Fine-Grained Bounds for Courcelle's Theorem

Daniel Lokshtanov
University of California Santa Barbara
Santa Barbara, USA
daniello@ucsb.edu

Fahad Panolan
University of Leeds
Leeds, United Kingdom
F.panolan@leeds.ac.uk

Saket Saurabh
The Institute of Mathematical Sciences, HBNI
Chennai, India
saket@imsc.res.in

Jie Xue
New York University Shanghai
Shanghai, China
jiexue@nyu.edu

Meirav Zehavi
Ben-Gurion University of the Negev
Beersheba, Israel
meiravze@bgu.ac.il

Abstract

Courcelle's theorem states that there exists an algorithm that takes as input a graph G of treewidth at most t and a MSO formula ϕ, and determines whether G satisfies ϕ in time $f(\phi,t)\cdot n$. It is folklore that the function f contains a tower of exponentials whose height depends as a linear function of the number of quantifier alternations of the input formula ϕ. A classic reduction of Frick and Grohe shows that, assuming the Exponential Time Hypothesis (ETH), the linear growth of the height of the tower is unavoidable. Nevertheless, there is still a huge gap between existing upper and lower bounds – after all, there is quite a difference between a single exponential and a double exponential running time. In addition, this only gives us a very coarse understanding in the time complexity of Courcelle's theorem. In this paper, we prove a fine-grained version of Courcelle's theorem with nearly ETH-tight dependence on the treewidth parameter t and the quantifier structure of ϕ (specifically, the number of first order and second order variables in each quantifier alternation block).

CCS Concepts

• **Theory of computation → Parameterized complexity and exact algorithms**; **Graph algorithms analysis**.

Keywords

Fixed-parameter algorithms, Treewidth, MSO model checking

ACM Reference Format:
Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. 2026. Fine-Grained Bounds for Courcelle's Theorem. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC '26), June 22–26, 2026, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3798129.3800927

1 Introduction

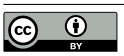
Courcelle's theorem [2, 4] is one of the most celebrated algorithmic meta-theorems, which shows that every graph property expressible in monadic second-order (MSO) logic can be checked in linear time on graphs of bounded treewidth. Formally, it states that given an n-vertex graph G and an MSO formula ϕ, one can test whether G satisfies ϕ in $f(\phi,t)\cdot n$ time for some (computable) function f, where $t=\mathbf{tw}(G)$ denotes the treewidth of G.

Due to the generality of the MSO logic and the importance of treewidth as a structural graph parameter, Courcelle's theorem has brought a profound impact on the theory of parameterized complexity. Specifically, it implies that a large variety of NP-hard graph problems are fixed-parameter tractable (FPT) parameterized by treewidth. In addition, for parameterized graph problems that can be defined using MSO formulas depending on the problem parameter k, Courcelle's theorem results in FPT algorithms parameterized by both treewidth and k.

While the running time of the algorithm in Courcelle's theorem is linear in n (which is optimal), its dependency on ϕ and t is rather intricate and less understood. It was known [4, 14, 19] that the function $f(\phi,t)$ in the bound is not elementary and contains a tower of exponentials whose height depends on ϕ. The seminal work of Frick and Grohe [10] proved that, assuming the ETH, having such a tower of exponentials in the time complexity is unavoidable even when ϕ is a first-order (FO) logic formula and G is a tree. These results, however, only provide us a very coarse understanding in what f should look like in the worst case. Therefore, a more "fine-grained" study for the function $f(\phi,t)$ in Courcelle's theorem turns out to be appealing. While a lot of efforts have been made to understand the optimal time complexity for specific instances of MSO-expressible problems over years [5–7, 13, 21, 22], little work focused on the general MSO testing problem.

Ideally, one wishes to give some concrete function $f(\phi,t)$ that can describe (either exactly or approximately), for *every* ϕ and t, the minimum amount of time required to test property ϕ on graphs of treewidth t. However, this is unfortunately impossible. Indeed, it is not difficult to show the undecidability of the following problem under the assumption P≠NP (we sketch a proof in Section 3): given as input an MSO formula ϕ, decide whether testing ϕ on graphs is polynomial-time solvable or not. This hardness result implies that there is even no way to characterize the formulas ϕ for which the function $f(\phi,t)$ can be made polynomial in t. As such, one cannot hope for any "reasonable" bound on $f(\phi,t)$ that is optimal for every individual ϕ.

Given this frustrating fact, the next best thing one can do towards a fine-grained understanding in Courcelle's theorem is to establish bounds on $f(\phi,t)$ which, while not being optimal for individual



STOC '26, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-2536-4/2026/06
https://doi.org/10.1145/3798129.3800927

formulas, are optimal (or near-optimal) in terms of certain structural parameter $S(\phi)$ of ϕ that can capture the "complexity" of ϕ as precisely as possible; here $S(\phi)$ can be a number or something more complicated. More formally, we want to shoot for the best function $f(S,t)$, which gives the time required to test an MSO property ϕ with $S(\phi) = S$ on graphs of treewidth t.

Motivated by the above discussion, in this paper, we initiate a systematic study on the complexity of Courcelle's theorem, with the goal of understanding its dependency on ϕ and t in a fine-grained way. We prove (almost) matching upper and lower bounds for the complexity of Courcelle's theorem in terms of the *quantifier structure* of ϕ (the detailed results will be discussed in the next section). We expect this work to be a starting point of the long-term research towards thoroughly understanding the time complexity of Courcelle's theorem.

Other related work. We briefly summarize the existing work regarding algorithms and lower bounds on bounded-treewidth graphs. A large body of work [5–7, 13, 21, 22] focused on obtaining tight bounds for classic (MSO-expressible) NP-hard problems parameterized by treewidth. Problems solvable in $2^{O(t)} \cdot n^{O(1)}$ time include Vertex Cover, Dominating Set, q-Coloring, Max Cut, Hamiltonian Cycle, Steiner Tree etc. Problems solvable in $2^{O(t \log t)} \cdot n^{O(1)}$ time include Cycle Packing, Chromatic Number, etc. The time bounds for these problems are all known to be tight. Several works focused on *classes* of problems. Prominent examples arise around Dominating Set, yielding optimal bounds for variants such as r-Domination [3] and (σ, ρ)-Domination [7], among many others [6, 13].

Besides, there has been extensive work on identifying *subclasses* of bounded-treewidth graphs on which every MSO/FO property is decidable with running time bounded by an *elementary* function, in contrast to the non-elementary bounds implied by Courcelle's theorem. For example, Lampis [18] showed that MSO properties can be decided in double exponential time on graphs with bounded vertex cover number, and FO properties can be decided in single exponential time on graphs with bounded max-leaf number. In a recent work, Lampis [20] proved that FO properties can be decided in elementary-function-bounded time on graphs with bounded *path-width* (as opposed to treewidth). Gajarský et al. [12] characterised subgraph-closed graph classes for which the FO-model checking problem is fixed-parameter tractable with an elementary dependency on the formula size. Gajarský and Hliněný [11] showed that in the universe of colored trees of fixed height, any MSO-expressible problem with r quantifiers admits a finite family of kernels whose size is bounded by an elementary function of r and the number of colors.

Kreutzer and Tazari [17] showed that for graph classes with mild closure properties, the presence of graphs with sufficiently large treewidth (already polylogarithmic in n) precludes polynomial-time model checking for MSO_2 formulas. In this sense, bounded treewidth forms the effective boundary for tractable MSO_2 model checking (also see [16]).

1.1 Our Results

In order to discuss our results, we first need to define formally the "quantifier structure" of an MSO formula under consideration. For simplicity of exposition, in this section, we only consider formulas in *prenex normal form* (PNF), which requires all quantifiers to appear at the beginning of the formula. The definition and our results apply to general MSO formulas[1] as well.

Consider an MSO formula ϕ in PNF, which consists of a sequence Q of quantifiers followed by a quantifier-free MSO formula on those quantified variables. We can describe the quantifier structure of ϕ by considering the following three aspects of Q.

- **Quantifier alternations.** The number of quantifier alternations in an MSO formula turns out to be an important parameter to measure its complexity, which influences the height of the tower of exponentials in the running time of Courcelle's theorem [4, 10]. As ϕ is in PNF, this parameter can be simply defined as the smallest integer $d \in \mathbb{N}$ such that one can partition the quantifier sequence Q into d consecutive "blocks" $Q_1, \ldots, Q_d$ each of which contains quantifiers of the same $(\exists, \forall)$-type.
- **Number of quantifiers.** Naturally, the number of quantifiers in the formula also captures how complex it is. Suppose Q is already partitioned into blocks $Q_1, \ldots, Q_d$ according to the quantifier alternations (assume $Q_1, \ldots, Q_d$ are sorted from left to right, or from outermost to innermost). Instead of simply considering the total number of quantifiers in Q, we should consider the number of quantifiers in each individual block Q_i. Note that the roles of $Q_1, \ldots, Q_d$ in ϕ are not exchangeable, and as we will see later in our results, the numbers of quantifiers in $Q_1, \ldots, Q_d$ indeed contribute to the time complexity in different ways.
- **Variable types.** In an MSO formula, there are two types of variables, i.e., vertex variables and set variables, which correspond to a single vertex and a set of vertices in the graph, respectively. An MSO formula with only vertex variables is just an FO formula. In many cases, checking FO graph properties is substantially easier than checking MSO graph properties. For example, testing a fixed FO formula on (general) graphs can always be done in polynomial time, while the problem of testing a fixed MSO formula can be NP-hard (e.g., 3-Coloring). As such, when considering the quantifiers in ϕ, we should distinguish the ones for vertex variables (called *vertex quantifiers*) and the ones for set quantifiers (called *set quantifiers*). For each block Q_i, we use k_i to denote the number of vertex quantifiers in Q_i and use s_i to denote the number of set quantifiers in Q_i. Note that the ordering of the $k_i + s_i$ quantifiers in Q_i does not matter, as all these quantifiers are of the same $(\exists, \forall)$-type.

Based on the above discussion, we can now naturally represent the quantifier structure of the formula ϕ using the sequence $S = ((k_1, s_1), \ldots, (k_d, s_d))$. We call ϕ an *S-MSO formula*. Formally, an MSO formula in PNF is an S-MSO formula if its quantifier sequence can be partitioned into d consecutive blocks $Q_1, \ldots, Q_d$ (sorted from left to right) such that each block Q_i consists of k_i vertex quantifiers and s_i set quantifiers of the same $(\exists, \forall)$-type. (One can further require the quantifiers in adjacent blocks to have different

[1] While every MSO formula can be modified to PNF, such a modification might increase the quantifier rank of the formula and thus makes the formula to have a more complex quantifier structure. As such, we do not make such a modification in our algorithms.

$(\exists, \forall)$-type, but this is not necessary.) The notion of S-MSO formulas can be easily generalized to general MSO formulas. The main focus of this paper is to understand the function $f(\phi, t)$ in Courcelle's theorem in terms of the sequence S representing the quantifier structure of ϕ. We formulate this as the following parameterized problem.

MSO Testing
Input: A graph G with $\mathbf{tw}(G) \leq t$ and an S-MSO formula ϕ.
Parameter: $t \in \mathbb{N}$ and $S = ((k_1, s_1), \ldots, (k_d, s_d))$.
Goal: Decide whether G satisfies ϕ or not

A particularly important special case of MSO Testing is the FO Testing problem, in which ϕ is an FO formula. Since FO formulas are just an MSO formulas without set variables, we can also represent their quantifier structures using sequences. Formally, for $S = (k_1, \ldots, k_d)$, we define an S-FO formula as a S^+-MSO formula where $S^+ = ((k_1, 0), \ldots, (k_d, 0))$.

FO Testing
Input: A graph G with $\mathbf{tw}(G) \leq t$ and an S-FO formula ϕ
Parameter: $t \in \mathbb{N}$ and $S = (k_1, \ldots, k_d)$
Goal: Decide whether G satisfies ϕ or not

Our main results are (almost) matching upper and lower bounds for the complexity of solving MSO/FO Testing (and their variants/extensions). Below we discuss these results in detail.

