Approximate Evaluation of First-Order Counting Queries*

Jan Dreier† Peter Rossmanith‡

Abstract

Kuske and Schweikardt introduced the very expressive first-order counting logic FOC(**P**) to model database queries with counting operations. They showed that there is an efficient model-checking algorithm on graphs with bounded degree, while Grohe and Schweikardt showed that probably no such algorithm exists for trees of bounded depth. We analyze the fragment FO($\{>0\}$) of this logic. While we remove for example subtraction and comparison between two non-atomic counting terms, this logic remains quite expressive: We allow nested counting and comparison between counting terms and arbitrarily large numbers. Our main result is an approximation scheme of the model-checking problem for FO($\{>0\}$) that runs in linear fpt time on structures with bounded expansion. This scheme either gives the correct answer or says "I do not know." The latter answer may only be given if small perturbations in the number-symbols of the formula could make it both satisfied and unsatisfied. This is complemented by showing that exactly solving the model-checking problem for FO($\{>0\}$) is already hard on trees of bounded depth and just slightly increasing the expressiveness of FO($\{>0\}$) makes even approximation hard on trees.

1 Introduction

One important task for database systems is to lookup information, which is usually done in the form of *queries*. For most modern relational database management systems, queries are written in the SQL language, whose logical foundation, the relational calculus, is equivalent to first-order logic [12]. This means in particular that every first-order sentence can be expressed in SQL.

Databases can be represented as relational structures. The fundamental problem for first-order logic that corresponds to the evaluation of a boolean SQL query in a database is the so called *model-checking problem*: Given a logical formula φ and a structure G, decide whether φ is true for G, i.e., whether G is a model of φ (commonly written $G \models \varphi$). We consider the model-checking problem to be *fixed-parameter tractable* (fpt) if it can be solved in time $f(|\varphi|)\|D\|^c$ for some function f and constant c (where $|\varphi|$ is the length of the formula and $\|D\|$ the size of the database). Already for first-order logic, the model-checking problem is AW[$*$]-complete and therefore unlikely to be fpt, which means that boolean SQL queries are also hard (and SQL allows other types of queries, too). In fact, even for purely existential formulas the model-checking problem is $W[1]$-hard because finding a k-clique is a special case [4]. It is therefore natural to ask which classes of structures still admit fpt model-checking algorithms. Since relational structures can be represented by their Gaifman graph (see, e.g., [15] for details of the construction), the above question can be reformulated as the question for *graph classes* with fpt model-checking algorithms.

Another motivation for the model-checking problem are *algorithmic meta-theorems* [18]. If a problem can be formulated in a certain logic then the model-checking algorithm for this logic can solve it. Therefore, model-checking results can be seen as meta-theorems that prove whole families of problems to be algorithmically tractable on certain classes of inputs.

For all graph classes with bounded tree-width Courcelle's theorem states that the model-checking problem for monadic second-order logic can be solved in time $f(|\varphi|)|G|$ and therefore is fpt [2]. Frick and Grohe showed that the dependence on φ is non-elementary [11] on specially constructed worst-case instances, while implementations exist that perform quite well on "usual" inputs [20]. However, graph classes with bounded tree-width are very restricted. It has been shown in a series of papers that the first-order model-checking problem is efficiently solvable for more and more sparse graph classes, such as those with bounded degree [27], excluded minor [9], or locally bounded tree-width [10], culminating in two relatively recent results: Dvořák, Kráľ, and Thomas found a linear fpt algorithm for graph classes of bounded expansion [7], and Grohe, Kreutzer, and Siebertz [13] provided an algorithm that has run time $f(\varepsilon, |\varphi|)|G|^{1+\varepsilon}$ for every $\varepsilon > 0$ for nowhere dense graph classes. Nešetřil and Ossona de Mendez introduced the concepts of bounded expansion and nowhere dense graph classes that generalize all previously mentioned sparse graph classes. They are general enough to capture certain real-world graphs, as some observations are suggesting [3]. On the other hand, first-order model-checking cannot be fixed parameter tractable on monotone graph classes that are not nowhere dense unless AW[$*$] = FPT [13]. This makes nowhere dense and bounded expansion graph classes two of the most general sparse graph classes that still are algorithmically

*Supported by the German Science Foundation (DFG) under grant no. DFG-927/15-1.

†RWTH Aachen University.

‡RWTH Aachen University.

useful.

While it is settled that (at least for monotone graph classes) we cannot find fpt first-order model-checking algorithms beyond nowhere dense graph classes unless AW[$*$] = FPT, it is still very much an open question by how much we can extend first-order logic while keeping the graph classes as general as possible. This is all the more important since many features of SQL, such as the COUNT operator, cannot be properly modeled in first-order logic. First-order formulas can only make counting-claims of the form "there are at least k elements with this property" for some fixed $k \in \mathbf{N}$. Model-checking results for extensions of first-order logic yield more general algorithmic meta-theorems, capturing wider ranges of problems.

A limited way to bring counting ability to first-order logic is the *query-counting* problem, where one is given a first-order formula $\varphi(\bar{x})$ with free variables $\bar{x} = x_1 \dots x_k$ and a structure G and asked to count the number of tuples $\bar{v}$ of vertices in G such that $G \models \varphi(\bar{v})$. This problem is fixed-parameter tractable on nowhere dense graph classes [15]. A closely related problem is the *query-enumeration* problem where one is asked to enumerate satisfying tuples (as in a typical SELECT-statement in SQL). Kazana and Segoufin showed that this problem is tractable on graph classes of bounded expansion [17] and later Schweikardt, Segoufin, and Vigny generalized it to nowhere dense graph classes [26].

