO sentence F (i.e. a formula where each variable falls under the scope of a corresponding quantifier) and a graph G, we write $G \vDash F$ when the graph G satisfies the sentence F, which is defined in the natural way.

MSO logic is an enrichment of FO, where we allow *quantification on sets of vertices*[1], usually denoted by capital variables, and we add the membership predicate $x \in X$. We skip the details here since for

[1]Sometimes MSO sentences are also allowed to quantify on set of edges. We do not discuss the matter any further since for our purposes (i.e. on trees or bounded treedepth graphs), it is known that quantifying on edges does not increase the expressive power of the sentences.

bounded treedepth graphs, it is known that FO and MSO have the same expressive power.

Theorem 3.2 ([26]). *For every integer d and MSO sentence φ, there exists a FO sentence ψ such that φ and ψ are satisfied by the same set of graphs of treedepth at most d.*

In Section 6, we are looking for a kernelization result for the model checking problem, where the kernel is checkable with small certificates. In particular, given a sentence φ and a graph G, we have to prove that the graph H output by our kernelization algorithm satisfies φ if and only if so does G. We actually show a stronger result, namely that for every integer k and every graph G, there exists a graph H_k satisfying the same set of sentences with at most k nested quantifiers as G. In that case, we write $G \simeq_k H_k$. This yields the required result when k is quantifier depth of φ.

The canonical tool to prove equivalence between structures is the so-called *Ehrenfeucht-Fraïssé game*. This game takes place between two players, Spoiler and Duplicator. The arena is given by two structures (here, graphs) and a number k of rounds. At each turn, Spoiler chooses a vertex in one of the graphs, and Duplicator has to answer by picking a vertex in the other graph. Let the positions played in the first (resp. second) graph at turn i be $u_1, \ldots, u_i$ (resp. $v_1, \ldots, v_i$). Spoiler wins at turn i if the mapping $u_j \mapsto v_j$ is not an isomorphism between the subgraphs induced by $\{u_1, \ldots, u_i\}$ and $\{v_1, \ldots, v_i\}$. If Spoiler does not win before the end of the k-th turn, then Duplicator wins. The main result about this game is the following, which relates winning strategies with equivalent structures for $\simeq_k$.

Theorem 3.3. *Let G, H be two graphs and k be an integer. Duplicator has a winning strategy in the k-round Ehrenfeucht-Fraïssé game on (G, H) if and only if $G \simeq_k H$.*

See [37] for a survey on Ehrenfeucht-Fraïssé games and its applications in computer science.

3.3 Local certification: definitions and basic techniques

We assume that the vertices of the graph are equipped with unique identifiers, also called IDs, in a polynomial range $[1, n^k]$ (k being a constant). Note that an ID can be written on $O(\log n)$ bits.

In this paper, a local certification is described by a local verification algorithm, which is an algorithm that takes as input the identifiers and the labels of a node and of its neighbors, and outputs a binary decision, usually called *accept* or *reject*. A local certification of a property is a local verification algorithm such that:

- If the graph satisfies the property, then there exists a label assignment, such that the local verification algorithm accepts at every vertex.
- If the graph does not satisfy the property, then for every label assignment, there exists at least one vertex that rejects.

A graph that satisfies the property is a *yes-instance*, and a graph that does not satisfy the property is a *no-instance*. The labels are called *certificates*. It is equivalent to consider that there is an entity, called the *prover*, assigning the labels (a kind of external oracle). The size $f(n)$ of a certification is the size of its largest label for graphs of size n. The certification size of a property or a set of properties is the (asymptotic) minimum size of a local certification.

A standard tool for local certification is to encode and certify a spanning tree, which can be done efficiently.

Proposition 3.4. *One can locally encode and certify a spanning tree with $O(\log n)$ bits. The number of vertices in the graph can also be certified with $O(\log n)$ bits.*

The idea of the certification of the spanning tree is to root the tree, and then to label the vertices with the distance to the root (to ensure acyclicity) and the ID of the root (to ensure connectivity). To certify the number of vertices, one also labels the vertices with the number of nodes in their subtrees. We refer to the tutorial [16], for intuitions, proofs, and history of these tools.

4 MSO CERTIFICATION ON TREES VIA TREE AUTOMATA

Theorem 2.2. *Any MSO formula can be certified on trees with certificates of size $O(1)$.*

The full formal proof of Theorem 2.2 is deferred to the full version, but we discuss the intuition here. The idea of the proof is to adapt results from the tree automata literature. Let us give some intuition with classic (word) automata. Consider a word as a directed path whose edges are labeled with letters, then this word is recognized by an automaton if we can label the vertices with states of the automaton, in such a way that each triplet $(u, (u, v), v)$ (where u and v are adjacent vertices) has a labeling (q, ℓ, q') (where q and q' are states, and ℓ is a letter) that is a proper transition, and the first and last vertices are labeled with initial and final states respectively. Now to certify that a word is recognized by an automaton, we can label every node with its state in an accepting run, and the verification can be done locally. Finally, Büchi-Elgot-Trakhtenbrot theorem [5, 13, 38] states that MSO properties are exactly the ones that are recognized by a regular automaton, thus we get Theorem 2.2 in the case of directed paths. The automata point of view (without the relation to logics) has been used before to understand the complexity of locally checkable labelings on cycles and paths, see in particular in [7].