Upper bounds. To present our algorithmic results, we need to first introduce some notations. We define $\exp^{(0)}(x) = x$ and $\exp^{(i)}(x) = 2^{\exp^{(i-1)}(x)}$ for all integer $i \geq 1$. In other words, $\exp^{(i)}(x)$ is a tower of exponentials of base 2 and height i with x on top of it. The notation $\hat{O}(\cdot)$ denotes the big-O that hides subpolynomial factors, i.e., $\hat{O}(x) = x^{1+o(1)}$. In other words, $\hat{O}(x)$ describes the bound that is *almost* linear in x. For a graph G and a number $t \in \mathbb{N}$, we denote by $T_{\mathsf{td}}(G, t)$ the time required for computing a tree decomposition G with width $t^{O(1)}$, provided that $\mathbf{tw}(G) \leq t$. Our main algorithmic result is the following.

Theorem 1. *There exists an algorithm for MSO Testing that solves an instance (G, t, S, ϕ) with $|V(G)| = n$ and $S = ((k_1, s_1), \ldots, (k_d, s_d))$ in time*

$$T_{\mathsf{td}}(G,t) + f(d) \cdot \Big(\sum_{i=1}^{d} \sum_{j=1}^{i-1} \exp^{(i)}(\hat{O}(s_j k_i)) + \sum_{i=1}^{d} 2^{O(s_d k_i)} + \sum_{i=1}^{d} \sum_{j=1}^{i} \exp^{(i)}(\hat{O}(t_j k_i)) + \sum_{i=1}^{d} \exp^{(i)}(\hat{O}(k_i \log t)) \Big) \cdot (n + |\phi|^{O(1)})$$

for a computable function f, where $t_j = \min\{k_j, t\}$ for $j \in [d]$ and $|\phi|$ is the description size of ϕ.

Using the previous algorithms for computing tree decompositions, e.g. [1, 9, 15], we can take either $T_{\mathsf{td}}(G, t) = t^{O(1)} n \log n$ or $T_{\mathsf{td}}(G, t) = 2^{O(t)} n$. When $d \geq 2$, if we set $T_{\mathsf{td}}(G, t) = 2^{O(t)} n$, then $T_{\mathsf{td}}(G, t)$ is dominated by the other part of the bound of Theorem 1 and thus can be removed from the bound for free. When $d = 1$, it results in an overhead of either $t^{O(1)} n \log n$ or $2^{O(t)} n$. Theorem 1 directly implies the following result for FO Testing.

Corollary 2. *There exists an algorithm for FO Testing that solves an instance (G, t, S, ϕ) with $|V(G)| = n$ and $S = (k_1, \ldots, k_d)$ in time $T_{\mathsf{td}}(G, t) +$*

$$f(d) \cdot \Big(\sum_{i=1}^{d} \sum_{j=1}^{i} \exp^{(i)}(\hat{O}(t_j k_i)) + \sum_{i=1}^{d} \exp^{(i)}(\hat{O}(k_i \log t)) \Big) \cdot (n + |\phi|^{O(1)})$$

for a computable function f, where $t_j = \min\{k_j, t\}$ for $j \in [d]$ and $|\phi|$ is the description size of ϕ.

The bound in Theorem 1 is complicated, and as we will see later, it is essentially the best one can hope for. Before moving to the lower bound part, we briefly discuss the bound in Theorem 1 and how it relies on the various parameters.

First, we consider the parameters $k_1, s_1, \ldots, k_d, s_d$, while assuming the treewidth parameter t is a constant. In this case, the bound in Theorem 1 becomes

$$f(d) \cdot \Big(\sum_{i=1}^{d} \sum_{j=1}^{i-1} \exp^{(i)}(\hat{O}(s_j k_i)) + \sum_{i=1}^{d} 2^{O(s_d k_i)} \Big) \cdot (n + |\phi|^{O(1)}),$$

which is the time complexity of the algorithm in Theorem 1 when applied to MSO Testing on trees (or graphs whose treewidth is a constant). If we further assume that $s_1, \ldots, s_d$ are constant numbers, then the bound simply becomes $f(d) \cdot (\sum_{i=1}^{d} \exp^{(i)}(\hat{O}(k_i))) \cdot (n + |\phi|^{O(1)})$, which is also the time for solving FO Testing on bounded-treewidth graphs. In other words, the dependency on k_i, i.e., the number of vertex quantifiers in the i-th block, forms an exponential tower of height i (rather than d) with k_i on top of it. On the other hand, the dependency on $s_1, \ldots, s_d$, i.e., the numbers of set quantifiers in the d blocks, is much worse. If $k_1, \ldots, k_d$ are constant numbers, then the bound becomes $f(d) \cdot (\sum_{i=1}^{d-1} \exp^{(d)}(\hat{O}(s_i)) + 2^{\hat{O}(s_d)}) \cdot (n + |\phi|^{O(1)})$. That says, all of the parameters $s_1, \ldots, s_d$, except s_d, appear on top of the highest exponential towers, which are of height d. Somewhat counterintuitively, however, the dependency on s_d is single exponential.

Next, we consider the treewidth parameter t, while assuming $k_1, s_1, \ldots, k_d, s_d$ are constant numbers. In this case, the bound in Theorem 1 becomes $f(d) \cdot \exp^{(d)}(\hat{O}(\log t)) \cdot (n + |\phi|^{O(1)})$. In fact, a more careful analysis can give us an improved bound in this case, in which the tower is $\exp^{(d)}(O(\log t))$, i.e., $\exp^{(d-1)}(t^{O(1)})$. Therefore, the dependency on t forms an exponential tower of height $d - 1$. This implies, for example, that for a fixed formula ϕ where the number of quantifier alternations is 1, the algorithm runs in polynomial time in t, and when the number of quantifier alternations is 2 (such as Independent Set and 3-Coloring), the running time is single exponential in t. Table 1 gives an intuitive illustration for the dependency of our algorithm on the parameters, showing the level of the highest exponential tower on top of which each parameter appears (as a polynomial).

Lower bounds. To complement our algorithmic results, we prove ETH-based lower bounds for MSO Testing, which demonstrates that the time complexity in Theorem 1 is already tight, modulo

Table 1: Dependency of the time complexity of our algorithms on various parameters.

Tower Height	Parameters				
d	k_d	$s_1, \ldots, s_{d-1}$			
$d-1$	k_{d-1}		t		
$\vdots$					
2	k_2				
1	k_1	s_d			
0				n	$\lvert\phi\rvert$

the subpolynomial factors hidden in the $\hat{O}(\cdot)$-notation. Specifically, our lower bounds imply that every tower of exponentials in the bound of Theorem 1 is necessary (and the height of tower cannot be decreased). All of our lower bounds hold even for the case where d is a constant and ϕ is in PNF. For convenience, we say a multivariate function $f(x_1, \ldots, x_r)$ is *independent* of the variable x_i if $f(x_1, \ldots, x_r) = g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_r)$ for some function g. Similarly, we can define functions independent of multiple variables.

We first consider the towers $\exp^{(i)}(\hat{O}(s_j k_i))$ and $2^{O(s_d k_i)}$. This part is independent of the treewidth parameter t, and we can show the corresponding lower bounds even for the problem on *trees*. The results are presented in the following two theorems.

Theorem 3. *Let $d, i, j \in \mathbb{N}$ such that $j < i \leq d$. Assuming the ETH, if an algorithm that solves MSO Testing on trees with $S = ((k_1, s_1), \ldots, (k_d, s_d))$ in time $f(k_1, s_1, \ldots, k_d, s_d) \cdot \exp^{(i)}(T(s_j, k_i)) \cdot (n+|\phi|)^{O(1)}$ for a function $f(k_1, s_1, \ldots, k_d, s_d)$ independent of s_j and k_i, then $T(x, y) = \Omega(xy)$.*

Theorem 4. *Let $d, i \in \mathbb{N}$ such that $i \leq d$. Assuming the ETH, if an algorithm solves MSO Testing on trees with $S = ((k_1, s_1), \ldots, (k_d, s_d))$ in time $f(k_1, s_1, \ldots, k_d, s_d) \cdot 2^{T(s_d, k_i)} \cdot (n + |\phi|)^{O(1)}$ for a function $f(k_1, s_1, \ldots, k_d, s_d)$ independent of s_d and k_i, then $T(x, y) = \Omega(xy)$.*

Next, consider the towers $\exp^{(i)}(\hat{O}(t_j k_i))$ and $\exp^{(i)}(\hat{O}(k_i \log t))$, where $t_j = \min\{k_j, t\}$. This part is independent of the parameters $s_1, \ldots, s_d$ describing the numbers of set quantifiers, and we can show the corresponding lower bounds even for FO Testing.

Theorem 5. *Let $d, i, j \in \mathbb{N}$ such that $j \leq i \leq d$. Assuming the ETH, if an algorithm solves FO Testing with $S = (k_1, \ldots, k_d)$ in time $f(k_1, \ldots, k_d, t) \cdot \exp^{(i)}(T(k_j, t, k_i)) \cdot (n + |\phi|)^{O(1)}$ for a function $f(k_1, \ldots, k_d, t)$ that is independent of t, k_j, and k_i, then $T(x, y, z) = \Omega(\min\{x, y\} \cdot z)$.*

Theorem 6. *Let $d, i \in \mathbb{N}$ such that $i \leq d$. Assuming the ETH, if an algorithm that solves FO Testing with $S = (k_1, \ldots, k_d)$ in time $f(k_1, \ldots, k_d, t) \cdot \exp^{(i)}(T(k_i, t)) \cdot (n + |\phi|)^{O(1)}$ for a function $f(k_1, \ldots, k_d, t)$ that is independent of t and k_i, then $T(x, y) = \Omega(x \log y)$.*

Extensions. Just as the original algorithm in Courcelle's theorem, our algorithm in Theorem 1 can also be extended to problems in more general settings.

The first extension is to the MSO_2 logic, in which the variables can represent not only (sets of) vertices but also (sets of) edges. Courcelle's theorem applies to MSO_2 logic as well. Similarly, Theorem 1 can be generalized to MSO_2 logic for free. Indeed, one can easily reduce an MSO_2 Testing instance (G, t, S, ϕ) to an MSO Testing instance (G', t', S, ϕ') with $|V(G')| = O(t|V(G)|)$, $t' \leq t + 1$, and $|\phi'| = |\phi|^{O(1)}$.

The second extension is to the counting and optimization versions of the MSO Testing problem. Consider an MSO formula $\phi(x_1, \ldots, x_k, X_1, \ldots, X_s)$ with free vertex variables $x_1, \ldots, x_k$ and free set variables $X_1, \ldots, X_s$. In the counting problem, our goal is to compute the number of satisfying assignments of ϕ in a graph G. In the optimization variants, we are further given a weight function $w : V(G) \to \mathbb{R}$. Define the weight of an assignment $(v_1, \ldots, v_k, V_1, \ldots, V_s)$ of ϕ to be $\sum_{i=1}^{k} w(v_i) + \sum_{i=1}^{s} \sum_{v \in V_i} w(v)$. Then our goal is to compute the minimum (or maximum) weight satisfying assignment of ϕ.

We formulate a problem, called MSO Scoring, using semi-fields, which simultaneously generalizes the two problems above. Let G be a graph and $\phi = \phi(x_1, \ldots, x_k, X_1, \ldots, X_s)$ be an MSO-formula. Also, let $\mathbb{F}$ be a semi-field[2] and $w : V(G) \to \mathbb{F}$ be a function. Assume we are provided an oracle that can do additions/multiplications and find multiplicative inverses on $\mathbb{F}$ in constant time. For each assignment $\alpha = (v_1, \ldots, v_k, V_1, \ldots, V_s)$ of ϕ where $v_1, \ldots, v_k \in V(G)$ and $V_1, \ldots, V_s \subseteq V(G)$, we write $w(\alpha) = (\prod_{i=1}^{k} w(v_i)) \cdot (\prod_{i=1}^{s} \prod_{v \in V_i} w(v))$. Then the *score* of ϕ on the vertex weighted graph (G, w) is defined as $\mathrm{scr}(G, w, \phi) = \sum_{\alpha \in \mathcal{A}_\phi(G)} w(\alpha)$. Here $\mathcal{A}_\phi(G)$ denotes the set of all satisfying assignments of ϕ in the graph G. For a sequence $S = ((k_1, s_1), \ldots, (k_d, s_d))$, an *$S$-MSO** formula is defined as a $((k_2, s_2), \ldots, (k_d, s_d))$-MSO formula with k_1 free vertex variables and s_1 free set variables. Then MSO Scoring is defined as follows.

MSO Scoring

Input: A graph G with $\mathbf{tw}(G) \leq t$, a function $w : V(G) \to \mathbb{F}$, and an S-MSO* formula ϕ.