For more powerful counting mechanisms we are required to extend first-order logic itself. While many ways have been considered to bring counting to first-order logic [29, 25, 21, 14], we consider the first-order counting logic FOC($\mathbf{P}$) recently introduced by Kuske and Schweikardt [19]. In this logic formulas are built according to the rules of first-order logic and from *counting terms*: A counting term is any number $N \in \mathbf{Z}$ as well as formulas such as $\#y\,\varphi$ standing for "the number of witnesses for y in φ." Counting terms are allowed to be multiplied, added, subtracted and compared using a collection $\mathbf{P}$ of *numerical predicates*. The precise syntax and semantics can be found in [19]. In this work, we restrict ourselves to the binary numerical predicate $>$ denoting the usual "greater than" relation. The semantics of FOC($\{>\}$) are best illustrated with the help of examples: The formula

$$\exists x_1 \cdots \exists x_k \big(\#\, y \bigvee_{i=1}^{k} (x_i = y \vee E(x_i, y)) > N\big), \tag{1.1}$$

when evaluated on graphs, expresses that there are k vertices that dominate more than N vertices. This describes the *partial dominating set problem*, which is a generalization of *dominating set* where *all* vertives have to be dominated.

The formula

$$\big(\#(x_1, \dots, x_k) \bigwedge_{i \neq j} E(x_i, x_j)\big) > \big(\#(x_1, \dots, x_k) \bigwedge_{i \neq j} \neg E(x_i, x_j)\big) \tag{1.2}$$

expresses that there are more cliques of size k than independent sets of size k. Note that the *length* of each number $N \in \mathbf{Z}$ in a formula is considered to be one. This means formula (1.1) always has constant length and an fpt model-checking algorithm for FOC($\mathbf{P}$) is required to evaluate it in the same time $f(k)\|G\|^c$ for any N, even if N depends on G.

Kuske and Schweikardt showed recently that the model-checking problem for FOC($\mathbf{P}$) is fixed parameter tractable for graphs with bounded degree [19]. However, already on simple structures, such as trees of bounded depth, the problem becomes AW[$*$]-complete [15], and therefore is most likely not fpt. It seems like the expressive power of FOC($\mathbf{P}$) is too strong to admit efficient model-checking algorithms on more general graph classes. This invites the question for fragments of FOC($\mathbf{P}$) that still admit efficient model-checking algorithms on graph classes with bounded expansion or nowhere dense classes. In this work, we identify such a fragment.

But let us first mention another fragment (orthogonal to ours), introduced by Grohe and Schweikardt [15]. The fragment FOC$_1$($\mathbf{P}$) is obtained from FOC($\mathbf{P}$) by allowing subformulas of the form $P(t_1, \dots, t_m)$ for some numerical predicate $P \in \mathbf{P}$ only if all counting terms $t_1, \dots, t_m$ together contain at most a single free variable. The above formula (1.2) is in FOC$_1$($\{>\}$), since both counting terms have zero free variables. However, the formula (1.1) for partial dominating set is not in FOC$_1$($\{>\}$) (unless $k = 1$) because the counting term $\#y \bigvee_{i=1}^{k}(x_i = y \vee E(x_i, y))$ has k free variables. Grohe and Schweikardt showed that the model-checking problem for FOC$_1$($\mathbf{P}$) is fixed parameter tractable on nowhere dense graph classes [15]. For bounded expansion graph classes, Toruńczyk presents an even stronger query language (orthogonal to ours) that extends first-order logic by aggregation in multiple semirings [28].

1.1 Results. In this work, we consider the fragment of FOC($\{>\}$) built recursively using the rules of first-order logic and the following rule:

> If φ is a formula, y is a variable, and $N \in \mathbf{Z}$, then $\#y\,\varphi > N$ is a formula.

Except for syntactic differences, this fragment is equivalent to the logic FO($\{>0\}$) that was defined by Kuske

and Schweikardt [19], where ">0" stands for the unary predicate testing whether a term is positive. In their definition of FO($\{>0\}$) one may write $\#y\,\varphi - N > 0$ instead of $\#y\,\varphi > N$. To avoid having multiple names for the same logic and because all our results are independent of such syntactic differences, we call our fragment FO($\{>0\}$) as well. This logic further exists under the name FO(C) [8]

Formula (1.1) modelling partial dominating set is in FO($\{>0\}$) while (1.2) compares two non-constant counting terms and therefore is *not* a FO($\{>0\}$) formula. This makes FO($\{>0\}$) and $\mathrm{FOC}_1(\{>\})$ incomparable. But while model-checking for $\mathrm{FOC}_1(\mathbf{P})$ is fixed parameter tractable on nowhere dense graph classes, FO($\{>0\}$) is still too expressive to allow efficient model-checking: We prove that, just like FOC($\mathbf{P}$), the model-checking problem for FO($\{>0\}$) is AW[$*$]-hard even on trees of bounded depth (Lemma 4.1).

For this reason, we define the concept of *approximate model-checking*. An approximate model-checking algorithm gets as input a graph G, a formula φ and an accuracy $\varepsilon > 0$, runs in time $f(|\varphi|,\varepsilon)|G|$, and either returns 1 (meaning $G \models \varphi$), 0 (meaning $G \not\models \varphi$), or $\perp$ (meaning "I do not know.") The symbol $\perp$ may only be returned if slight perturbations in the constants of φ make the formula both satisfied and unsatisfied. For smaller values of ε, these perturbations must be smaller, too. Our main result is the following:

THEOREM 1.1. *There is a linear fpt model-checking approximation scheme for* FO($\{>0\}$) *on labeled graph classes with bounded expansion.*

This means that for every graph class with bounded expansion there exists a function f such that the model-checking problem for FO($\{>0\}$) on this graph class can be approximated with an arbitrary accuracy $\varepsilon > 0$ in time $f(|\varphi|,\varepsilon)|G|$ (Definition 2.4).