Now, a tree automaton is the analogue of a regular automaton for rooted trees. In particular, the transitions specify states for a vertex and its children. Again, there is a nice relation with MSO: MSO logic on trees is exactly the set of languages recognized by tree automata [36]. Therefore, the same labeling-by-states strategy basically works, but there are some technicalities. Indeed, the result of [36] is for rooted trees with bounded degree and with an order on the children of each node, and the properties expressible in MSO in this type of trees are a bit different from the ones in our unrooted, unordered trees with unbounded degrees. But we can get the result by describing a root in the certificates, and using less classical results for other types of tree automata, adapted to our type of trees [3].

Interestingly, the tree automata that capture MSO properties on trees can be described as checking that the multiset of states of the neighbors satisfies some simple inequalities. We discuss in the full version how this provides interesting directions to generalize the

classic and well-understood setting of locally checkable labelings (LCLs) [30].

5 TREEDEPTH CERTIFICATION VIA ANCESTORS LISTS

This section is devoted to the proof of the following theorem.

Theorem 2.4. *We can certify that a graph has treedepth at most t with $O(t \log n)$ bits.*

Let v be a vertex, and w be its parent in the tree, we define an *exit vertex of* v as a vertex u of G_v connected to w. Note that such a vertex must exist, if the model is coherent.

We now describe a certification. In a *yes*-instance, the prover finds a coherent elimination tree of depth at most t, and assigns the labels in the following way.

- Every vertex v is given the list of the identifiers of its ancestors, from its own identifier to the identifier of the root.
- For every vertex v, except the root, the prover describes and certifies a spanning tree of G_v, pointing to the exit vertex of v. (See Subsection 3.3 for the certification of spanning trees.) The vertices of the spanning tree are also given the depth k of v in the elimination tree.

Note that the length of the lists is upper bounded by t, and that every vertex holds a piece of spanning tree certification only for the vertices of its list, therefore the certificates are on $O(t \log n)$ bits. Now, the local verification algorithm is the following. For every vertex v with a list L of length $d+1$, check that:

(1) $d \leq t$, and L starts with the identifier of the vertex, and ends with the same identifier as in the lists of its neighbors in the graph.
(2) The neighbors in G have lists that are suffixes or extensions by prefix of L.
(3) There are d spanning trees described in the certificates.
(4) For every $1 \leq k \leq d$, for the spanning trees associated with depth k:
   - The tree certification is locally correct.
   - The neighbors in the tree have lists with the same $(k+1)$-suffix.
   - If the vertex is the root, then it has a neighbor whose list is the k-suffix of its own list.

It is easy to check that on *yes*-instances the verification goes through. Now, consider an instance where all vertices accept. We shall prove that then we can define a forest, such that the lists of identifiers given to the nodes are indeed the identifiers of the ancestors in this forest. Once this is done, the fact that Steps 1 and 2 accept implies that the forest is a tree of the announced depth, and is a model of the graph. Let us first prove the following claim:

Claim 1. *For every vertex u, with a list L of size at least two, there exists another vertex v in the graph whose list is the same as L but without the first element.*

Consider a vertex u like in Claim 1, at some depth d. If all vertices accept, then this vertex is has a spanning tree corresponding to depth d (by Step 3), where all vertices have the same $(d+1)$-suffix, and the root of this tree has a neighbor whose list is L, without the first identifier, by Step 4. This vertex is the v of the claim.

The claim implies that the whole tree structure is correct. Indeed, if we take the vertex set of G, and add a pointer from every vertex u to its associated vertex v (with the notations of the claim), then the set of pointers must form a forest. In particular, there cannot be cycles, because the size of the list is decremented at each step. Also, if the ancestors are consistent at every node, then they are consistent globally. This finishes the proof of Theorem 2.4.

6 MSO/FO CERTIFICATION IN BOUNDED TREEDEPTH GRAPHS VIA KERNELIZATION

In this section, we prove the following theorem.

Theorem 2.6. *Every FO (and hence MSO) sentence φ can be locally certified with $O(t \log n + f(t, \varphi))$-bit certificates on graphs of treedepth at most t (for some function f).*

The proof is based on a kernelization result: we show that for every integer t and k, for every graph of treedepth t, we can associate a graph, called *a kernel*, such that (1) it satisfies the same FO formulas with quantifier depth at most k, and (2) it has a size that is independent of n (that is, depends only on t and k). The idea is then to locally describe and certify this kernel, and to let the vertices check that the kernel satisfies the formula.