Parameter: $t \in \mathbb{N}$ and $S = ((k_1, s_1), \ldots, (k_d, s_d))$

Goal: Compute $\mathrm{scr}(G, w, \phi)$

The intuition behind our definition of S-MSO* formulas is the following: testing an S-MSO formula ϕ on a graph G can be reduced to counting satisfying assignments of an S-MSO* formula ϕ' on G, where ϕ' is obtained from ϕ by removing the first block of quantifiers and replacing the corresponding quantified variables with free variables.

It is easy to see that when $\mathbb{F} = \mathbb{R}$ with the normal addition and multiplication operators and $w(v) = 1$ for all $v \in V(G)$, $\mathrm{scr}(G, w, \phi)$ is just the number of satisfying assignments of ϕ and hence MSO Scoring generalizes the counting problem. Also, when $\mathbb{F} = \mathbb{R} \cup \{-\infty, \infty\}$ with addition operator $\min\{\cdot, \cdot\}$ (resp., $\max\{\cdot, \cdot\}$) and multiplication operator that is the normal + on real numbers, $\mathrm{scr}(G, w, \phi)$ is just the minimum (resp., maximum) weight of an satisfying assignment of ϕ and hence MSO Scoring generalizes

[2] A *semi-field* is the same as a field except that the elements are not required to have additive inverses.

the optimization problems. We have the following theorem, which is a generalization of Theorem 1.

Theorem 7. *There is an algorithm for MSO Scoring that solves an instance (G, w, t, S, ϕ) with $|V(G)| = n$ and $S = ((k_1, s_1), \ldots, (k_d, s_d))$ in time*

$$T_{\mathrm{td}}(G,t) + f(d) \cdot \Big(\sum_{i=1}^{d} \sum_{j=1}^{i-1} \exp^{(i)}(\hat{O}(s_j k_i)) + \sum_{i=1}^{d} 2^{O(s_d k_i)} + \sum_{i=1}^{d} \sum_{j=1}^{i} \exp^{(i)}(\hat{O}(t_j k_i)) + \sum_{i=1}^{d} \exp^{(i)}(\hat{O}(k_i \log t)) \Big) \cdot (n + |\phi|^{O(1)})$$

for a computable function f, where $t_j = \min\{k_j, t\}$ for $j \in [d]$ and $|\phi|$ is the description size of ϕ.

1.2 Our Approaches

We briefly summarize the approaches we use to prove the main theorems. A more detailed presentation is given in Section 2. To prove Theorem 1, the high-level framework of our algorithm is still the typical one: dynamic programming on a tree decomposition of G. One of the main new insights is that when doing DP at a node x of the tree decomposition, we exploit not only the small size of the bag of x, but also the small treewidth of the entire subgraph "below x". Formally, let (T, β) be a small-width tree decomposition of G. For a node $x \in V(T)$, denote by $\gamma(x)$ the union of $\beta(y)$ for all nodes y in the subtree of T rooted at x. In most DP algorithms on tree decompositions (in particular, the previous proofs of Courcelle's theorem), when computing the DP table at some node $x \in V(T)$, the algorithm actually no longer cares about whether the graph $G[\gamma(x)]$ has a small treewidth or not, and the time cost of this single step only depends on the size of $\beta(x)$ rather than the structure of $G[\gamma(x)]$. In contrast, in our proof, we make heavy use of the bounded treewidth of the graphs $G[\gamma(x)]$ in the DP procedure, in order to reduce the size of the DP tables as well as the time for computing them. To this end, we introduce a *combinatorial* invariant of graphs, called *S-signatures*, which characterizes the satisfiability of all S-MSO formulas on a graph. Essentially, we prove the following nice properties of the signatures.

(i) The (description) size of the S-signatures of *bounded-treewidth* graphs is small: it only depends on S and the treewidth parameter t, and satisfies the bound in Theorem 1.
(ii) For each $x \in V(T)$, the S-signature of $G[\gamma(x)]$ can be computed given the S-signatures of $G[\gamma(y)]$ for all children y of x, in time polynomial in the sizes of the signatures. In particular, one can efficiently compute the S-signature of G by applying DP on (T, β).
(iii) Given the S-signature of G, one can test whether G satisfies an S-MSO formula ϕ in time polynomial in the size of the signature and $|\phi|$.

Our algorithm then uses property (ii) to compute the S-signature of G and then uses property (iii) to test whether G satisfies ϕ or not. Finally, property (i) bounds the time complexity of the entire algorithm. Among the three properties, the proof of property (i) is the most interesting, which requires a clever combination of the propeller decomposition technique [8, 23] and various structural properties of the signatures. See Section 2.1 for a more detailed discussion. The S-signature can be viewed as a "succinct" representation of the MSO-type of a graph (with respect to S-MSO formulas). It not only allows us to do MSO-testing efficiently, but also gives an upper bound on the number of different MSO-types, which is at most exponential in the size of S-signatures.

To prove the lower bounds, we build on the basic ideas in the reduction of Frick and Grohe [10], and apply additional tricks to make the lower bound tight and more general. The proof of Frick and Grohe implies (while not stated explicitly in [10]) that for any given $c \in \mathbb{N}$, one can reduce a problem with ETH lower bound $2^{\Omega(n)}$ to an FO Testing instance on trees with an $(k_1, \ldots, k_d)$-FO formula ϕ, where $d = 2c + O(1)$, $k_i = O(1)$ for all $i \in [d-1]$, and $\exp^{(c)}(k_d) = 2^{O(n)}$. A drawback of this result is that it only gives us a lower bound roughly $\exp^{(d/2-O(1))}(\Omega(k_d))$, far away from the lower bound $\exp^{(d)}(\Omega(k_d))$ we want. To achieve the desired bound, we need a much more careful reduction. In particular, we give a more efficient way to encode numbers by trees in the sense that the (in)equality of two numbers can be checked using FO formulas with a much smaller number of quantifier alternations. Furthermore, we construct new gadgets that allow us to obtain lower bounds regarding the numbers $s_1, \ldots, s_d$ of set quantifiers and the treewidth parameter t, which are not considered in [10]. Again, we provide more details in Section 2.2.

2 Technical Overview

In this section, we provide an informal overview for the ideas used to prove our upper bounds and lower bounds. We shall focus on the main insights, while omitting the details and calculations.

2.1 Upper Bounds

As mentioned in the introduction, to prove Theorem 1, we need to introduce the notion of *S-signatures*, which is a combinatorial invariant of graphs that characterizes the satisfiability of S-MSO formulas on a graph. Formally, the *satisfiability* of S-MSO formulas on a graph G can be defined as a function $\mathsf{SAT}_{S,G}$ that maps each S-MSO formula ϕ to True if G satisfies ϕ and to False if G does not satisfy ϕ. If we use $\mathsf{sgn}_S(G)$ to denote the S-signature of G, we want the following condition to hold: for any graphs G and H, $\mathsf{sgn}_S(G) = \mathsf{sgn}_S(H)$ iff $\mathsf{SAT}_{S,G} = \mathsf{SAT}_{S,H}$.

To explain some intuition, let us now consider a sequence $S = ((k_1, s_1), \ldots, (k_d, s_d))$. Every S-MSO formula (in PNF) can be written of the form

$$\phi = \mathrm{Q} x_1 \ldots \mathrm{Q} x_{k_1} \mathrm{Q} X_1 \ldots \mathrm{Q} X_{s_1} \, \phi'(x_1, \ldots, x_{k_1}, X_1, \ldots, X_{s_1}),$$

where $\mathrm{Q} \in \{\exists, \forall\}$ and ϕ' is an S'-MSO formula (with free variables) for $S' = ((k_2, s_2), \ldots, (k_d, s_d))$. This recursive definition allows us to relate the satisfiability of S-MSO formulas to that of S'-MSO formulas as follows. Let $\mathcal{A}_{G,k_1,s_1}$ be the set of sequences $(v_1, \ldots, v_{k_1}, V_1, \ldots, V_{s_1})$ where $v_1, \ldots, v_{k_1} \in V(G)$ and $V_1, \ldots, V_{s_1} \subseteq V(G)$. We view it as the set of assignments to the formulas of the form $\phi'(x_1, \ldots, x_{k_1}, X_1, \ldots, X_{s_1})$ in G. For each $A \in \mathcal{A}_{G,k_1,s_1}$, define $\mathsf{SAT}_{S',G,A}$ as the function that maps each S'-MSO formula $\phi'(x_1, \ldots, x_{k_1}, X_1, \ldots, X_{s_1})$ to the value $\phi'(A) \in \{\mathsf{True}, \mathsf{False}\}$ evaluated in G. It turns out that $\mathsf{SAT}_{S,G} = \mathsf{SAT}_{S,H}$ iff both of the following are true:

- for every $A \in \mathcal{A}_{G,k_1,s_1}$, there exists $B \in \mathcal{A}_{H,k_1,s_1}$ such that $\mathsf{SAT}_{S',G,A} = \mathsf{SAT}_{S',H,B}$,
- for every $B \in \mathcal{A}_{H,k_1,s_1}$, there exists $A \in \mathcal{A}_{G,k_1,s_1}$ such that $\mathsf{SAT}_{S',G,A} = \mathsf{SAT}_{S',H,B}$.

Equivalently, $\mathsf{SAT}_{S,G} = \mathsf{SAT}_{S,H}$ iff $\{\mathsf{SAT}_{S',G,A} : A \in \mathcal{A}_{G,k_1,s_1}\} = \{\mathsf{SAT}_{S',H,B} : B \in \mathcal{A}_{H,k_1,s_1}\}$. Inspired by this nice relation, we obtain a natural idea to define the S-signatures: defining them *inductively* based on the S'-signatures. However, so far this idea does not quite work, since the satisfiability of S'-MSO formulas we use is already different from the original definition – it takes into account free variables and assignments.

In order to make the idea work, we need to introduce a more general class of graphs, called *labeled* and *colored* graphs, which can "encode" assignments to free variables. Let G be a graph. For $p \in \mathbb{N}_0$ (here $\mathbb{N}_0 = \{0, 1, 2, \dots\}$), a *$p$-labeling* on G is a function $\lambda : [p] \to V(G)$. If $\lambda : [p] \to V(G)$ is a p-labeling on G and $\lambda' : [p'] \to V(G)$ is a p'-labeling on G, we define $\lambda \oplus \lambda' : [p + p'] \to V(G)$ as $(\lambda \oplus \lambda')(i) = \lambda(i)$ if $i \le p$ and $(\lambda \oplus \lambda')(i) = \lambda'(i - p)$ if $i > p$, which is a $(p + p')$-labeling on G. For a set P, a *P-coloring* on G is a function $\mu : V(G) \to P$. If $\mu : V(G) \to P$ is a P-coloring on G and $\mu' : V(G) \to P'$ is a P'-coloring on G, then we define a function $\mu \otimes \mu' : V(G) \to P \times P'$ as $(\mu \otimes \mu')(v) = (\mu(v), \mu'(v))$, which is a $(P \times P')$-coloring on G. A *p-labeled and P-colored graph* is a triple (G, λ, μ) where G is a graph, λ is a p-labeling on G, and μ is a P-coloring on G. Two p-labeled and P-colored graphs (G, λ, μ) and (G', λ', μ') are *isomorphic* if there exists an isomorphism $\pi : V(G) \to V(G')$ of G and G' such that $\lambda' = \pi \circ \lambda$ and $\mu = \mu' \circ \pi$. Isomorphic labeled and colored graphs are viewed as the *same* (or in other words, we only care about the isomorphic type of such graphs). In particular, if $\mathcal{G}$ is a set of p-labeled and P-colored graphs, then different elements in $\mathcal{G}$ are always non-isomorphic. If (G, λ, μ) is a p-labeled and P-colored graph and $A \subseteq V(G)$ is a subset, then we can naturally obtain a $|\lambda^{-1}(A)|$-labeled and P-colored graph $(G[A], \lambda_A, \mu_A)$ as follows. The coloring μ_A is simply defined as $\mu_A = \mu_{|A}$. To define λ_A, suppose $\lambda^{-1}(A) = \{x_1, \dots, x_r\}$ where $x_1 < \cdots < x_r$. Then we define $\lambda_A : [|\lambda^{-1}(A)|] \to A$ by setting $\lambda_A(i) = \lambda(x_i)$. We call $(G[A], \lambda_A, \mu_A)$ the *restriction* of (G, λ, μ) to A.