Let us now describe when the approximation algorithm is allowed to answer $\perp$. For $\lambda > 1$ we call two formulas λ-*similar* if one formula can be obtained from the other one by changing the constant counting terms by a factor between $1/\lambda$ and λ. The two FO($\{>0\}$)-formulas

$$\#y(\#z\,\varphi(yz) < 500) > 1000$$

$$\#y(\#z\,\varphi(yz) < 498) > 1010$$

are 1.01-similar, but not 1.009-similar. We further say a formula φ is λ-*unstable* on a graph G if φ is λ-similar to two formulas φ' and φ'' such that $G \models \varphi'$ and $G \not\models \varphi''$. For a given $\varepsilon > 0$, the approximation algorithm is only allowed to answer $\perp$ if the input formula φ is $(1+\varepsilon)$-unstable on the input graph G. Note that formulas without counting quantifiers are never unstable and may never lead to the answer $\perp$. Our approximation scheme therefore generalizes the first-order model-checking problem.

It can be argued that answering queries approximately is in many applications almost as good as an exact answer because the involved numbers (like a maximal debt of one million dollars or a maximum allowed temperature of 1000 degrees) are often only ballpark numbers. Furthermore, if $\perp$ is returned we know that the formula is "close" to being satisfied and unsatisfied, which sometimes may be interesting in itself. For example, if the partial dominating set formula (1.1) is $(1+\varepsilon)$-unstable then there exists a solution dominating more than $N/(1+\varepsilon)$ vertices, but none dominating more than $(1+\varepsilon)N$ vertices.

A natural question that arises is whether FO($\{>0\}$) can be generalized while keeping the ability of efficient approximate model-checking. We answer this question negatively. If we make FO($\{>0\}$) just slightly more powerful then approximate model-checking already becomes AW[$*$]-hard on trees of bounded depth. This happens if we allow either counting quantification on pairs of variables, subtraction, multiplication, or comparison between two non-constant counting terms, and even for very large "approximation ratios" (Lemma 4.3). The FO($\{>0\}$) fragment seems to be at the edge of what can still be efficiently approximated on bounded expansion graph classes.

Nevertheless, we can identify certain FO($\{>0\}$)-formulas that can be evaluated exactly. They are of the form $\exists x_1 \ldots \exists x_k \#y\,\varphi(y\bar{x})$ where $\varphi(y\bar{x})$ is a first-order formula. Since the previously mentioned partial dominating set formula (1.1) is of this shape, we get the following result:

COROLLARY 1.1. *Partial dominating set can be solved in linear fpt time on graph classes with bounded expansion.*

Amini, Fomin, and Saurabh [1] showed that partial dominating set can be solved in fpt time on minor-closed graph classes, but the complexity on graph classes with bounded expansion has remained open. Moreover, the running time of Amini et al.'s algorithm for an H-minor free graph class is of the form $f(k)n^{c_H}$, where n is the number of vertices and c_H is a constant that depends on H, while our running time is linear in n. We get the same running time for similar problems such as distance-r dominating set or variants of partial vertex cover. We are further able to solve a general optimization problem where the goal is to retrieve an optimal witnesses for the free variables of a counting formula.

THEOREM 1.2. *Let $\mathcal{C}$ be a labeled graph class with bounded expansion. There exists a function f such that for a given graph $G \in \mathcal{C}$ and first-order formula $\varphi(y\bar{x})$ one can compute in time $f(|\varphi|)\|G\|$ a tuple $\bar{u}^* \in V(G)^{|\bar{x}|}$ such that*

$$[\![\#y\,\varphi(y\bar{u}^*)]\!]^G = \operatorname*{opt}_{\bar{u}\in V(G)^{|\bar{x}|}} [\![\#y\,\varphi(y\bar{u})]\!]^G,$$

where opt *is either* min *or* max.

This means, for example, that we can find an optimal partial dominating set of size k in linear fpt time, which is faster than using self-reducibility. The proof of Theorem 1.2 has been omitted. It can be found in the corresponding full version of this paper [5].

1.2 Techniques. Most of our proofs use functional structures to represent graphs. The overall strategy of our main result in Theorem 1.1 is the use of quantifier elimination to replace the model-checking problem by one with one counting quantifier less until we reach a quantifier-free formula. To eliminate a counting quantifier, we perform a sequence of transformations on counting terms of the form $\#y\,\varphi(y\bar{x})$ where $\varphi(y\bar{x})$ is quantifier-free. We replace them with a sum of gradually simpler counting terms until they are simple enough to be directly evaluated. Most transformations preserve the value of the counting term. In the end, however, we have to replace each summand with an approximation of it. This leads to a problem at one point. We can express a counting term via inclusion-exclusion as $a - b$ for two terms a and b. If we have an approximation a' of a and b' of b with good relative error and $a = b$ are very large then $a' - b'$ may be a very bad approximation of $a - b = 0$. This has to be avoided. If b is rather small we can ignore it and just use a as a good approximation of $a - b$. If b is big, however, something needs to be done. By a preprocessing of the graph during which we add so-called "flip" arcs we modify it in such a way that the subtraction $a - b$ can be done exactly whenever necessary. This is the most crucial step in the proof as we have to be very careful to add enough arcs to achieve the necessary precision, while still staying in a graph class with bounded expansion. At last, we have approximated the counting term $\#y\,\varphi(y\bar{x})$ using a sum of simpler counting terms of the form $\#y\,\psi(yx_i)$. Each simpler counting term depends only on one free variable and can therefore be evaluated in linear fpt time. Then we round the resulting numbers into a constant number of intervals and introduce unary predicates indicating the intervals in which the numbers lie. This is the second situation where we loose precision. Using these predicates we can finally get rid of the counting quantifier and replace a subformula of the form $\#y\,\varphi(y\bar{x}) > N$ with a quantifier-free one. Due to the previously introduced errors, the new formula may not always give us the correct answer. Therefore we build a pair of quantifier-free formulas: one over- and one underapproximation. If they agree, we know the correct answer. If they disagree, we can be sure that the situation is unstable.