Actually, such a kernel always exists, even without the treedepth assumption. Indeed, since we have a bounded number of formulas of quantifier depth at most k (up to semantic equivalence), we have a bounded number of equivalent classes of graphs for $\simeq_k$. We can associate to each class the smallest graph of the class, whose size is indeed bounded by a function of only k. However, this definition of H_k is not constructive, which makes it impossible to manipulate for certification. We note that a constructive kernelization result already exists for graphs of bounded shrubdepth [24], which implies bounded treedepth. We however cannot use this result either, because we cannot locally certify the kernel of [24]. Hence, we need to design our own certifiable kernel. Incidentally, certifying bounded shrubdepth and the associated model checking problem are interesting open questions.

6.1 Description of the kernel

Let G be a graph of treedepth at most t, and let k be an integer. Let $\mathcal{T}$ be a t-model of G. Let v be a vertex of depth i in the decomposition. We define the *ancestor vector of* v as the $\{0,1\}$-vector of size i, where the j-th coordinate is 1, if and only if, v is connected in G to its ancestor at depth j.

We can now define the *type of a vertex* v as the subtree rooted on v where all the nodes of the subtree are labeled with their ancestor vector. Note that in this construction, the ID of the nodes do not appear, hence several nodes might have the same type while being at completely different places in the graph or the tree.

Let us now define a subgraph of G that we will call the *k-reduced graph*. If a node u has more than k children of the same type, a *valid pruning* operation consists in removing in $\mathcal{T}$ (and in G) the vertices of the subtree rooted at one of these children (including the children). Note that afterwards $\mathcal{T}$ is still a t-model of the resulting graph. Moreover, by doing so, we change the structures of the subtree of u and the subtrees of its ancestors, thus we also update their

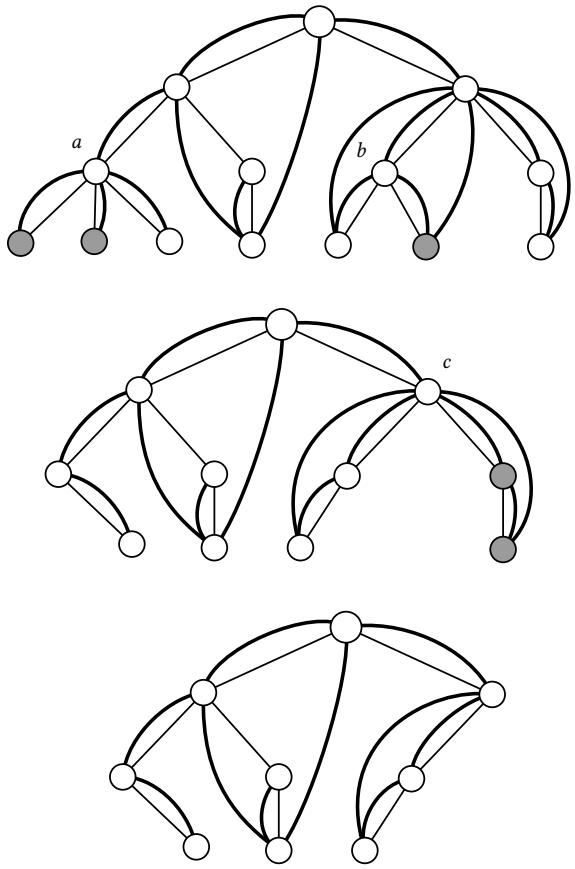

Figure 2: Explanation of the kernelization for $k = 1$, step-by-step. The first picture represents a graph (with bold edges) and a (non-necessarily optimal) t-model of it (thin edges). For the first step, we look at vertices at depth $t-1$, that is, 2 on the example. The vertices a and b have more than k children with the same type (since these are leaves, we only compare the ancestors "lists"). We keep only k copies, and remove the gray vertices. This leads to the second picture. On the second step, we look at vertices at depth $t-2$. Here c has two equivalent subtrees (same shape, same ancestors), thus we remove one. We get to the third picture, which cannot be reduced further, and is our kernel.

types. A *k-reduced graph* H (that is, the kernel for this parameter k) of G is a graph obtained from G by iteratively applying valid pruning operations on a vertex of the largest possible depth in $\mathcal{T}$ while it is possible. A vertex v is *pruned* for a valid pruning sequence if it is the root of a subtree that is pruned in the sequence. Note that there are some vertices of $G \setminus H$ that have been deleted, but that are not pruned.

Let G be a graph, and H be a k-reduced graph of G. The *end type* (with respect to H)[2] of a vertex v of G is: its type in H if it has not been deleted, and the last type it has had otherwise (that is, its type in the graph G' which is the current graph when it was deleted).