Now we explain how to use labeled and colored graphs to encode assignments. Again, consider a graph G and some assignment $A = (v_1, \dots, v_k, V_1, \dots, V_s) \in \mathcal{A}_{G,k,s}$. Naturally, the part $(v_1, \dots, v_k)$ for vertex variables can be represented as a k-labeling $\lambda : [k] \to V(G)$ where $\lambda(i) = v_i$. Also, the part $(V_1, \dots, V_s)$ for set variables can be represented as a $\{0,1\}^s$-coloring $\mu : V(G) \to \{0,1\}^s$ where $\mu(v) = (\mathbf{1}_{v \in V_1}, \dots, \mathbf{1}_{v \in V_s})$; here $\mathbf{1}_{v \in V_1} = 1$ if $v \in V_1$ and $\mathbf{1}_{v \in V_1} = 0$ if $v \notin V_1$. Therefore, the k-labeled and $\{0,1\}^s$-colored graph (G, λ, μ) encodes the information of A. To understand the intuition of the operators $\oplus$ and $\otimes$, consider a $((k_1, s_1), (k_2, s_2))$-MSO formula. Suppose that the first block of quantifiers choose an assignment $A_1 \in \mathcal{A}_{G,k_1,s_1}$, and we already encode this assignment as above to obtain a k_1-labeled and $\{0,1\}^{s_1}$-colored graph (G, λ_1, μ_1). Followed by this, the second block of quantifiers also choose an assignment $A_2 \in \mathcal{A}_{G,k_2,s_2}$. We want to change the graph (G, λ_1, μ_1) so that it further encodes A_2. Now we construct the k_2-labeling λ_2 and the $\{0,1\}^{s_2}$-coloring μ_2 on G corresponding to A_2. Then we simply take $(G, \lambda_1 \oplus \lambda_2, \mu_1 \otimes \mu_2)$, which is just the desired graph that encodes the information of both A_1 and A_2.

In order to use labelings and colorings to replace the assignments, we also need to enhance the ability of MSO formulas a bit so that they can take into account the labels and colors. Roughly speaking, an *enhanced* MSO formula, which is designed for labeled and colored graphs, has two additional abilities. First, it can refer to vertices with specific labels in the graph. In other words, if $\lambda : [p] \to V(G)$ is the labeling of the graph, then the vertices $\lambda(1), \dots, \lambda(p)$ are viewed as arguments of the formula (and thus the formula can test the equality/adjacency among them and other quantified vertex variables). Second, it can have atomic formulas which test whether a vertex has a specific color (in the set used for coloring the graph). In other words, if $\mu : V(G) \to P$ is the coloring of the graph, then the formula can contain equations of the form $\mu(x) = a$, where x is a vertex variable (possibly a labeled vertex) and $a \in P$. Here we omit the formal definition of such formulas, as this concept is only introduced for intuitively understanding the definition of signatures (which is not needed in our actual proof).

Now we consider the satisfability of enhanced S-MSO formulas on a labeled and colored graph (G, λ, μ), which can be defined as a function $\mathsf{SAT}_{S,(G,\lambda,\mu)}$ that maps each enhanced S-MSO formula ϕ to True or False depending on whether (G, λ, μ) satisfies ϕ or not. This generalized definition allows us to recursively characterize the satisfability of enhanced S-MSO formulas using the idea at the beginning of this section. Suppose $S = ((k, s)) + S'$. Here + denotes the concatenation operator for sequences. Let (G, λ, μ) and (H, γ, τ) be two p-labeled and P-colored graphs. Using the argument before, we can prove that $\mathsf{SAT}_{S,(G,\lambda,\mu)} = \mathsf{SAT}_{S,(H,\gamma,\tau)}$ if and only if

$$\{\mathsf{SAT}_{S',(G,\lambda\oplus\lambda',\mu\otimes\mu')} : \lambda' \in \Lambda_{G,k} \text{ and } \mu' \in U_{G,\{0,1\}^s}\} = \{\mathsf{SAT}_{S',(H,\gamma\oplus\gamma',\tau\otimes\tau')} : \gamma' \in \Lambda_{H,k} \text{ and } \tau' \in U_{H,\{0,1\}^s}\},$$

where $\Lambda_{G,k}$ (resp., $\Lambda_{H,k}$) is the set of all k-labelings on G (resp., H) and $U_{G,\{0,1\}^s}$ (resp., $U_{H,\{0,1\}^s}$) is the set of all $\{0,1\}^s$-coloring on G (resp., H). This nice characterization gives us a natural definition for S-signatures on labeled and colored graphs. Suppose we have already define S'-signatures which characterize the satisfability of enhanced S'-MSO formulas on labeled and colored graphs. Now we simply define the S-signature of a labeled and colored graph (G, λ, μ) as $\mathsf{sgn}_S(G, \lambda, \mu) =$

$$\{\mathsf{sgn}_{S'}(G, \lambda \oplus \lambda', \mu \oplus \mu') : \lambda' \in \Lambda_{G,k} \text{ and } \mu' \in U_{G,\{0,1\}^s}\}.$$

The characterization above guarantees that S-signatures characterize the satisfability of enhanced S-MSO formulas. To complete the definition, we still need to define S-signatures for the base case, i.e., $S = ()$ is the empty sequence. An enhanced $()$-MSO formula is quantifier-free and thus it can only test the equality/adjacency among the labeled vertices and the colors of the labeled vertices. So, on a graph (G, λ, μ), the induced subgraph $(G[\mathrm{Im}(\lambda)], \lambda, \mu_{|\mathrm{Im}(\lambda)})$ characterizes the satisfability of enhanced $()$-MSO formulas; here $\mathrm{Im}(\lambda)$ is the image of λ and we abuse the symbol λ to denote the labeling on $G[\mathrm{Im}(\lambda)]$ while its codomain is $V(G)$. As such, we simply define $\mathsf{sgn}_{()}(G, \lambda, \mu) = (G[\mathrm{Im}(\lambda)], \lambda, \mu_{|\mathrm{Im}(\lambda)})$. Finally, for a graph G, we can view it a labeled and colored graph by equipping it with the dummy labeling $- : \emptyset \to V(G)$ and the dummy coloring $- : V(G) \to \{0\}$ that maps every vertex $v \in V(G)$ to 0,

and define $\mathrm{sgn}_S(G) = \mathrm{sgn}_S(G, -, -)$. One can verify that $\mathrm{sgn}_S(G)$ characterizes the satisfiability of (normal) S-MSO formulas on G.

Let $S = ((k_1, s_1), \ldots, (k_d, s_d))$. By construction, the S-signature of a p-labeled and P-colored graph is a d-layer nested set with $(p + \sum_{k=1}^{d} k_i)$-labeled and $(P \times \prod_{i=1}^{d} \{0,1\}^{s_i})$-colored graphs at the bottommost level. We write $\|\mathrm{sgn}_S(G, \lambda, \mu)\|$ as the *recursive size* of $\mathrm{sgn}_S(G, \lambda, \mu)$, which is defined as follows: if $\mathrm{sgn}_S(G, \lambda, \mu)$ is a set, then $\|\mathrm{sgn}_S(G, \lambda, \mu)\| = \sum_{x \in \mathrm{sgn}_S(G,\lambda,\mu)} \|x\|$; and $\|\mathrm{sgn}_S(G, \lambda, \mu)\| = 1$, otherwise. In fact, $\|\mathrm{sgn}_S(G, \lambda, \mu)\|$ is just the number of graphs at the bottommost level of the nested set $\mathrm{sgn}_S(G, \lambda, \mu)$. The recursive size of an S-signature can be essentially viewed as the description size of the signature (i.e., the number of bits to encode the signature), modulo the description of each labeled and colored graph at the bottommost level which is anyway polynomial in the numbers in S. Later we will sketch a proof that bounds the S-signatures of *bounded-treewidth* graphs, which is a crucial part of our result. Before this, we first briefly discuss how to do MSO Testing using signatures and how to compute the signatures by DP on tree decomposition.

Testing MSO via signatures. In fact, from the construction of S-signatures, it is not difficult to see that given $\mathrm{sgn}_S(G, \lambda, \mu)$ and an (enhanced) S-MSO formula ϕ, one can test whether (G, λ, μ) satisfies ϕ in $(\|\mathrm{sgn}_S(G, \lambda, \mu)\| + |\phi|)^{O(1)}$ time. If $S = ()$, then this trivially holds. Suppose $S = ((k, s)) + S'$ and we already have an algorithm $\mathrm{Test}_{S'}(\mathrm{sgn}_{S'}(H, \gamma, \tau), \phi')$ which returns True or False depending on whether (H, γ, τ) satisfies ϕ or not in $(\|\mathrm{sgn}_{S'}(H, \gamma, \tau)\| + |\phi'|)^{O(1)}$ time. Then the algorithm $\mathrm{Test}_S(\mathrm{sgn}_S(G, \lambda, \mu), \phi)$ essentially works as follows. If the first block of quantifiers in ϕ are $\exists$-quantifiers, then we simply return $\bigvee_{x \in \mathrm{sgn}_S(G,\lambda,\mu)} \mathrm{Test}_{S'}(x, \phi')$, where ϕ' is the part of ϕ after the first block of quantifiers. On the other hand, if the first block of quantifiers in ϕ are $\forall$-quantifiers, then we simply return $\bigwedge_{x \in \mathrm{sgn}_S(G,\lambda,\mu)} \mathrm{Test}_{S'}(x, \phi')$.

Computing signatures by DP on tree decomposition. Let (T, β) be a tree decomposition of G. We want to compute $\mathrm{sgn}_S(G)$ by DP on (T, β). For a node $x \in V(T)$, denote by T_x the subtree of T rooted at x and define $\gamma(x) = \bigcup_{y \in V(T_x)} \beta(y)$. A natural idea is to compute, at each node $x \in V(T)$, the signature $\mathrm{sgn}_S(G[\gamma(x)])$, based on the signatures $\mathrm{sgn}_S(G[\gamma(y)])$ for children y of x. However, this does not directly work. In fact, only having $\mathrm{sgn}_S(G[\gamma(x)])$ is not sufficient for the DP. Instead, we have to view $G[\gamma(x)]$ as a *boundaried* graph with boundary $\beta(x)$, and the S-signature computed for $G[\gamma(x)]$ should take into account the boundary vertices. To this end, we again make use of coloring. We arbitrarily choose an ordering σ of $V(G)$. For each node $x \in V(T)$, define a $[|\beta(x)|]_0$[3]-coloring $\mu_x : \gamma(x) \to [|\beta(x)|]_0$ on $G[\gamma(x)]$ that maps all vertices in $\gamma(x) \backslash \beta(x)$ to 0 and maps the vertices in $\beta(x)$ bijectively to $[|\beta(x)|]$ following the ordering σ. During the DP procedure, at each $x \in V(T)$, instead of computing $\mathrm{sgn}_S(G[\gamma(x)])$, we compute $\mathrm{sgn}_S(G[\gamma(x)], -, \mu_x)$. With the help of the colorings μ_x, the DP works and can be done efficiently, thanks to the following lemma. Recall that a *separation* of a graph G is a pair (A, B) with $A, B \subseteq V(G)$ such that $A \cup B = V(G)$ and there is no edge between $A \backslash B$ and $B \backslash A$ in G.

Lemma 8 (informal). *Let (G, λ, μ) be a p-labeled and P-colored graph and (A, B) be a separation of G. Also, let $\mu' : V(G) \to [|A \cap B|]_0$ map all vertices in $V(G) \backslash (A \cap B)$ to 0 and map $A \cap B$ bijectively to $[|A \cap B|]$. Suppose $(G[A], \lambda_A, \mu_A)$ and $(G[B], \lambda_B, \mu_B)$ are the restrictions of $(G, \lambda, \mu \otimes \mu')$ to A and B, respectively. Then for any S, one can (efficiently) compute $\mathrm{sgn}_S(G, \lambda, \mu)$ by only knowing $\mathrm{sgn}_S(G[A], \lambda_A, \mu_A)$, $\mathrm{sgn}_S(G[B], \lambda_B, \mu_B)$, $\lambda^{-1}(A)$, and $\lambda^{-1}(B)$.*

The above lemma is not only used for the computation of signatures. Below we shall also use it to bound the signature sizes for bounded-treewidth graphs.