If the term $\#y\,\varphi(y\bar{x})$ is not part of a larger formula, we we can stop the quantifier elimination step early. We avoid the problem of subtraction and the encoding into unary predicates and instead evaluate the simplified intermediate counting terms directly using standard methods. This means that we can solve the model-checking problem for such formulas *exactly* in linear fpt time, giving rise to Theorem 1.2.

The remaining paper is structured as follows: We start by introducing the necessary notation. Then in Section 3 we develop the machinery for quantifier elimination in functional representations of graphs. We then prove the main result stating that there is an approximate model-checking algorithm for FO($\{>0\}$) on graph classes with bounded expansion (Theorem 1.1). In Section 4 we prove the hardness of exact model-checking for FO($\{>0\}$) and approximate model-checking for generalizations of FO($\{>0\}$) (Lemma 4.1 and 4.3).

2 Definitions and Notation

Graphs. In this paper we consider *labeled graphs*. A labeled graph is a tuple $G = (V, E, P_1, \ldots, P_m)$, where V is the vertex set, E is the edge set and $P_1, \ldots, P_m \subseteq V$ the labels of G. The *order* $|G|$ of G equals $|V|$. We define the *size* $\|G\|$ of G as $|V| + |E| + |P_1| + \cdots + |P_m|$. Unless otherwise noted, our graphs are undirected. For a directed graph G, the indegree of a node v equals the number of vertices u such that there is an arc uv in G. The maximal indegree of all nodes in G is denoted by $\Delta^-(G)$.

Logic. We consider fragments of the very general first-order counting logic FOC($\mathbf{P}$) defined by Kuske and Schweikardt [19]. It depends on a collection $\mathbf{P}$ of numerical predicates, where each predicate $P \in \mathbf{P}$ has semantics specified by $[\![P]\!]$. We consider fragments of FOC($\{>\}$) where $>$ is the usual "greater than" predicate with $[\![>]\!] = \{\,(a, b) \in \mathbf{Z}^2 \mid a > b\,\}$. As we will only use a subset of FOC($\{>\}$), we refrain from giving the whole definition. Instead, we define fragments of FOC($\{>\}$), as we introduce them. The semantics of FOC($\mathbf{P}$) are as expected and we refer the reader to [19] for a rigorous definition.

DEFINITION 2.1. *We define* FO($\{>0\}$) *to be the fragment of* FOC($\{>\}$) *built using the rules of first-order*

logic (rule 1,2,3 in [19, Definition 2.1]) and the following rule:

> *If φ is a formula, y is a variable, and $N \in \mathbf{Z}$, then $\#y\,\varphi > N$ is a formula.*

Except for syntactic differences, this definition is equivalent to the original definition of FO($\{>0\}$) provided by Kuske and Schweikardt [19] (with the original syntax one has to write $\#y\,\varphi - N > 0$ instead of $\#y\,\varphi > N$).

We say a FOC($\mathbf{P}$) formula is *quantifier-free* if it contains no $\exists$, $\forall$, or $\#$ quantifiers. If two formulas φ_1 and φ_2 are logically equivalent we write $\varphi_1 \equiv \varphi_2$. The *length* of a formula φ is denoted by $|\varphi|$ and equals its number of symbols. In particular, the length of any number-symbol $N \in \mathbf{Z}$ in a FOC($\mathbf{P}$) formula is one (and should not be confused with the length of a binary encoding of N). For two signatures we write $\sigma \subseteq \rho$ to indicate that ρ extends σ. All signatures are finite and the cardinality $|\sigma|$ of a signature equals its number of symbols. We often interpret a conjunctive clause $\omega \in$ FO as a set of literals and write $l \in \omega$ to indicate that l is a literal of ω.

We denote the universe of a structure G by $V(G)$. We interpret a labeled graph $G = (V, E, P_1, \dots, P_m)$ as a logical structure with universe V, binary relation E and unary relations $P_1, \dots, P_m$.

By $\bar{x}$ we denote a non-empty tuple $x_1 \dots x_{|\bar{x}|}$. We write $\varphi(\bar{x})$ to indicate that a formula φ has free variables $\bar{x}$. Let G be a structure, $\bar{u} \in V(G)^{|\bar{x}|}$ be a tuple of elements from the universe of G, and β be the assignment with $\beta(x_i) = u_i$ for $i \in \{1, \dots, |\bar{x}|\}$. For simplicity, we write $G \models \varphi(\bar{u})$ and $[\![\varphi(\bar{u})]\!]^G$ instead of $(G, \beta) \models \varphi(\bar{x})$ and $[\![\varphi(\bar{x})]\!]^{(G,\beta)}$.

Further notation concerned with functional structures and formulas is introduced in Section 3.2.

Model-Checking. Let $\mathcal{C}$ be a class of structures and L be a logic. The *parameterized model-checking problem for* L *on* $\mathcal{C}$ is the defined as follows: The input is a structure $G \in \mathcal{C}$ and a sentence $\varphi \in$ L with matching signatures. The parameter is $|\varphi|$. The question is whether $G \models \varphi$. The parameterized first-order model-checking problem on the class of all graphs is a complete problem for the complexity class AW[$*$]. As the whole W-hierarchy is contained in AW[$*$] it is generally assumed that AW[$*$] $\not\subseteq$ FPT.

A Model-Checking Approximation Scheme. Next we define the novel notion of a model-checking approximation scheme, an fpt algorithm that is only allowed to answer "I do not know" if the fact whether the structure is a model of the formula is sensitive to slight perturbations in the constants of the formula.