[2]One can prove that it actually does not depend on H but we do not need it in our proof.

6.2 Size of the kernel and number of end types

Since we apply pruning operations on a vertex of the largest possible depth, if at some point we remove a vertex of depth i, then we never remove a subtree rooted on a vertex of depth strictly larger than i afterwards. It implies that when a vertex at depth i is deleted, the types of the nodes at depth at least i are their end type. The following lemma, proved in the full version, describes the structure of the end types in the k-reduced graph.

Lemma 6.1. *Let G be a graph and H be a k-reduced graph of G. Let $u \notin H$ and $v \in H$, such that u is a child of v. Then there exists exactly k children of v in H whose end type is the end type of u.*

Observe that the end type of a vertex v depends only on the adjacency of v with its ancestors and on the number of children of v of each possible end type. Combining this with Lemma 6.1, we prove the following statement.

Proposition 6.2. *The number of possible end types of a node at depth d in a k-reduced graph of treedepth at most t is bounded by $f_d(k,t) := 2^d \cdot (k+1)^{f_{d+1}(k,t)}$. It follows that the size of each k-reduced graph only depends on k and t.*

The proof of Proposition 6.2 is in the full version. The idea is to have a bottom-up induction. For the leaves of the tree, the type only depends on the adjacency of the vertex to its ancestors in the tree, therefore there are only 2^t types. Then, for an internal node, as there can be only k children with the same type, the fact that there is a bounded number of children types implies that there is a bounded number of types for this internal vertex.

6.3 Correctness of the kernel

Proposition 6.3. *Let G be a graph of treedepth t, $\mathcal{T}$ be a t-model of G, and G' be a k-reduced graph of G. Then $G \simeq_k G'$ (using the notation of Subsection 3.2).*

Proof. Observe that G' is a subgraph of G, and denote by $\mathcal{T}'$ the restriction of $\mathcal{T}$ to the vertices of G'. If $S \subset V(G)$, we denote by $\mathcal{T}_S$ the subtree of $\mathcal{T}$ induced by the vertices of S and their ancestors. In particular, $\mathcal{T}' = \mathcal{T}_{V(G')}$. Moreover, two rooted trees are said to be *equivalent* if there is an end type-preserving isomorphism between them.

By Theorem 3.3, proving Proposition 6.3 is equivalent to finding a winning strategy for Duplicator in the Ehrenfeucht-Fraissé game on G, G' in k rounds. To this end, we prove that she can play by preserving the following invariant.

Claim 2. *Let $x_1, \dots, x_i$ (resp. $y_1, \dots, y_i$) be the positions played in G (resp. G') at the end of the i-th turn. Then the rooted trees $\mathcal{T}_{\{x_1,\dots,x_i\}}$ and $\mathcal{T}'_{\{y_1,\dots,y_i\}}$ are equivalent.*

The invariant holds for $i = 0$, since the two trees are empty. Assume now that it is true for some $i < k$. We consider the case where Spoiler plays on vertex x_{i+1} in G, the other case being similar (and easier). Consider the shortest path in $\mathcal{T}_{\{x_1,\dots,x_{i+1}\}}$ between x_{i+1} and a vertex of $\mathcal{T}_{\{x_1,\dots,x_i\}}$. We call this path $u_1, \dots, u_p$, with u_1 a node of $\mathcal{T}_{\{x_1,\dots,x_i\}}$ and $u_p = x_{i+1}$. Note that, necessarily, for all $j \in [1, i]$,

u_j is the parent of u_{j+1} in the tree. For $j = 1, \dots, p$, we will find a vertex u'_j in G' such that $\mathcal{T}_{\{x_1,\dots,x_i,u_j\}}$ is equivalent to $\mathcal{T}'_{\{y_1,\dots,y_i,u'_j\}}$ (this implies that u_j and u'_j have the same end type).

For $j = 1$, first observe that $\mathcal{T}_{\{x_1,\dots,x_i,u_1\}} = \mathcal{T}_{\{x_1,\dots,x_i\}}$, because u_1 belongs to $\mathcal{T}_{\{x_1,\dots,x_i\}}$. Then, since $\mathcal{T}_{\{x_1,\dots,x_i\}}$ is equivalent to $\mathcal{T}'_{\{y_1,\dots,y_i\}}$, we can define u'_1 as the copy of u_1 in $\mathcal{T}'_{\{y_1,\dots,y_i\}}$.