Bounding the signature size. We now sketch our proof for bounding the recursive sizes of the signatures. Although our final goal is to bound the recursive sizes $\|\mathrm{sgn}_S(G, \lambda, \mu)\|$ of the signatures, the main step here is to bound the sizes of the signatures *as sets*, i.e., $|\mathrm{sgn}_S(G, \lambda, \mu)|$. Indeed, due to the definition of the recursive size, once we can bound $|\mathrm{sgn}_S(G, \lambda, \mu)|$ for every S and every (G, λ, μ), we can also bound $\|\mathrm{sgn}_S(G, \lambda, \mu)\|$ for every S and every (G, λ, μ).

For simplicity, in this overview, we assume $t = O(1)$ and bound the size of the signatures using only the parameters in the sequence S. For a set $\mathcal{G}$ of p-labeled and P-colored graphs, we define $\Delta_S(\mathcal{G}) = |\{\mathrm{sgn}_S(G, \lambda, \mu) : (G, \lambda, \mu) \in \mathcal{G}\}|$, which is the number of different S-signatures the graphs in $\mathcal{G}$ have. Denote by $\mathcal{G}_{p,P}$ the set of all p-labeled and P-colored graphs of treewidth at most t.

Consider a graph $(G, \lambda, \mu) \in \mathcal{G}_{p,P}$. Let $S = ((k, s)) + S'$ be a sequence of pairs of natural numbers. By construction, the size of $\mathrm{sgn}_S(G, \lambda, \mu)$ is just to equal to the number of different signatures $\mathrm{sgn}_{S'}(G, \lambda \oplus \lambda', \mu \otimes \mu')$ we can obtain by choosing $\lambda' \in \Lambda_{G,k}$ and $\mu' \in U_{G,\{0,1\}^s}$. Note that this number could be way smaller than the trivial bound $|\Lambda_{G,k}| \cdot |U_{G,\{0,1\}^s}|$, because different choices of λ' and μ' may result in graphs $(G, \lambda \oplus \lambda', \mu \otimes \mu')$ with the same S'-signature. To establish a good bound for this number, our plan is to associate with each choice $(\lambda', \mu') \in \Lambda_{G,k} \times U_{G,\{0,1\}^s}$ an "object" $\Gamma(\lambda', \mu')$ that satisfies the following properties:

- **(P1)** if $\Gamma(\lambda'_1, \mu'_1) = \Gamma(\lambda'_2, \mu'_2)$, then $\mathrm{sgn}_{S'}(G, \lambda \oplus \lambda'_1, \mu \otimes \mu'_1) = \mathrm{sgn}_{S'}(G, \lambda \oplus \lambda'_2, \mu \otimes \mu'_2)$,
- **(P2)** one can easily obtain a good upper bound for the number of different $\Gamma(\lambda', \mu')$.

Property **(P1)** guarantees that $|\mathrm{sgn}_S(G, \lambda, \mu)|$ is at most the number of different $\Gamma(\lambda', \mu')$, and then property **(P2)** will allow us to find a bound for the latter. To define the object $\Gamma(\lambda', \mu')$, we need the following decomposition lemma for bounded-treewidth graphs, which is known as propeller decomposition in [23]. We omit its proof in this overview.

Lemma 9 ([8, 23]). *Let G be a graph with $\mathbf{tw}(G) = O(1)$. Then for any $R \subseteq V(G)$ with $|R| \le r$, there exist $V_0, V_1, \ldots, V_{r'} \subseteq V(G)$ where $r' = O(r)$ satisfying the following conditions:*

(i) $V(G) = \bigcup_{i=0}^{r'} V_i$,
(ii) $R \subseteq V_0$ *and* $|V_0| \le O(r)$,
(iii) $N_G(V_i \backslash V_0) \subseteq V_0 \cap V_i$ *and* $|V_0 \cap V_i| = O(1)$ *for all* $i \in [r']$.

Consider a choice $(\lambda', \mu') \in \Lambda_{G,k} \times U_{G,\{0,1\}^s}$. We apply the above lemma on the graph G with $R = \mathrm{Im}(\lambda \oplus \lambda')$ to obtain $V_0, V_1, \ldots, V_{r'} \subseteq V(G)$ satisfying the three conditions. Note that $|R| \le p + k$ and thus $r' = O(p + k)$. Condition (ii) of the lemma implies $\mathrm{Im}(\lambda \oplus \lambda') \subseteq V_0$ and $|V_0| = O(r) = O(p + k)$. Condition (iii) guarantees that there is no edge between $V_i \backslash V_0$ and $V_j \backslash V_0$ for any different $i, j \in [r']$. For convenience, we assume without loss of

[3]Here the notation $[\cdot]_0$ is defined as $[n]_0 = \{0, 1, \ldots, n\}$.

generality that $|V_0 \cap V_1| = \cdots = |V_0 \cap V_{r'}| = z$; we have $z = O(1)$ by condition (iii).

For $i \in [r']_0$, let $(G[V_i], \lambda_i, \mu_i)$ be the restriction of $(G, \lambda \oplus \lambda', \mu \otimes \mu')$ to V_i. Also, for $i \in [r']$, let $\pi_i : [z] \to V_0$ be a function that maps the numbers in $[z]$ bijectively to the vertices in $V_0 \cap V_i$, which is a z-labeling on $G[V_0]$, and let $\tau_i : V_i \to [z]_0$ be the function defined as $\tau_i(v) = \pi_i^{-1}(v)$ for $v \in V_0 \cap V_i$ and $\tau_i(v) = 0$ for $v \in V_i \setminus V_0$, which is a $[z]_0$-coloring on $G[V_i]$. Now the key observation is that $\mathrm{sgn}_{S'}(G, \lambda \oplus \lambda', \mu \otimes \mu')$ is uniquely characterized by the graph $(G[V_0], \lambda_0 \oplus \pi_1 \oplus \cdots \oplus \pi_{r'}, \mu_0)$ and the signatures $\mathrm{sgn}_{S'}(G[V_1], \lambda_1, \mu_1 \otimes \tau_1), \ldots, \mathrm{sgn}_{S'}(G[V_{r'}], \lambda_{r'}, \mu_{r'} \otimes \tau_{r'})$.

Lemma 10. *One can compute* $\mathrm{sgn}_{S'}(G, \lambda \oplus \lambda', \mu \otimes \mu')$, *knowing only* $(G[V_0], \lambda_0 \oplus \pi_1 \oplus \cdots \oplus \pi_{r'}, \mu_0)$ *and* $\mathrm{sgn}_{S'}(G[V_1], \lambda_1, \mu_1 \otimes \tau_1), \ldots, \mathrm{sgn}_{S'}(G[V_{r'}], \lambda_{r'}, \mu_{r'} \otimes \tau_{r'})$.

Proof sketch. Let $(G_i, \lambda_i^*, \mu_i^*)$ be the restriction of $(G, \lambda \oplus \lambda', \mu \otimes \mu')$ to $V_0 \cup (\bigcup_{j=1}^{i} V_j)$. Roughly speaking, the idea for computing $\mathrm{sgn}_{S'}(G, \lambda \oplus \lambda', \mu \otimes \mu')$ is to keep applying Lemma 8 to iteratively compute $\mathrm{sgn}_{S'}(G_i, \lambda_i^*, \mu_i^*)$ for $i = 0, 1, \ldots, r'$. By construction, $(V(G_{i-1}), V_i)$ is a separation of G_i. As such, Lemma 8 allows us to compute $\mathrm{sgn}_{S'}(G_i, \lambda_i^*, \mu_i^*)$ from $\mathrm{sgn}_{S'}(G_{i-1}, \lambda_{i-1}^*, \mu_{i-1}^*)$ and the given signature $\mathrm{sgn}_{S'}(G[V_i], \lambda_i, \mu_i \otimes \tau_i)$, together with some other information that is encoded in the given graph $(G[V_0], \lambda_0 \oplus \pi_1 \oplus \cdots \oplus \pi_{r'}, \mu_0)$. Note that $(G_{r'}, \lambda_{r'}^*, \mu_{r'}^*) = (G, \lambda \oplus \lambda', \mu \otimes \mu')$. So we end up with the desired signature $\mathrm{sgn}_{S'}(G, \lambda \oplus \lambda', \mu \otimes \mu')$.

We remark that the actual proof of this lemma is more technical than our discussion above. In fact, applying Lemma 8 does not enable us to compute $\mathrm{sgn}_{S'}(G_i, \lambda_i^*, \mu_i^*)$ from $\mathrm{sgn}_{S'}(G_{i-1}, \lambda_{i-1}^*, \mu_{i-1}^*)$ and $\mathrm{sgn}_{S'}(G[V_i], \lambda_i, \mu_i \otimes \tau_i)$. Therefore, in the actual proof, we need to compute the S'-signature of a graph more complicated than $(G_i, \lambda_i^*, \mu_i^*)$ in each iteration, in order to make the induction work. For simplicity, we omit the details here. □

Now we simply define $\Gamma(\lambda', \mu')$ to be the sequence

$$((G[V_0], \lambda_0 \oplus \pi_1 \oplus \cdots \oplus \pi_{r'}, \mu_0), \mathrm{sgn}_{S'}(G[V_1], \lambda_1, \mu_1 \otimes \tau_1), \ldots, \mathrm{sgn}_{S'}(G[V_{r'}], \lambda_{r'}, \mu_{r'} \otimes \tau_{r'})).$$

Lemma 10 directly implies property **(P1)** of $\Gamma(\lambda', \mu')$. Next, we consider property **(P2)**, i.e., how to bound the number of different $\Gamma(\lambda', \mu')$. We first observe some basic facts about $\Gamma(\lambda', \mu')$. The first element in $\Gamma(\lambda', \mu')$ is a $(p+k+r'z)$-labeled and $(P \times \{0,1\}^s)$-colored graph of treewidth at most t, which has $O(p+k)$ vertices because $|V_0| = O(p+k)$. For each $i \in [r']$, the graph $(G[V_i], \lambda_i, \mu_i \otimes \tau_i)$ is a $|\mathrm{Im}(\lambda \oplus \lambda') \cap V_i|$-labeled and $(P \times \{0,1\}^s \times [z]_0)$-colored graph of treewidth at most t. Since $\mathrm{Im}(\lambda \oplus \lambda') = R \subseteq V_0$, we have $|\mathrm{Im}(\lambda \oplus \lambda') \cap V_i| \leq |V_0 \cap V_i| = z$. Thus, each of the remaining elements in $\Gamma(\lambda', \mu')$ is the S'-signature of a graph in $\mathcal{G}_{q,P \times \{0,1\}^s \times [z]_0}$ for some $q \in [z]$.

With these observations, we are ready to bound the number of different sequences $\Gamma(\lambda', \mu')$ for $(\lambda', \mu') \in \Lambda_{G,k} \times U_{G,\{0,1\}^s}$. Since $r' = O(p+k)$, $t = O(1)$, and $z = O(1)$, it turns out that the number of $(p+k+r'z)$-labeled and $(P \times \{0,1\}^s)$-colored graphs with $O(p+k)$ vertices and treewidth at most t is $(p+k)^{O(p+k)} \cdot (2^s|P|)^{O(p+k)}$ (up to isomorphism), which bounds the number of possible values for the first element in $\Gamma(\lambda', \mu')$. For each of the remaining elements in $\Gamma(\lambda', \mu')$, the number of possible values is bounded by $\sum_{q=1}^{z} \Delta_{S'}(\mathcal{G}_{q,P \times \{0,1\}^s \times [z]_0})$, as it is the S'-signature of a graph in $\mathcal{G}_{q,P \times \{0,1\}^s \times [z]_0}$ for some number $q \in [z]$. It is easy to see that $\sum_{j=1}^{z} \Delta_{S'}(\mathcal{G}_{j,P \times \{0,1\}^s \times [z]_0}) \leq z \cdot \Delta_{S'}(G_{z,P \times \{0,1\}^s \times [z]_0})$. There can be $(p+k)^{O(p+k)} \cdot (2^s|P|)^{O(p+k)} \cdot (\Delta_{S'}(\mathcal{G}_{z,P \times \{0,1\}^s \times [z]_0}))^{O(p+k)}$ different sequences $\Gamma(\lambda', \mu')$ in total, which implies the bound

$$|\mathrm{sgn}_S(G, \lambda, \mu)| \leq (p+k)^{O(p+k)} (2^s|P|)^{O(p+k)} (\Delta_{S'}(\mathcal{G}_{z,P \times \{0,1\}^s \times [z]_0}))^{O(p+k)} \quad (1)$$

To make use of this inequality, we need to further upper bound $\Delta_{S'}(\mathcal{G}_{z,P \times \{0,1\}^s \times [z]_0})$ on the right-hand side. Interestingly, as we will see, our argument above also gives a *recursive* bound for Δ-values.