Definition 2.2. (λ-similarity) *Let $\lambda > 1$ and φ be a* FOC($\mathbf{P}$) *formula. A* FOC($\mathbf{P}$) *formula φ' is* λ-similar *to φ if φ' can be obtained from φ by replacing each atomic counting term $t \in \mathbf{Z}$ of φ by $t' \in \mathbf{Z}$ with $t/\lambda \leq t' \leq \lambda t$.*

Definition 2.3. (λ-stability) *Let $\lambda > 1$, G be a structure and φ be a* FOC($\mathbf{P}$) *sentence. We say φ is* λ-stable *on G if for every* FOC($\mathbf{P}$) *sentence φ' which is λ-similar to φ it holds that $G \models \varphi$ iff $G \models \varphi'$. Otherwise we say that φ is* λ-unstable *on G.*

Definition 2.4. (linear fpt model-checking approximation scheme) *Let $\mathcal{C}$ be an arbitrary class of labeled graphs, and* L *be a fragment of* FOC($\mathbf{P}$). *A* linear fpt model-checking approximation scheme *for the logic* L *on the class $\mathcal{C}$ is an algorithm that gets as input a sentence $\varphi \in$ L, a graph $G \in \mathcal{C}$ and $\varepsilon > 0$, runs in time at most $f(|\varphi|, \varepsilon)\|G\|$ for some function f and returns either 1, 0, or $\perp$.*

- *If the algorithm returns 1 then $G \models \varphi$.*
- *If the algorithm returns 0 then $G \not\models \varphi$.*
- *If the algorithm returns $\perp$ then φ is $(1+\varepsilon)$-unstable on G.*

3 Approximate Model-Checking

We will work with graph classes with bounded expansion and use their characterization via transitive fraternal augmentations (Section 3.1). An undirected graph is first replaced with a directed graph by orienting the edges in such a way that the indegree is bounded by a constant that depends only on the graph class. We represent this directed graph by a functional structure (Section 3.2). The signature of this structure consists of a constant number of function symbols (usually denoted by f, g, h) and unary predicate symbols. The function symbols represent arcs. If $f(u) = v$ for some function f, then the corresponding directed graph has an arc vu. In this way, we need only as many function symbols as the indegree of the directed graph.

Our model-checking algorithm works via *quantifier elimination*. This means, we gradually simplify the input formula by iteratively removing the innermost quantifier. We compensate every removed quantifier by adding new arcs and unary relations to our input structure (maintaining bounded expansion). When no quantifiers are left we can easily evaluate the formula. In this procedure, the subformulas spanned by the innermost quantifier are of the form $\#y\,\varphi(y\bar{x}) > N$ (where φ is quantifier-free). We want to replace such a formula with two almost equivalent quantifier-free formulas (Section 3.5). If evaluating these formulas on a graph gives two different results then we know that

$\#y\,\varphi(y\bar{x})$ is up to a factor of $(1+\varepsilon)$ close to N and we are allowed to return $\perp$.

In Section 3.3 we gradually transform the innermost counting term $\#y\,\varphi(y\bar{x})$ into simpler terms while expanding the corresponding functional structure with new arcs and unary relations. In the end, we obtain a sum of t simpler terms of the form $\#y\,\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i)$. If we ignore $\psi(\bar{x})$, the simpler terms only have a single free variable and can be evaluated in linear time for all inputs. We divide the numbers from 0 to N into t/ε buckets and introduce unary predicates $R_0, \ldots, R_{t/\varepsilon}$, where $R_l(x_i)$ is true if and only if the value of $\#y\,\tau(y) \wedge f(y) = g(x_i)$ is in the lth bucket. Thus if for each summand and each l we know the value of $\psi(\bar{x})$ and $R_l(x_i)$, we either know that $\#y\,\varphi(y\bar{x})$ is greater than N or can reconstruct its value of up to a factor of $(1+\varepsilon)$ (Section 3.4). This reconstruction can be done in a quantifier-free first-order formula with free variables $\bar{x}$, which completes the quantifier elimination. Note that summands are not allowed to be negative since due to cancellation the magnitude of individual summands could be considerably larger than the final sum and the bucket-rounding technique would not work.

The main challenge is to find a decomposition of $\#y\,\varphi(y\bar{x})$ into summands of the form $\#y\,\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i)$. This transformation is done in several stages. The first intermediate step are formulas that consist of conjunctive clauses that can be grouped as $\tau(y) \wedge \psi(\bar{x}) \wedge \Delta^{=}(y\bar{x}) \wedge \Delta^{\neq}(y\bar{x})$ and is carried out in a similar way to what Kazana and Segoufin did [17]. Here all $\tau(y)$, $\psi(\bar{x})$, $\Delta^{=}(y\bar{x})$, and $\Delta^{\neq}(y\bar{x})$ are conjunctions of atomic formulas, which we also call *literals*. Those literals that contain only y are grouped into $\tau(y)$, those with variables only from $\bar{x}$ into $\psi(\bar{x})$. We call the remaining ones the *mixed literals*, as they depend on y and $\bar{x}$. We make sure that they are either of the form $f(y) = g(x_i)$ or $f(y) \neq g(x_i)$. The former ones are placed into $\Delta^{=}(y\bar{x})$ and the latter ones into $\Delta^{\neq}(y\bar{x})$.

Let us replace the input graph with its 1-transitive fraternal augmentation. By the fraternal rule, for every original function symbol f, f' and every vertex v there is a function symbol h in the transitive fraternal augmentation with either $f(v) = h(f'(v))$ or $h(f(v)) = f'(v)$. By expanding each conjunctive clause, we take care that $\tau(y)$ contains every possible literal of the form $h(f(y)) = g(y)$ or its negation. Similarly for literals $h(f(x_i)) = g(x_j)$ in $\psi(\bar{x})$. This creates redundancy that helps us replace mixed literals. We will proceed in a similar way as Kazana and Segoufin, but have to be a bit more careful about not overcounting, as we eliminate a counting quantifier rather than an existential one. As the next step, we make sure that $\Delta^{=}(y\bar{x})$ contains only one literal. The resulting formulas $\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge \Delta^{\neq}(y\bar{x})$ are one step closer to their final form. It remains to eliminate the negative literals in $\Delta^{\neq}(y\bar{x})$. A standard way to do so would be inclusion-exclusion. But this is not allowed since it would lead to subtraction, which cannot be approximated. Finally, we have to use very different techniques than Kazana and Segoufin [17].