Assume now that $u'_1, \dots, u'_j$ are constructed. Let T be the end type of u_{j+1} in G, and r be the number of children of u_j having T as their end type (including u_{j+1}). By construction of G' and u'_j, we know that u'_j has $\min(r, k)$ children with type T in $\mathcal{T}'$. Observe that at most $\min(r-1, i)$ children of u_j of type T in $\mathcal{T}$ can lie in $\mathcal{T}_{\{x_1,\dots,x_i\}}$. Indeed, since u_{j+1} does not belong to $\mathcal{T}_{\{x_1,\dots,x_i\}}$, we get the $r-1$ term, and since $\mathcal{T}_{\{x_1,\dots,x_i\}}$ is made by i vertices and their ancestors, not more than i vertices of $\mathcal{T}_{\{x_1,\dots,x_i\}}$ can have the same parent. Also, using $i < k$, we get $\min(r-1, i) \leqslant \min(r, k) - 1$. Therefore, there exists a child u'_{j+1} of u'_j of type T in $\mathcal{T}' \setminus \mathcal{T}'_{\{y_1,\dots,y_i\}}$.

By taking $y_{i+1} = u'_p$, we finally obtain that $\mathcal{T}_{\{x_1,\dots,x_i,u_p\}} = \mathcal{T}_{\{x_1,\dots,x_{i+1}\}}$ is equivalent to $\mathcal{T}'_{\{y_1,\dots,y_i,u'_p\}} = \mathcal{T}'_{\{y_1,\dots,y_{i+1}\}}$, as required. □

6.4 Certification of the kernel

Proposition 6.4. *Let k be an integer. Let G be a graph of treedepth at most t with a coherent model $\mathcal{T}$. Let H be a k-reduction of G obtained via a valid pruning from $\mathcal{T}$. Then we can certify with certificates of size $O(t \log n + g(k,t))$ that H is a k-reduction of G from $\mathcal{T}$ (for some function g).*

Proof. Let us describe a local certification. On a *yes*-instance, the prover gives to every vertex v the following certificate:

- The $O(t \log n)$-bit certificate of v for the t-model $\mathcal{T}$ of G given in Theorem 2.4.
- A list of d booleans that says, for any ancestor x of v, including v, if x is pruned, i.e. the subtree rooted on x has been pruned at some step.
- For every ancestor w of v including v, the end type of w, coded on $\log(f_i(k,t))$ bits, where i is the depth of w (by Proposition 6.2).

Every node v at depth d thus receives a certificate of size at most $O(t \log n + d + \sum_{i=1}^{d} \log(f_i(k,t)))$. Let us now describe the local verification algorithm, as well as why it is sufficient for checkability.

Recall that the end type of a vertex only depends on its adjacency with its list of ancestors as well as the end types of its children. So first, the node v can check that its adjacency with its list of ancestors is compatible with its end type. Then, it checks that, if one of its children w has been pruned, then it has exactly k children with the type of w that have not been pruned (there is no type T such that more than k children of type T are left after pruning). Note that v has access to all this information since, for every child w, there is a vertex x in the subtree rooted on w adjacent to v, because $\mathcal{T}$ is coherent. Finally, since the end type of v is determined by the end types of its children, v simply has to check that its end type is consistent with the list of end types of its children.

As in the proof of Theorem 2.4, for any child w of v, if the prover has cheated and the type of w has been modified between w and the exit vertex of w, then one node of the path from w to the exit vertex should discover it, which ensures that the certification is correct. □

7 LOWER BOUNDS VIA NON-DETERMINISTIC COMMUNICATION COMPLEXITY

In this section, we will prove our two lower bounds, namely Theorem 2.3 and 2.5. To do so, we will first define a framework for reduction from two-party non-deterministic communication complexity, and then use it for the two proofs.

Such reductions from communication complexity have been used before in local certification in [6, 18, 25]. But in all these works, the reduction was used to establish lower bounds in the polynomial regime (*e.g.* $\Omega(n)$ or $\Omega(n^2)$), whereas our second lower bound (Theorem 2.5) is for the logarithmic regime. For both our lower bound and the lower bounds of [6, 18, 25], the proof is essentially about proving that a set S of vertices have to collectively know the exact structure of a far-away subgraph. The difference is that in previous works, either the subgraph was dense or the set S was small, whereas in our second bound, the subgraph is sparse and the set S is large, which leads to lower bounds for a lower regime. One can naturally wonder if the other $\Omega(\log n)$ lower bounds of the area (in particular for acyclicity) can be obtained by communication complexity instead of the usual cut-and-plug techniques (that is, the combination of indistinguishability and counting arguments).

7.1 Framework for reductions from communication complexity

Non-deterministic communication complexity. Let us describe the non-deterministic communication complexity setting. (This is not the same exact setting that is used in other similar reductions, we discuss the differences at the end of this subsection.) There are two players, Alice and Bob, and a prover. Alice has a string s_A and Bob a string s_B. Both strings have length ℓ. The prover chooses a string s_P of length m, called *certificate*, that is given to Alice and Bob. Alice decides to accept or to reject by only looking at s_A and s_P. Let f_A the function that corresponds to this process. Same for Bob with s_B and f_B, instead of s_A and f_A. We say that a protocol, described by f_A and f_B decides EQUALITY, if:

- For every instance where $s_A = s_B$, there exists s_P such that $f_A(s_A, s_P) = f_B(s_B, s_P) = 1$.
- For every instance where $s_A \neq s_B$ for all strings s_P, $f_A(s_A, s_P) = 0$ or $f_B(s_B, s_P) = 0$.