Recall our construction of the sequences $\Gamma(\lambda', \mu')$. While we constructed $\Gamma(\lambda', \mu')$ with respect to the graph (G, λ, μ) and the choice $(\lambda', \mu') \in \Lambda_{G,k} \times U_{G,\{0,1\}^s}$, the sequence $\Gamma(\lambda', \mu')$ indeed *only* depends on the $(p+k)$-labeled and $(P \times \{0,1\}^s)$-colored graph $(G, \lambda \oplus \lambda', \mu \otimes \mu')$. In other words, using the same construction, we can associate with every graph in $\mathcal{G}_{p+k,P \times \{0,1\}^s}$ such a Γ-sequence that uniquely determines its S'-signature. Therefore, the bound in Inequality 1 applies to not only $\mathrm{sgn}_S(G, \lambda, \mu)$ but also its superset $K = \{\mathrm{sgn}_{S'}(H, \gamma, \tau) : (H, \gamma, \tau) \in \mathcal{G}_{p+k,P \times \{0,1\}^s}\}$. Note that $\{\mathrm{sgn}_S(G, \lambda, \mu) : (G, \lambda, \mu) \in \mathcal{G}_{p,P}\} \subseteq 2^K$. So we have the following recursive bound

$$\Delta_S(\mathcal{G}_{p,P}) \leq 2^{|K|} \leq 2^{(p+k)^{O(p+k)} \cdot (2^s|P|)^{O(p+k)} \cdot (\Delta_{S'}(\mathcal{G}_{z,P \times \{0,1\}^s \times [z]_0}))^{O(p+k)}} \quad (2)$$

Now we can bound $|\mathrm{sgn}_S(G, \lambda, \mu)|$ by first applying Inequality 1 and then repeatedly applying Inequality 2. For the base case, we can show that $\Delta_{(k,s)}(\mathcal{G}_{p,P}) \leq 2^{(p+k)^{O(p+k)} \cdot |P|^{O(p+k)}}$, which is (surprisingly) independent of s. We omit the proof of the base case and the calculation for working out the bound of $|\mathrm{sgn}_S(G, \lambda, \mu)|$. But we can get some intuition about the bound just by checking Inequalities 1 and 2. Suppose $S = ((k_1, s_1), \ldots, (k_d, s_d))$. When applying Inequality 1, the parameter k_1 appears in the single exponential position, i.e., the top of an exponential tower of height 1. A nice property of this inequality is that the part $\Delta_{S'}(\mathcal{G}_{z,P \times \{0,1\}^s \times [z]_0})$ on the right-hand side, where $S' = ((k_2, s_2), \ldots, (k_d, s_d))$ in this case, is *independent* of k_1, and the number z is a constant (only depending on t). Therefore, while $\Delta_{S'}(\mathcal{G}_{z,P \times \{0,1\}^s \times [z]_0})$ might contain higher exponential towers, the parameter k_1 remains in the single exponential position. When we further expand $\Delta_{S'}(\mathcal{G}_{z,P \times \{0,1\}^s \times [z]_0})$ using Inequality 2, the parameter k_2 appears in the double exponential position, i.e., the top of an exponential tower of height 2. Again, the recursive part we obtain is independent of k_2, and therefore k_2 remains in the double exponential position towards the end. By keep expanding the Δ-part in the bound using Inequality 2, we can see that each k_i appears in the i-th exponential position. In contrast, the parameters $s_1, \ldots, s_{d-1}$ eventually all climb to the top of the exponential tower of height d (due to our lower bounds, this is unavoidable). This is because every time we expand the Δ-part, the set P brings the s-parameters in the previous levels to the next level. The bound we obtain for $|\mathrm{sgn}_S(G, \lambda, \mu)|$ is independent of s_d, because the base case is independent of s_d. When further using it to bound $\|\mathrm{sgn}_S(G, \lambda, \mu)\|$, however, s_d will appear in the single exponential position, as the size of a $((k_d, s_d))$-signature is single exponential in s_d.

This completes the overview for bounding the recursive sizes of the S-signatures when $t = O(1)$. The actual proof is substantially more involved since we need to consider the dependency on t as well. But the overview already covers most of the key insights in our proof.

2.2 Lower Bounds

Proof Theorem 4 is by a simple reduction from 3-CNF SAT. We explain the technical overview of proofs of Theorems 3, 5, and 6. We give reductions from 3-Coloring to prove Theorems 3, 5, and 6, when $i \geq 2$. In 3-Coloring, the objective is to test whether the given graph G has a proper vertex coloring using three colors. Assuming ETH, no algorithm for 3-Coloring runs in time $2^{o(n)}n^{O(1)}$, where n is the number of vertices in the input graph. Our reduction algorithms will take an n vertex graph G as input, and outputs a graph G' and formula ϕ such that $G' \models \phi$ if and only of G is 3-colorable. In addition to the binary edge relation adj, we also use a finite number of unary label predicates in our formulas. We can eliminate these label predicates, but the use of them eases our explanation. For a label predicate L, we write $\exists x \in \mathsf{L}\ \psi$ to denote $\exists x(\mathsf{L}(x) \wedge \psi)$ and $\forall x \in \mathsf{L}\ \psi$ to denote $\forall x(\mathsf{L}(x) \implies \psi)$. For all the reductions the initial part of the construction of G' is a tree (let us call it base tree). So, first we explain the construction of the base tree. Along with the base tree, we define a formula which is an FO formula except that it contains a function id which returns a non-negative integer. So, let us call such a formula as FO+id formula. Then, for different values of i in Theorems 3, 5, and 6, we explain how to replace id with a valid FO/MSO subformula.

Construction of base tree. Let G be an instance of 3-Coloring, with vertex set $V(G) = \{1, 2, \ldots, n\}$ and edge set $E(G) = \{e_1, \ldots, e_m\}$. Let α be a constant selected based on the runtime of the S-FO/MSO Testing algorithm, under the assumption—made for contradiction—that our lower bound results do not hold. We partition $V(G)$ into α groups $V_1, \ldots, V_\alpha$ such that for each $i \in [\alpha]$, $|V_i| \leq \lceil \frac{n}{\alpha} \rceil$. Let $\ell = 3^{\lceil \frac{n}{\alpha} \rceil}$. For each $i \in [\alpha]$, there are at most ℓ proper 3-colorings of $G[V_i]$. Let us call these 3-colorings $c_{i,1}, \ldots c_{i,\ell_i}$.

Now we construct a tree T_1 rooted at a node rt as follows. The root rt has $\alpha + m$ children and we name them $U_1, \ldots, U_\alpha$ and $f_1, \ldots, f_m$. That is, each node U_i corresponds to the vertex subset V_i of G and each node f_i corresponds to the edge e_i of G. Each node f_i has two children corresponding to the endpoints of e_i. Let us name these nodes with $f_{i,a}$ and $f_{i,b}$, where a and b are the endpoints of e_i. Now, we explain the children of each U_i. Recall that $\{c_{i,1}, \ldots, c_{i,\ell_i}\}$ is the set of all proper 3-colorings of $G[V_i]$. The node U_i has ℓ_i children, and they are named $C_{i,1}, \ldots, C_{i,\ell_i}$. Each $C_{i,j}$ has $|V_i|$ children, and each of them corresponds to a vertex in V_i. That is, each $a \in V_i$, $C_{i,j}$ has a child node named $v_{i,j,a}$. Now, each $v_{i,j,a}$ has two children $id_{i,j,a}$ and $c_{i,j,a}$. See Figure 1 for an illustration.

In the formal proof, we define nine label predicates. However, for the purpose of this overview, we present only the following three. For each $q \in [3]$,

$$\mathsf{Q}_q = \{c_{i,j,a} \ : \ a \text{ is colored with } q \text{ in the proper coloring } c_{i,j} \text{ of } G[V_i]\}.$$

Now, we define the function id on nodes corresponding to the vertices in G as follows.

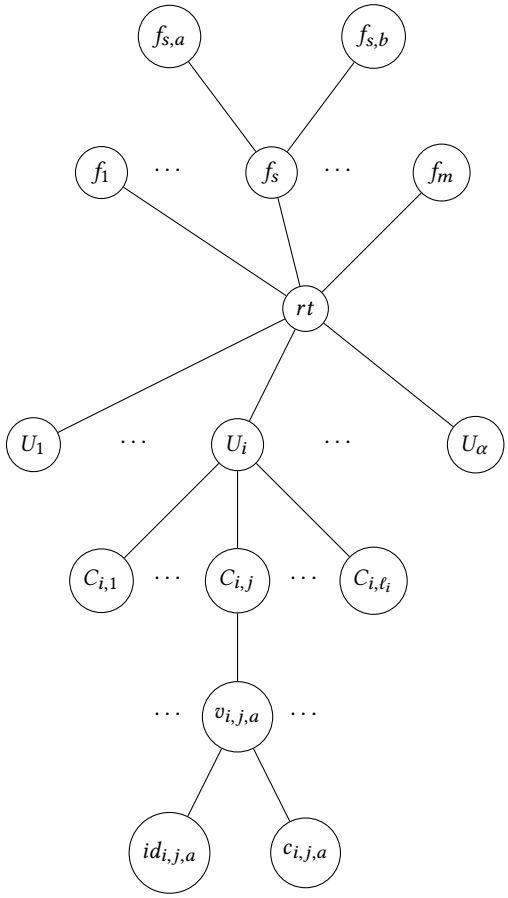


Figure 1: Illustration of construction of base tree T_1

$$\mathrm{id}(x) = \begin{cases} b & \text{if} \quad x = f_{s,b} \\ a & \text{if} \quad x = id_{i,j,a} \end{cases}$$

Now we define an FO+id formula. We want to encode the statement that "there exist nodes $C_{1,j_1}, C_{2,j_2}, \ldots, C_{\alpha,j_\alpha}$ that correspond to proper 3-colorings of $G[V_1], \ldots G[V_\alpha]$, respectively, such that for any node f_s, the endpoints of the edge corresponding to f_s should get different colors according to the selected proper 3-colorings of $G[V_1], \ldots G[V_\alpha]$". This can be encoded as follows, using variable names that match the node names for clarity.

$$\begin{aligned} \psi \quad \equiv \quad & \exists C_{1,j_1}\ \exists C_{2,j_2}\ \ldots\ \exists C_{\alpha,j_\alpha} \\ & \forall f_s\ \forall f_{s,a}\ \forall f_{s,b}\ \forall v_{i,j_i,a}\ \forall v_{i',j_{i'},b} \\ & \forall id_{i,j_i,a} \forall id_{i',j_{i'},b}\ \forall c_{i,j_i,a}, c_{i',j_{i'},b} \in \mathsf{Q}_1 \cup \mathsf{Q}_2 \cup \mathsf{Q}_3 \\ & \psi_{\mathsf{evalid}} \wedge (\psi_{\mathsf{uvalid}} \Rightarrow (\psi_{\mathsf{id}} \Rightarrow \psi_{\mathsf{color}})) \\ \equiv \quad & \exists C_{1,j_1}\ \exists C_{2,j_2}\ \ldots\ \exists C_{\alpha,j_\alpha} \\ & \forall f_s\ \forall f_{s,a}\ \forall f_{s,b}\ \forall v_{i,j_i,a}\ \forall v_{i',j_{i'},b} \\ & \forall id_{i,j_i,a} \forall id_{i',j_{i'},b}\ \forall c_{i,j_i,a}, c_{i',j_{i'},b} \in \mathsf{Q}_1 \cup \mathsf{Q}_2 \cup \mathsf{Q}_3 \\ & \psi_{\mathsf{evalid}} \wedge (\neg\psi_{\mathsf{uvalid}} \vee \neg\psi_{\mathsf{id}} \vee \psi_{\mathsf{color}}) \end{aligned} \tag{3}$$

We explain the meaning of each subformulas in ψ below.