If the negative mixed literals in $\Delta^{\neq}(y\bar{x})$ are satisfied by almost all of the witnesses for y, then removing them increases the final count only a little bit. This way we obtain a good enough approximation. If this does not work, we introduce so called "flip" arcs (see Section 3.1) to the graph and exploit the redundant literals added to $\tau(y)$ and $\psi(\bar{x})$. The redundancy in $\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i)$ together with the new arcs then imply $\Delta^{\neq}(y\bar{x})$ to be either always true or always false. In the former case, we remove $\Delta^{\neq}(y\bar{x})$ from its conjunctive clause, and in the latter case we can remove the whole conjunctive clause. Our key observation is that we only need to introduce a small amount of "flip" arcs and therefore stay within a graph class with bounded expansion.

3.1 Transitive Fraternal Flip Augmentations.

A directed graph G' is a 1-*transitive fraternal augmentation* of a directed graph G if it has the same vertex set as G and satisfies the following conditions [24]:

- **Transitivity.** If the arcs uv and vw are present in G then uw is present in G'.
- **Fraternity.** If uw and vw are present in G then uv or vu are present in G'.
- **Tightness.** If G' contains an arc that is not present in G it must have been added by one of the previous two rules.

Let G be an undirected graph. We call a sequence $G_0 \subseteq G_1 \subseteq \cdots$ a *transitive fraternal augmentation* of G if G_0 is a directed graph obtained by orienting the edges of G, and G_{i+1} is a 1-transitive fraternal augmentation of G_i for $i \geq 0$. For any graph class $\mathcal{C}$ with bounded expansion Nešetřil and Ossona de Mendez devised an algorithm [23] that computes a transitive fraternal augmentation $G_0 \subseteq G_1 \subseteq G_2 \subseteq \cdots$ of G such that $\Delta^{-}(G_i) \leq \Gamma_{\mathcal{C}}(i)$, $i \in \mathbf{N}$ for a function $\Gamma_{\mathcal{C}}$ that depends only on the graph class $\mathcal{C}$. The orientation G_0 can be computed in time $O(\|G\|)$ from G, and G_{i+1} can be computed from G_i in time $O(\|G_i\|)$. We will assume that this algorithm is used to compute augmentations and orientations and call the corresponding output *the* 1-transitive fraternal augmentation and *the* orientation, similarly to what Kazana and Segoufin did [17].

We will need a generalization of this construction. We say a directed graph G' is a flip of G if G' is a supergraph of G with the same vertex set that can contain additional "flipped" arcs, i.e., G' can have an additional arc uv only if G already contains vu. Moreover, we require that $\Delta^-(G') \leq \Delta^-(G) + 1$. In a flip, with other words, we can add reverse arcs for arcs that are already present without increasing the maximal indegree by more than one. A sequence $G_0 \subseteq G_1 \subseteq G_2 \subseteq \cdots$ is called a *transitive fraternal flip augmentation* of G if G_{i+1} is a flip or the 1-transitive fraternal augmentation of G_i for all $i \geq 0$ and G_0 is the orientation of G. Note that we can apply *any* flip but can only apply the "well-behaved" orientation and augmentation devised by Nešetřil and Ossona de Mendez [23]. We can characterize graph classes with bounded expansion via transitive fraternal flip augmentations.

LEMMA 3.1. *Let $\mathcal{C}$ be a graph class. Then $\mathcal{C}$ has bounded expansion if and only if there exists a function $\Gamma_{\mathcal{C}} \colon \mathbf{N} \to \mathbf{N}$ such that every graph $G \in \mathcal{C}$ and every transitive fraternal flip augmentation $G_0 \subseteq G_1 \subseteq G_2 \subseteq \ldots$ of G has $\Delta^-(G_i) \leq \Gamma_{\mathcal{C}}(i)$ for every $i \geq 0$.*

The proof of Lemma 3.1 uses standard arguments from [22, 23] and can be found in the full version of this paper [5].

3.2 Functional Representations. We prove our results using a functional representation of graphs. They were used heavily by Durand and Grandjean [6] and again by Kazana and Segoufin [17], but partially also by Dvořák, Kráľ, and Thomas in the first proof that first-order model-checking is ftp on bounded expansion graph classes [7]. One big advantage of functional representations is the ability to talk about short paths without using quantifiers as long as all indegrees are bounded.

A *functional signature* is a finite signature containing functional symbols of arity one and unary predicates. For a functional signature σ, we will denote the set of function symbols by σ_{fun}. A *functional representation* of a labeled directed graph G is a σ-structure $\vec{G}$. The universe of $\vec{G}$ is $V(G)$. For every label of G there is one unary predicate in $\vec{G}$ representing it. The arcs of G are represented using functions. An arc uv is present in G if and only if $f^{\vec{G}}(v) = u$ for some function symbol $f \in \sigma_{fun}$. Note that we need only $\Delta^-(G)$ different function symbols. Unused function symbols are mapped to the vertex itself, in particular an isolated vertex v has $f^{\vec{G}}(v) = v$ for every f in σ_{fun}. For solely technical reasons, we further require a special function symbol f_{id} where $f_{\mathrm{id}}^{\vec{G}}(v) = v$ for all $v \in V(\vec{G})$. We call G the *underlying directed graph* of $\vec{G}$ and by the *underlying undirected graph* of $\vec{G}$ we mean the underlying undirected graph of G.

We define the *size* $\|\vec{G}\|$ of $\vec{G}$ as $|\vec{G}||\sigma|$ where σ is the signature of $\vec{G}$. We transfer all remaining notation from directed graphs to functional representations as expected. For example $\Delta^-(\vec{G})$ is defined as $\Delta^-(G)$. For a given functional signature σ we define $\mathcal{G}(\sigma)$ to be the class of all functional representations with signature σ.