The following theorem ensures that there is asymptotically no better protocol than to have the full string written in the certificate.

Theorem 7.1 ([1]). *Any non-deterministic communication protocol for EQUALITY for strings of length ℓ requires a certificate of size $\Omega(\ell)$.*

Framework for reductions. Let ℓ be an integer. For any pair of strings (s_A, s_B) of length ℓ, we define a graph $G(s_A, s_B)$.

The set of vertices of $G(s_A, s_B)$ is partitioned into fours sets $V = V_A \cup V_B \cup V_\alpha \cup V_\beta$. In our reductions, the edge set of $G(s_A, s_B)$ will be composed of two parts. One will be independent of s_A and s_B (and will only depends on which graph class we want to obtain a lower bound and ℓ) and a part that will depend on s_A and s_B.

The set of edges independent of s_A, s_B, denoted by E_P, is such that every edge in E_P is in one of the following sets: $V_A \times V_\alpha$, $V_\alpha \times V_\alpha$, $V_\alpha \times V_\beta$, $V_\beta \times V_\beta$ and $V_\beta \times V_B$ (see Figure 3 for an illustration). Let $k = |V_A| = |V_B|$ and $r = |V_\alpha \cup V_\beta|$.

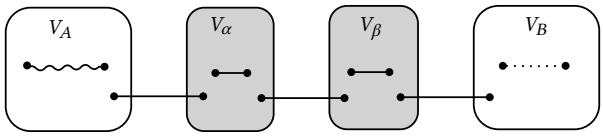

Figure 3: Illustration of the construction of $G(s_A, s_B)$. The straight edges are the five possible types for edges of E_P. The curvy edge corresponds to an edge of Alice, and the dotted edge to an edge of Bob.

Let t_A be an injection from the set of strings of length ℓ to the set of subgraphs of V_A. Let t_B be the analogue for V_B. The graph $G(s_A, s_B)$ is the graph with vertex set V, and edge set $E = t_A(s_A) \cup t_B(s_B) \cup E_P$. Note that, by construction, the vertices of $V_A \cup V_\alpha$ are not adjacent to the verticess of V_B, and the vertices of $V_B \cup V_\beta$ are not adjacent to the vertices of V_A.

This graph is equipped with an identifier assignment, such that the vertices of $V_\alpha \cup V_\beta$ get the identifiers from 1 to r (in an arbitrary order).

Proposition 7.2. *Let $\mathcal{P}$ be a graph property that is satisfied by $G(s_A, s_B)$ if and only if $s_A = s_B$. Then a local certification for $\mathcal{P}$ requires certificates of size $\Omega(\ell/r)$.*

The proof of Proposition 7.2 is deferred to the full version. The idea is that Alice and Bob can use a certification in the following way. First, they build the graph (V, E_P) that corresponds to the length ℓ of their strings. Then Alice adds the edges $t_A(s_A)$ on her copy, and Bob adds the edges of $t_B(s_B)$ on his copy. Finally, they interpret the certificate given by the prover as an assignment of local certificate to the vertices of V_α and V_β. They can now simulate the local verification on their part of the graph, namely the vertices of $V_A \cup V_\alpha$ and $V_B \cup V_\beta$ respectively, and thus decide if the graph has property $\mathcal{P}$ or not, which by assumption is equivalent to solve the EQUALITY problem. Now if the local certification uses certificates that are very small, it implies that the certificate used in the simulation is also small which would contradict Theorem 7.1.

Discussion of the framework. Reduction to two-party non-deterministic complexity has already been used several times in local certification [6, 18, 25], but for the sake of simplicity in the reduction we use a slightly different setting. First, we use a single certificate instead of one for each player. Second, we say that the instance is rejected if at least one player rejects, instead of having both players reject. Finally, we do not use communication between Alice and Bob: they only read the same certificate. It is known that these changes do not change the asymptotic complexity of the problem.

Note that the framework applies to a where the vertices can receive both a global certificate and local certificates as in [21]. Also, by having V_α and V_β of large enough diameter, one can derive bounds for constant-distance view, or even non-constant views (as in [18, 25]).

7.2 Application to fixed-point free automorphism of trees of bounded depth

We will use the framework described in Section 7.1 to prove the following theorem.

Theorem 2.3. *Certifying the trees that have an automorphism without fixed-point requires certificates of size $\tilde{\Omega}(n)$ (where $\tilde{\Omega}$ hides polylogarithmic factors), even if we restrict to trees of bounded depth.*

The same bound (without the logarithmic factors) was proved in [25] for trees of unbounded depth, via a counting argument. Given that we have results on bounded treedepth, it is necessary to have a lower bound on bounded depth trees, to allow fair comparisons between MSO properties and non-MSO properties (*e.g.* isomorphism-like properties).