- ψ_{evalid} ensures that $C_{1,j_1}, C_{2,j_2}, \ldots, C_{\alpha,j_\alpha}$ are nodes in T_1 that correspond to proper 3-colorings of $G[V_1], \ldots G[V_\alpha]$, respectively.
- ψ_{uvalid} ensures that all the variables with universal quantifiers are selected appropriately. For example, f_s corresponds to an edge in G, $f_{s,a}$ and $f_{s,b}$ are children of f_s. Also, $v_{i,j_i,a}$ and $v_{i,j_{i'},b}$ are nodes corresponding to vertices in G and each connected to a vertex in $\{C_{1,j_1}, C_{2,j_2}, \ldots, C_{\alpha,j_\alpha}\}$. Moreover, $id_{i,j_i,a}$ and $c_{i,j_i,a}$ are children of $v_{i,j_i,a}$ and $id_{i',j_{i'},b}$ and $c_{i',j_{i'},b}$ are children of $v_{i',j_{i'},b}$
- ψ_{id} is true if and only if $\mathsf{id}(f_{s,a}) = \mathsf{id}(id_{i,j_i,a})$ and $\mathsf{id}(f_{s,b}) = \mathsf{id}(id_{i',j_{i'},b})$.
- ψ_{color} is true if and only if $c_{i,j_i,a}$ and $c_{i,j_{i'},b}$ are different colors.

One can prove that $T_1 \models \psi$ if and only of G is 3-colorable. But, ψ is a $(\alpha, 9)$-FO+id formula. We design various methods to formulate ψ_{id}.

Encoding ids. The formula ψ constructed above is an $(\alpha, 9)$-FO+id formula that contains a subformula $\neg\psi_{\mathsf{id}}$. Here,

$$\begin{aligned}\neg\psi_{\mathsf{id}} &\equiv \neg(\mathsf{id}(id_{i,j_i,a}) = \mathsf{id}(f_{s,a}) \wedge \mathsf{id}(id_{i',j_{i'},b}) = \mathsf{id}(f_{s,b}))\\ &\equiv \mathsf{id}(z_1) \neq \mathsf{id}(y_1) \vee \mathsf{id}(z_2) \neq \mathsf{id}(y_2),\end{aligned}$$

where we substituted $z_1 = id_{i,j_i,a}, y_1 = f_{s,a}, z_2 = id_{i',j_{i'},b}$, and $y_2 = f_{s,b}$, for convenience. We encode (in)equality of ids using FO/MSO formulas leading to different cases of lower bound results

$\log n$-length FO identifier test. Notice that, since $V(G) = [n]$, $\mathsf{id}(x)$ for any node $x \in V(T_1)$, belongs to $[n]$ (if $\mathsf{id}(x)$ is defined on x). We need an FO/MSO formula to represent $\mathsf{id}(z) = \mathsf{id}(y)$ for two variables z and y. Towards that we create two label predicates $\mathbf{1}$ and $\mathbf{0}$. Let k be the smallest integer such that $k \geq \log n$. For any node x in T_1 such that $\mathsf{id}(x)$ is defined, we do the following. Let $b_1, \ldots, b_k$ be the binary representation of the number $\mathsf{id}(x)$. Let π_x be a path $a_1, \ldots, a_k, x$ on $k+1$ vertices. Now, we replace node x with path π_x. For each $i \in [k]$, $a_i \in \mathbf{0}$ if $b_i = 0$ and $a_i \in \mathbf{1}$ otherwise. The tree constructed as explained above is T_2. Now, for two nodes z and y in T, $\mathsf{id}(z) = \mathsf{id}(y)$ can be encoded as

$$\exists a_1, a'_1 \in \mathbf{O} \cup \mathbf{1} \ldots \exists a_k, a'_k \in \mathbf{O} \cup \mathbf{1} \left(\bigwedge_{i \in [k]} (a_i \in \mathbf{0} \Leftrightarrow a'_i \in \mathbf{0}) \right) \wedge \mathsf{path}(a_1, \ldots, a_k, z) \wedge \mathsf{path}(a'_1, \ldots, a'_k, y) \quad (4)$$

where, $\mathsf{path}(w_1, \ldots, w_q) \equiv (\bigwedge_{i \in [q-1]} \mathsf{adj}(w_i, w_{i+1}))$. Notice that the number of quantifiers in the above formula is $2 \log n$. Now, by substituting (4) in ψ we get an $(\alpha, 9 + 2\log n)$-FO formula. Thus, any algorithm of running time $\exp^{(2)}(o(k_2))$ for testing $T_2 \models \psi$, where ψ is an $(O(1), O(\log n))$-FO formula leads to a $2^{o(n)}$ time algorithm for 3-Coloring, a contradiction to ETH. This is a proof overview for Theorem 3 for $i = 2$ and $s_j = 0$ for all j.

Next we explain how to use set variables to reduce the length of the path k above as follows. Let k and s be two positive integers such that $k \cdot s \geq \log n$. Now let $b_1, \ldots, b_k$ be the base 2^s representation of the number $\mathsf{id}(x)$. As before, we have a path $\pi_x = a_1, \ldots, a_k, x$ of length $k+1$ that replaces the node x. Then, define $\mathsf{id}(a_{j'}) = b_{j'}$ for all j'.

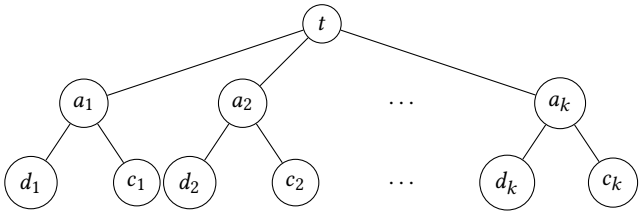


Figure 2: Subtree to replace the node t

Since $2^{k \cdot s} \geq n$, each number in $[n]$ can be uniquely represented as above. Now, we use s set variables $W_1, \ldots, W_s$ to encode ids. Let $b''_1, \ldots, b''_s$ be the binary representation of $\mathsf{id}(a_r)$ and $b'_1, \ldots, b'_s$ be the binary representation of $\mathsf{id}(a'_r)$. Suppose we are able to force the set variables in such a way that for all $j' \in [s]$, $a_r \in W_{j'}$ if and only if $b''_{j'} = 1$ and $a'_r \in W_{j'}$ if and only if $b'_j = 1$. Under that condition, $\mathsf{id}(a_r) = \mathsf{id}(a'_r)$ can be encoded as

$$\bigwedge_{j' \in [s]} (a_r \in W_{j'}) \Leftrightarrow (a'_r \in W_{j'})$$

To satisfy the additional condition mentioned above, we create a formula $\psi_{\mathsf{set}}(W_1, \ldots W_s)$ such that $\psi_{\mathsf{set}}(W_1, \ldots W_s)$ is true if and only if the following is true. For any node a, let $b_1, \ldots, b_s$ be the binary representation of $\mathsf{id}(a)$. Then, the formula $\psi_{\mathsf{set}}(W_1, \ldots W_s)$ is true if and only if for any vertex a and $j' \in [s]$, $a \in W_{j'}$ only when $b_{j'} = 1$. This formula is a $O(1)$-MSO formula with the only set variables are the free variables $W_1, \ldots, W_s$, and all the vertex variables are quantified with universal quantifiers. Then, $\mathsf{id}(z) = \mathsf{id}(y)$ can be encoded as

$$\exists W_1, \ldots, \exists W_s, \exists a_1, a'_1 \ldots \exists a_k, a'_k \left(\bigwedge_{r \in [k]} \bigwedge_{j' \in [s]} (a_r \in W_{j'}) \Leftrightarrow (a'_r \in W_{j'}) \right) \wedge \psi_{\mathsf{set}}(W_1, \ldots W_s) \wedge \mathsf{path}(a_1, \ldots, a_k, z) \wedge \mathsf{path}(a'_1, \ldots, a'_k, y) \quad (5)$$

The new id test leads to Theorem 3 for $i = 2$.

$\log\log n$-length FO identifier test. Recall that we need an FO/MSO formula to represent $\mathsf{id}(z) = \mathsf{id}(y)$ for two variables z and y where $\mathsf{id}(z), \mathsf{id}(y) \in [n]$. Let k and k' be the smallest integers such that $k \geq \log n$ and $k' \geq \log\log n$. For any node $t \in V(T_1)$ such that $\mathsf{id}(t)$ is defined, let $b_1 \ldots b_k$ be the binary representation of the number $\mathsf{id}(t)$. Now we replace t with a subtree as shown in Figure 2. Here, each $a_{j'}$ represents $b_{j'}$ in the following way. We set $\mathsf{id}(d_{j'}) = j'$, $c_{j'} \in \mathbf{O}$ if $b_{j'} = 0$ and $c_{j'} \in \mathbf{1}$ if $b_{j'} = 1$. Here, $\mathbf{O}$ and $\mathbf{1}$ are label predicates to represent whether the bits corresponding to the vertices is 0 and 1, respectively. In other words, $\mathsf{id}(d_{j'})$ represents the position of $b_{j'}$ in the binary representation $b_1 \ldots b_k$ and $c_{j'}$ denotes the value of the bit $b_{j'}$. The crucial observation is that for each $j' \in [k]$, $\mathsf{id}(d_{j'})$ is a positive integer less than or equal to $\lceil \log n \rceil$.

Informally, for two nodes z and y in T_1, $\mathsf{id}(z) = \mathsf{id}(y)$ is true if and only if for any child a of z and any child a' of y if the *id of the left child of a* and the *id of the left child of a'* are equal, then the *corresponding bits (encoded in the right child of a and a')* are same. This can be encoded as follows.

$$\begin{aligned}\mathsf{id}(z) = \mathsf{id}(y) \quad &\equiv \quad \forall a, a', \forall d, d' \forall c, c' \;\; \phi_{\mathsf{valid}} \Rightarrow (\phi_{\mathsf{id}} \Rightarrow \phi_{\mathsf{bit}}) \\ &\equiv \quad \forall a, a', \forall d, d' \forall c, c' \;\; (\neg\phi_{\mathsf{valid}} \vee \neg\phi_{\mathsf{id}} \vee \phi_{\mathsf{bit}}) \quad (6)\end{aligned}$$

Here, ϕ_{valid}, ϕ_{id} and ϕ_{bit} are defined below.

$$\begin{aligned}\phi_{\mathsf{id}} \quad &\equiv \quad \mathsf{id}(d) = \mathsf{id}(d') \\ \phi_{\mathsf{bit}} \quad &\equiv \quad c \in \mathbf{0} \Leftrightarrow c' \in \mathbf{0} \\ \phi_{\mathsf{valid}} \quad &\equiv \quad \mathsf{adj}(a, z) \wedge \mathsf{adj}(a', y) \wedge \mathsf{adj}(d, a) \\ &\quad\;\; \wedge \mathsf{adj}(c, a) \wedge \mathsf{adj}(d', a') \wedge \mathsf{adj}(c', a')\end{aligned}$$

The formula ϕ_{valid} is true if and only if a and a' are children of z and y, respectively, d and c are children of a, and d' and c' are children of a'. The formula ϕ_{bit} is true if and only if they both *encode* the same bit. Clearly, the formula in (6) is not an FO formula, but an FO+id formula with the value of $\mathsf{id}(d)$ and $\mathsf{id}(d')$ are positive integers less than or equal to $\lceil \log n \rceil$. So, we apply the $\log n'$-FO identifier test where $n' = \log n$ and get an $(O(\log\log n))$-FO formula to represent $\mathsf{id}(d) = \mathsf{id}(d')$ where all the quantifiers are existential. By substituting $(O(\log\log n))$-FO formula for $\mathsf{id}(d) = \mathsf{id}(d')$ in (6) we get a $(O(\log\log n))$-FO formula where all quantifiers are universal, because of the negation symbol before ϕ_{id}. Now, if we use this formula to test equality of two identifiers in ψ, we get Theorem 3 for $i = 3$.

We would like to mention that if we apply the same strategy recursively, we can prove Theorem 3 for $i > 3$. In other words, we design a $(k_1, \ldots, k_{i-1})$-FO+id formula, where $\mathsf{id}(x) \leq \log^{(i-2)} n$ and $k_r = O(1)$ for all $r \in [i-1]$. Finally, we apply the $(\log\log n')$-FO/MSO identifier test to get a required formula as the output of the reduction algorithm, where $n' \leq \log^{(i-2)} n$. For the case of $i = 1$, we give a simple reduction from 3-CNF SAT.