Our logics FO or FOC($\mathbf{P}$) are defined in the usual way for this functional setting. Note that in particular we allow nested function terms such as $f(g(x))$. The *functional depth* of a formula is the maximum number of nested function applications. For example $f(g(x)) = y$ has functional depth 2. We define $\mathrm{FO}[d, \sigma]$ to be all first-order formulas with functional signature σ and functional depth at most d.

For a given graph G we later want to have a sequence of functional representations $\vec{G}_0 \subseteq \vec{G}_1 \subseteq \ldots$ such that the sequence of underlying directed graphs $G_0 \subseteq G_1 \subseteq \ldots$ forms a transitive fraternal augmentation of G and additionally $\vec{G}_{i+1}$ is an expansion of $\vec{G}_i$ for $i \geq 0$. We will later heavily exploit that for every sentence φ with the same signature as $\vec{G}_0$ and $i > 0$ it holds that $\vec{G}_0 \models \varphi$ iff $\vec{G}_i \models \varphi$.

We extend our notion of 1-transitive fraternal augmentations and flips to functional representations. For a given functional representation $\vec{G}$, we obtain the 1-*transitive fraternal augmentation* $\vec{G}'$ of $\vec{G}$ by adding new function symbols to $\vec{G}$ representing all newly introduced arcs. The functions representing the transitive edges are added in a special way: For all function symbols f, g in the signature of $\vec{G}$ we add a function symbol $h_{f,g}$ to the signature of $\vec{G}'$ and define $h_{f,g}^{\vec{G}'} = g^{\vec{G}'} \circ f^{\vec{G}'}$ representing the newly introduced transitive edges obtained from f and g. This step will later help us simplify our formulas by replacing nested functions of the form $g(f(x))$ with a single function $h_{f,g}(x)$. Fraternal edges are added as well, of course, but we do not require any special naming for them. The construction of $\vec{G}'$ is not necessarily deterministic, but we can assume it to be. Note that if $\Delta^-(\vec{G}')$ is bounded then the signature of $\vec{G}'$ has only a constant number of new function symbols.

A *flip* $\vec{G}'$ of $\vec{G}$ is an expansion of $\vec{G}$ with the same universe and one more functional symbol representing the flipped edges. Since the indegree of a flip may increase by at most one, one new function symbol is sufficient. At last, we define what it means for a class of functional representations to have bounded expansion.

DEFINITION 3.1. *We say a class $\mathcal{C}$ of functional representations has bounded expansion if there exists a functional signature σ such that $\mathcal{C} \subseteq \mathcal{G}(\sigma)$ and the class of all underlying undirected graphs has bounded expansion.*

COROLLARY 3.1. *Let $\mathcal{C}$ be a class of functional representations with bounded expansion. Consider the class $\mathcal{C}'$ of all functional representations $\vec{G}'$ such that $\vec{G}'$ is either the 1-transitive fraternal augmentation or a flip of $\vec{G}$ for some $\vec{G} \in \mathcal{C}$. Then $\mathcal{C}'$ has bounded expansion.*

Let $\mathcal{C}$ be a graph class with bounded expansion. For a graph $G \in \mathcal{C}$ we can compute a functional representation $\vec{G}$ of the orientation of G in time $O(\|\vec{G}\|)$. Let us assume $\vec{G}$ has signature σ. The functional formula $\eta(x,y) = \bigvee_{f \in \sigma_{fun}} f(x) = y \vee f(y) = x$ is true for some pair of vertices in $\vec{G}$ if and only if there is an edge between them in G. Instead of evaluating some relational formula on G, we can replace every edge relation $E(x,y)$ with $\eta(x,y)$ and evaluate the resulting functional formula on $\vec{G}$.

Similar to Kazana and Segoufin [17], we restrict ourselves to finding algorithms for classes of functional representations with bounded expansion. As discussed above (and in [17]), they also work for graph classes with bounded expansion.

Most of the time we will be using functional representations. To be less verbose (and when it is clear from the context), we will call functional signatures simply *signatures*, classes of functional representations simply *classes*.

3.3 Approximating Counting Terms using Positive Sums.

We will often deal with formulas in disjunctive normal form, i.e., a disjunction of conjunctions of literals. We will call the conjuncts often *conjunctive clauses* and sometimes only *clauses* when the exact meaning is clear form the context. An important technical tool in the upcoming proofs are special forms of conjunctive clauses that will be defined next. They are partially *complete* in the sense that they must contain certain atomic formulas or their negation. This completeness will force the value of other literals and allow us to remove them from the conjunctive clause, which is one simplification step of many more to come.

DEFINITION 3.2. *Let σ, ρ be signatures with $\sigma \subseteq \rho$. A conjunctive clause $\tau(y) \wedge \psi(\bar{x}) \wedge \Delta^{=}(y\bar{x}) \wedge \Delta^{\neq}(y\bar{x})$ with $\bar{x} = x_1, \dots, x_k$ is called a k-σ-ρ-*canonical conjunctive clause *if*

1. *$\tau(y) \in \mathrm{FO}[2,\rho]$ is a conjunctive clause that contains for every $f,g,h, \in \rho_{fun}$, either the literal $f(y) = h(g(y))$ or its negation.*
2. *$\psi(\bar{x}) \in \mathrm{FO}[2,\rho]$ is a conjunctive clause that contains for every $i,j \in \{1,\dots,k\}$ and $f,g,h, \in \rho_{fun}$ either the literal $f(x_i) = h(g(x_j))$ or its negation.*
3. *$\Delta^{=}(y\bar{x}) \in \mathrm{FO}[1,\sigma]$ is a nonempty conjunction of positive literals of the form $f(y) = g(x_i)$ with $f,g \in \sigma_{fun}$ and $i \in \{1,\dots,k\}$,*
4. *$\Delta^{\neq}(y\bar{x}) \in \mathrm{FO}[1,\sigma]$ is a conjunction of negative literals of the form $f(y) \neq g(x_i)$ with $f,g \in \sigma_{fun}$ and $i \in \{1,\dots,k\}$.*

We denote the set of all such canonical conjunctive clauses by $\mathbf{C}(k,\sigma,\rho)$.