The proof is deferred to the full version. It is a relatively direct use of the framework: Both V_α and V_β are reduced to a single vertex connected to each other. Then V_A and V_B will be rooted trees whose root is connected to respectively V_α and V_β. The result follows from the fact that the logarithm of the number of trees of depth k is $\tilde{\Omega}(n)$, as soon as $k \geq 3$ [33], which allows having an injection from the set of strings to the set of bounded depth trees.

7.3 Application to treedepth certification

Theorem 2.5. *Certifying that the treedepth of the graph is at most t requires $\Omega(\log n)$ bits, for any $t \geq 5$.*

Proof. We first prove the theorem for $k = 5$, and then explain how to modify the argument for any $k \geq 4$. Again, we will use the framework of Subsection 7.1. Let ℓ, n be two integers such that there is an injection f from the set of strings of length ℓ to the set of matchings between two (labelled) sets of size n. Our construction is illustrated in Figure 4. Each set V_A, V_B, V_α and V_β consists of two sets of n vertices, that we denote with exponents, e.g. V_A^1 and V_A^2. In each of these sets, the vertices are indexed between 1 and n. We also add a vertex u, that is adjacent to all the vertices of V_α. In the construction, it will behave like a vertex of V_α (hence simulated by Alice). The set of edges E_P is the collection of $2n$ disjoint paths on four nodes, of the form $(V_A^j[i], V_\alpha^j[i], V_\beta^j[i], V_B^j[i])$ for every $i \leq n$ and every $j \in \{1, 2\}$. Note that the graph is connected (even without Alice and Bob's private edges), thanks to the vertex u which is complete to V_α and then adjacent to every path.

Let us now describe the part that is private to Alice. Let s_A be the string of length ℓ given to Alice and M_A be the matching $f(s_A)$ between V_A^1 and V_A^2. Bob does the same for its string s_B. We say that the matchings are equal if, for all i, j, $(V_A^1[i], V_A^2[j])$ is in Alice's matching if and only if $(V_B^1[i], V_B^2[j])$ is in Bob's matching.

Lemma 7.3. *If the matchings are equal, the graph has treedepth 5, otherwise it has treedepth at least 6.*

The proof of this result can be found in the full version.

Once again, we are exactly in the situation of Proposition 7.2, and we want to optimize the parameters. The number of matchings on n vertices is $n!$, thus the logarithm of this quantity is of order $n \log n$. Therefore, we can take $\ell \sim n \log n$. As the size of $V_\alpha \cup V_\beta$ is $2n$, by Proposition 7.2 we get a $\Omega(\log n)$ lower bound.

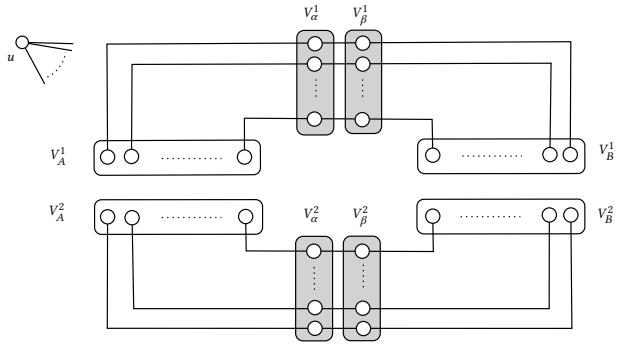

Figure 4: Illustration of the basis of construction of $G(s_A, s_B)$ for bounded treedepth. On the picture, the upper part contains the sets V_A^1, V_α^1, V_β^1, and V_B^1, and the lower part contains V_A^2, V_α^2, V_β^2, and V_B^2. The vertex u is adjacent to all the vertices of V_α.

To extend this proof to the case $k > 5$, it is sufficient to remark that by adding vertices on the edges that have right angles in Figure 4 (*e.g.* the edges of the form $(V_A^1[i], V_\alpha^1[i])$, we can increase the length of the cycles, which changes the threshold between correct instances and incorrect instances, without changing the the rest of the argument. One can actually have a proof for $k = 4$, but without using in the exact framework described above, in particular removing the vertices of V_α and V_β, to get shorter cycles. □

ACKNOWLEDGMENTS

We thank Louis Esperet for fruitful discussions, Rotem Oshman for pointing out references about triangle-free graphs certification, Anca Muscholl for discussions about tree automata. We also thank the reviewers for careful reading and useful advice.

This work is supported by the Agence Nationale de la Recherche (ANR) ANR-18-CE40-0032 (https://oc.g-scop.grenoble-inp.fr/grr/).