Proof overview of Theorems 5 and 6. Let us discuss the case when $i = 2$. For the case when $i > 2$, the approach is similar to the case of Theorem 3 along with the ideas used below for $i = 2$ here. First, we construct the base tree T_1 and an $(\alpha, 9)$-FO+id formula ψ as mentioned in (3). Recall the formula (5) created for id test. Here, we have s set variables $W_1, \ldots, W_s$ and for any node a_r its id is denoted using inclusion of it in the sets $W_1, \ldots, W_s$. Here, notice that $\mathsf{id}(a_r) \in [2^s - 1]$ and $2^{s \cdot k} \geq n$. In Theorem 5, instead of using set variables, we add s new nodes $\{w_1, \ldots, w_s\}$. Recall the role of set variables. For two node a_r and a'_r $\mathsf{id}(a_r) = \mathsf{id}(a'_r)$ is encoded by

$$\bigwedge_{i' \in [s]} (a_r \in W_{i'}) \Leftrightarrow (a'_r \in W_{i'}).$$

Now, to get rid of set variables, we add edges between $\{w_1, \ldots, w_s\}$ and nodes in T for which id is defined as follows. Let x be a node and $b_1, \ldots, b_s$ be the binary representation of $\mathsf{id}(x)$. Then, x is adjacent to t_r if and only if $b_r = 1$. Clearly, the treewidth of the new graph is s. Then, we can replace (5) with the following formula.

$$\exists a_1, a'_1 \ldots \exists a_k, a'_k \left(\bigwedge_{r \in [k]} \bigwedge_{i' \in [s]} (\mathsf{adj}(a_r, w_{i'})) \Leftrightarrow (\mathsf{adj}(a'_r, w_{i'}) \right) \wedge \mathsf{path}(a_1, \ldots, a_k, z) \wedge \mathsf{path}(a'_1, \ldots, a'_k, y) \quad (7)$$

Now, by substituting (7) in ψ, we get Theorem 5 for $i = 2$. Next, we explain the idea for Theorems 6. In this case we add $t = 2^s$ vertices to get rid of set variables. Let $w_0, \ldots, w_{t-1}$ be the new vertices added. Then, a node x is adjacent to w_r if and only if $\mathsf{id}(x) = r$. Clearly, the treewidth of the new graph is at most t. Then, we can replace (5) with the following formula.

$$\exists a_1, a'_1 \ldots \exists a_k, a'_k \left(\bigwedge_{r \in [k]} \bigwedge_{i' \in [t]} (\mathsf{adj}(a_r, w_{i'})) \Leftrightarrow (\mathsf{adj}(a'_r, w_{i'}) \right) \wedge \mathsf{path}(a_1, \ldots, a_k, z) \wedge \mathsf{path}(a'_1, \ldots, a'_k, y) \quad (8)$$

Now, by substituting (8) in ψ, we get Theorem 6 for $i = 2$.

3 Undecidability of Polynomial Time Testable MSO Properties

Consider the following problem: given an MSO formula ϕ, decide whether there exists a graph satisfying ϕ or not. This problem, which we call MSO Realization, is known to be undecidable [24]. Next, we consider our problem: given an MSO formula ϕ, decide whether testing ϕ on graphs is polynomial-time solvable or NP-hard (assuming P≠NP). We call this problem PTime MSO Testability. We show the undecidability of PTime MSO Testability by reducing MSO Realization to it as follows. Let ϕ be an instance of MSO Realization. We construct two MSO formulas ϕ_1 and ϕ_2, where ϕ_1 expresses that "G is 3-colorable and there exists $A \subseteq V(G)$ such that $G[A]$ is connected and satisfies ϕ" and ϕ_2 expresses that "$V(G)$ can be partitioned into A and B such that $G[A]$ is connected and satisfies ϕ, and $G[B]$ is 3-colorable".

Assume there exists an algorithm $\mathcal{A}$ that solves PTime MSO Testability, and we run it on ϕ_1 and ϕ_2. If $\mathcal{A}$ returns P (i.e., polynomial-time solvable) for both ϕ_1 and ϕ_2, we conclude that there does not exist a graph satisfying ϕ; otherwise, we conclude that there exists a graph satisfying ϕ. To see our conclusion is correct, consider three cases. First, assume there does not exist a connected graph satisfying ϕ. In this case, no graph can satisfy ϕ_1 or ϕ_2, and therefore, both ϕ_1 and ϕ_2 can be tested on graphs trivially in polynomial time by simply returning No. Thus, $\mathcal{A}$ returns Yes for both ϕ_1 and ϕ_2, and our conclusion is correct. Second, assume there exists a connected graph G_0 satisfying ϕ which is 3-colorable. In this case, testing ϕ_1 is NP-hard. Indeed, one can reduce an instance G of 3-Coloring to the task of testing ϕ_1 on the graph that is the disjoint union of G and G_0. So $\mathcal{A}$ returns NP for ϕ_1, and our conclusion is correct. Finally, assume there does not exist a connected graph satisfying ϕ which is 3-colorable, but there exists a connected graph G_0 satisfying ϕ that is not 3-colorable. In this case, testing ϕ_2 is NP-hard. Again, one can reduce an instance G of 3-Coloring to the task of testing ϕ_2 on the disjoint union of G and G_0. So $\mathcal{A}$ returns NP for ϕ_2, and our conclusion is correct.

4 Conclusion and Future Work

In this paper, we systematically study the time complexity of Courcelle's theorem, towards understanding its dependency on the MSO formula ϕ and the treewidth parameter t in a fine-grained way. We prove (almost) matching upper and lower bounds for the time complexity in terms of the quantifier structure of ϕ. We expect this work

to be a starting point of the long-term research towards thoroughly understanding the time complexity of Courcelle's theorem.

Below we pose some open questions for future study. First, the bound in Theorem 1 (and also Theorem 7) is only *almost* tight because of the $\hat{O}(\cdot)$-notation. It is thus natural to ask whether one can replace $\hat{O}(\cdot)$ with $O(\cdot)$ to make the bound *exactly* tight. Second, we only investigated the time complexity in terms of the *quantifier* structure of ϕ, while completely ignoring the quantifier-free part of ϕ. It might be interesting to study Courcelle's theorem in terms of even more fine-grained structural parameter of ϕ. Finally, we only considered treewidth in this paper. Variants of MSO Testing have been also studied with other width parameters of graphs, e.g., clique-width. Therefore, it is also worth studying fine-grained bounds for the complexity of (variants of) Courcelle's theorem with those parameters.

Acknowledgments

The research of Saket Saurabh was partially supported by the European Research Council (ERC) grant no. 819416, and Swarnajayanti Fellowship no. DST/SJF/MSA01/2017-18. The research of Meirav Zehavi was partially supported by the European Research Council (ERC) grant no. 101039913. The authors would like to thank the anonymous reviewers for their insightful comments on this paper.

References

[1] Hans L Bodlaender, Pål Grønås Drange, Markus S Dregi, Fedor V Fomin, Daniel Lokshtanov, and Michał Pilipczuk. 2016. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.* 45, 2 (2016), 317–378. doi:10.1137/130947374

[2] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. 1992. Automatic Generation of Linear-Time Algorithms from Predicate Calculus Descriptions of Problems on Recursively Constructed Graph Families. *Algorithmica* 7, 5&6 (1992), 555–581. doi:10.1007/BF01758777

[3] Glencora Borradaile and Hung Le. 2016. Optimal Dynamic Program for r-Domination Problems over Tree Decompositions. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark (LIPIcs, Vol. 63)*, Jiong Guo and Danny Hermelin (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 8:1–8:23. doi:10.4230/LIPICS.IPEC.2016.8

[4] Bruno Courcelle. 1990. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and computation* 85, 1 (1990), 12–75. doi:10.1016/0890-5401(90)90043-H

[5] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. 2022. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. *ACM Trans. Algorithms* 18, 2 (2022), 17:1–17:31. doi:10.1145/3506707

[6] Baris Can Esmer, Jacob Focke, Dániel Marx, and Pawel Rzazewski. 2024. Fundamental Problems on Bounded-Treewidth Graphs: The Real Source of Hardness. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia (LIPIcs, Vol. 297)*, Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 34:1–34:17. doi:10.4230/LIPICS.ICALP.2024.34

[7] Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. 2023. Tight Complexity Bounds for Counting Generalized Dominating Sets in Bounded-Treewidth Graphs. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, Nikhil Bansal and Viswanath Nagarajan (Eds.). SIAM, 3664–3683. doi:10.1137/1.9781611977554.CH140

[8] Fedor V Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. 2012. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 470–479. doi:10.1109/FOCS.2012.62

[9] Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, Michał Pilipczuk, and Marcin Wrochna. 2018. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms (TALG)* 14, 3 (2018), 1–45. doi:10.1145/3186898

[10] Markus Frick and Martin Grohe. 2004. The complexity of first-order and monadic second-order logic revisited. *Annals of pure and applied logic* 130, 1-3 (2004), 3–31. doi:10.1016/j.apal.2004.01.007

[11] Jakub Gajarský and Petr Hlinený. 2015. Kernelizing MSO Properties of Trees of Fixed Height, and Some Consequences. *Log. Methods Comput. Sci.* 11, 1 (2015). doi:10.2168/LMCS-11(1:19)2015

[12] Jakub Gajarský, Michał Pilipczuk, Marek Sokołowski, Giannos Stamoulis, and Szymon Toruńczyk. 2024. Elementary first-order model checking for sparse graphs. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science* (Tallinn, Estonia) *(LICS '24)*. Association for Computing Machinery, New York, NY, USA, Article 36, 14 pages. doi:10.1145/3661814.3662094

[13] Tim A. Hartmann and Dániel Marx. 2025. Independence and Domination on Bounded-Treewidth Graphs: Integer, Rational, and Irrational Distances. In *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025, March 4-7, 2025, Jena, Germany (LIPIcs, Vol. 327)*, Olaf Beyersdorff, Michal Pilipczuk, Elaine Pimentel, and Kim Thang Nguyen (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 44:1–44:19. doi:10.4230/LIPICS.STACS.2025.44

[14] Joachim Kneis and Alexander Langer. 2009. A practical approach to Courcelle's theorem. *Electronic Notes in Theoretical Computer Science* 251 (2009), 65–81. doi:10.1016/j.entcs.2009.08.028

[15] Tuukka Korhonen. 2023. A single-exponential time 2-approximation algorithm for treewidth. *SIAM J. Comput.* 0 (2023), FOCS21–174. doi:10.1137/22M147551X

[16] Stephan Kreutzer and Siamak Tazari. 2010. Lower Bounds for the Complexity of Monadic Second-Order Logic. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*. IEEE Computer Society, 189–198. doi:10.1109/LICS.2010.39

[17] Stephan Kreutzer and Siamak Tazari. 2010. On Brambles, Grid-Like Minors, and Parameterized Intractability of Monadic Second-Order Logic. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, Moses Charikar (Ed.). SIAM, 354–364. doi:10.1137/1.9781611973075.30

[18] Michael Lampis. 2012. Algorithmic Meta-theorems for Restrictions of Treewidth. *Algorithmica* 64, 1 (2012), 19–37. doi:10.1007/S00453-011-9554-X

[19] Michael Lampis. 2023. First Order Logic on Pathwidth Revisited Again. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 261)*, Kousha Etessami, Uriel Feige, and Gabriele Puppis (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 132:1–132:17. doi:10.4230/LIPIcs.ICALP.2023.132

[20] Michael Lampis. 2023. First Order Logic on Pathwidth Revisited Again. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany (LIPIcs, Vol. 261)*, Kousha Etessami, Uriel Feige, and Gabriele Puppis (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 132:1–132:17. doi:10.4230/LIPICS.ICALP.2023.132

[21] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2018. Known Algorithms on Graphs of Bounded Treewidth Are Probably Optimal. *ACM Trans. Algorithms* 14, 2 (2018), 13:1–13:30. doi:10.1145/3170442

[22] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2018. Slightly Superexponential Parameterized Problems. *SIAM J. Comput.* 47, 3 (2018), 675–702. doi:10.1137/16M1104834

[23] Daniel Lokshtanov, Ivan Mikhailin, Ramamohan Paturi, and Pavel Pudlák. 2018. Beating brute force for (quantified) satisfiability of circuits of bounded treewidth. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 247–261. doi:10.1137/1.9781611975031.18

[24] BA Trakhtenbrot. 1950. The impossibility of an algorithm for the decision problem for finite domains (Russian). *Doklady Akademii Nauk SSSR* 70 (1950), 569–572.

Received 2025-11-04; accepted 2026-02-01