REFERENCES

[1] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *27th Annual Symposium on Foundations of Computer Science (FOCS 1986)*, pages 337–347, 1986.
[2] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.
[3] Iovka Boneva and Jean-Marc Talbot. Automata and logics for unranked and unordered trees. In Jürgen Giesl, editor, *Term Rewriting and Applications, 16th International Conference, RTA 2005*, volume 3467, pages 500–515, 2005.
[4] Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Local certification of graph decompositions and applications to minor-free classes. In *25th International Conference on Principles of Distributed Systems, OPODIS 2021*, volume 217 of *LIPIcs*, pages 22:1–22:17, 2021.
[5] J Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6), 1960.
[6] Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theor. Comput. Sci.*, 811:112–124, 2020.
[7] Yi-Jun Chang, Jan Studený, and Jukka Suomela. Distributed graph problems through an automata-theoretic lens. In Tomasz Jurdzinski and Stefan Schmid, editors, *Structural Information and Communication Complexity - 28th International Colloquium, SIROCCO 2021*, volume 12810 of *Lecture Notes in Computer Science*, pages 31–49, 2021.
[8] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.
[9] Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In *33rd International Symposium on Distributed Computing, DISC 2019*, volume 146 of *LIPIcs*, pages 13:1–13:17, 2019.
[10] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *ACM Symposium on Principles of Distributed Computing, PODC '14*, pages 367–376. ACM, 2014.
[11] Michael Elberfeld, Martin Grohe, and Till Tantau. Where first-order and monadic second-order logic coincide. *ACM Trans. Comput. Log.*, 17(4):25:1–25:18, 2016.
[12] Gábor Elek. Planarity is (almost) locally checkable in constant-time. *CoRR*, abs/2006.11869, 2020.
[13] Calvin C Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961.
[14] Louis Esperet and Benjamin Lévêque. Local certification of graphs on surfaces. *Theor. Comput. Sci.*, 2022.
[15] Louis Esperet and Sergey Norin. Testability and local certification of monotone properties in minor-closed classes. *CoRR*, abs/2202.00543, 2022.
[16] Laurent Feuilloley. Introduction to local certification. *Discret. Math. Theor. Comput. Sci.*, 23(3), 2021.
[17] Laurent Feuilloley and Pierre Fraigniaud. Survey of distributed decision. *Bull. EATCS*, 119, 2016.
[18] Laurent Feuilloley, Pierre Fraigniaud, Juho Hirvonen, Ami Paz, and Mor Perry. Redundancy in distributed proofs. *Distributed Comput.*, 34(2):113–132, 2021.
[19] Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Eric Rémila, and Ioan Todinca. Local certification of graphs with bounded genus. *CoRR*, abs/2007.08084, 2020.
[20] Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and Ioan Todinca. Compact distributed certification of planar graphs. *Algorithmica*, 83(7):2215–2244, 2021.
[21] Laurent Feuilloley and Juho Hirvonen. Local verification of global proofs. In *32nd International Symposium on Distributed Computing, DISC 2018*, volume 121 of *LIPIcs*, pages 25:1–25:17, 2018.
[22] Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. A meta-theorem for distributed certification. *CoRR*, abs/2112.03195, 2021.
[23] Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of pure and applied logic*, 130(1-3):3–31, 2004.
[24] Jakub Gajarský and Petr Hlinený. Kernelizing MSO properties of trees of fixed height, and some consequences. *Log. Methods Comput. Sci.*, 11(1), 2015.
[25] Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory Comput.*, 12(1):1–33, 2016.
[26] Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM (JACM)*, 64(3):1–32, 2017.
[27] Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.
[28] Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Comput.*, 22(4):215–233, 2010.
[29] Michael Lampis. Model checking lower bounds for simple graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 673–683. Springer, 2013.
[30] Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM J. Comput.*, 24(6):1259–1277, 1995.
[31] Jaroslav Nešetřil and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006.
[32] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Bounded Height Trees and Tree-Depth*, pages 115–144. Springer, 2012.
[33] Péter Pál Pach, Gabriella Pluhár, András Pongrácz, and Csaba A. Szabó. The number of rooted trees of given depth. *Electron. J. Comb.*, 20(2):P38, 2013.
[34] Fabian Reiter. Distributed graph automata. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015*, pages 192–201, 2015.
[35] Neil Robertson and Paul D. Seymour. Graph minors. i. excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983.
[36] James W. Thatcher and Jesse B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Math. Syst. Theory*, 2(1):57–81, 1968.
[37] Wolfgang Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science. In *TAPSOFT'93: Theory and Practice of Software Development, International Joint Conference CAAP/FASE*, volume 668 of *Lecture Notes in Computer Science*, pages 559–568. Springer, 1993.
[38] Boris Avraamovich Trakhtenbrot. Finite automata and the logic of one-place predicates. *Sibirskii Matematicheskii Zhurnal*, 3(1):103–131, 1962.