We call the literals in $\Delta^{=}(y\bar{x})$ and $\Delta^{\neq}(y\bar{x})$ the *mixed* literals of a canonical conjunctive clause. The requirement that $\Delta^{=}(y\bar{x})$ is nonempty is a technical assumption that we will need later. In the following lemma the literal $f_{\mathrm{apx}}(y) = f_{\mathrm{apx}}(x_1)$ is needed to make sure this assumption is fulfilled. Over the course of this section we will gradually decompose a quantifier-free formula into more and more simple combinations of canonical conjunctive clauses.

LEMMA 3.2. *For two signatures σ, ρ with $\sigma \subseteq \rho$ and a given quantifier-free formula $\varphi(y\bar{x}) \wedge f_{\mathrm{apx}}(y) = f_{\mathrm{apx}}(x_1) \in \mathrm{FO}[1,\sigma]$ one can compute a set of canonical conjunctive clauses $\Omega \subseteq \mathbf{C}(|\bar{x}|,\sigma,\rho)$ such that for every $\vec{G} \in \mathcal{G}(\rho)$ and every tuple of vertices $v\bar{u} \in V(\vec{G})^{|y\bar{x}|}$*

$$\vec{G} \models \varphi(v\bar{u}) \wedge f_{\mathrm{apx}}(v) = f_{\mathrm{apx}}(u_1) \quad \textit{iff} \quad \vec{G} \models \omega(v\bar{u}) \textit{ for some } \omega \in \Omega.$$

Furthermore, Ω is mutually exclusive in the sense that for every $\vec{G} \in \mathcal{G}(\rho)$ and tuple $v\bar{u} \in V(\vec{G})^{|y\bar{x}|}$ there is at most one $\omega \in \Omega$ with $\vec{G} \models \omega(v\bar{u})$.

Proof. We can assume $\varphi(y\bar{x})$ to be given in disjunctive normal form. Consider a conjunctive clause $\omega(y\bar{x})$ of this normal form and any literal $l(y\bar{x})$. We can replace $\omega(y\bar{x})$ with two clauses $\omega(y\bar{x}) \wedge l(y\bar{x})$ and $\omega(y\bar{x}) \wedge \neg l(y\bar{x})$. The result is still a disjunctive normal form of $\varphi(y\bar{x})$. We can therefore assume that every clause of $\varphi(y\bar{x})$ contains

- for every valid literal l in $\mathrm{FO}[1,\sigma]$ with free variables from $y\bar{x}$ either l or $\neg l$,
- for every literal l of the form $h(f(y)) = g(y)$ or $h(f(x_i)) = g(x_j)$ with $f,g,h \in \rho_{fun}$ and $i,j \in \{1,\dots,|\bar{x}|\}$ either l or $\neg l$.

Let Ω be the set of conjunctive clauses of $\varphi(y\bar{x})$. Any two clauses in Ω disagree in at least one literal. Thus, they cannot be satisfied by the same interpretation. This means Ω is mutually exclusive. Furthermore, since Ω is obtained from a disjunctive normal form, for every $\vec{G} \in \mathcal{G}(\rho)$ and every $v\bar{u} \in V(\vec{G})^{|y\bar{x}|}$

$$\vec{G} \models \varphi(v\bar{u}) \text{ iff } \vec{G} \models \omega(v\bar{u}) \text{ for some } \omega \in \Omega.$$

However, the formulas in Ω are not yet canonical conjunctive clauses. We fix a clause from Ω and decompose it into four subclauses $\tau(y) \wedge \psi(\bar{x}) \wedge \Delta^{=}(y\bar{x}) \wedge \Delta^{\neq}(y\bar{x})$, where $\tau(y)$, $\psi(\bar{x})$ contain all literals depending on y and $\bar{x}$, and $\Delta^{=}(y\bar{x})$, $\Delta^{\neq}(y\bar{x})$ contain the remaining positive and negative literals, respectively.

The clauses $\tau(y)$, $\psi(\bar{x})$ are of the form mentioned in Definition 3.2, while $\Delta^{=}(y\bar{x})$, $\Delta^{\neq}(y\bar{x})$ might not. We will modify them to fit Definition 3.2. Besides the allowed literals, $\Delta^{=}(y\bar{x})$ may also contain literals of the form $f(x_i) = y$, $f(y) = x_i$ or $x_i = y$. Using the identify function f_{id}, we can artificially turn them into equivalent literals $f(x_i) = f_{\mathrm{id}}(y)$, $f(y) = f_{\mathrm{id}}(x_i)$, or $f_{\mathrm{id}}(x_i) = f_{\mathrm{id}}(y)$ of the allowed form. We proceed similarly for $\Delta^{\neq}(y\bar{x})$. We also add the literal $f_{\mathrm{apx}}(y) = f_{\mathrm{apx}}(x_1)$ to $\Delta^{=}(y\bar{x})$. Therefore, $\Delta^{=}(y\bar{x})$ is nonempty. Now all clauses are of the form stated in Definition 3.2. We apply this procedure to every clause in Ω. Then $\Omega \subseteq \mathbf{C}(|\bar{x}|, \sigma, \rho)$. Because we added $f_{\mathrm{apx}}(y) = f_{\mathrm{apx}}(x_1)$ to every canonical conjunctive clause, we have

$$\vec{G} \models \varphi(v\bar{u}) \wedge f_{\mathrm{apx}}(v) = f_{\mathrm{apx}}(u_1) \text{ iff } \vec{G} \models \omega(v\bar{u}) \text{ for some } \omega \in \Omega.$$

□

In the previous lemma, it would have been okay to discard unsatisfiable formulas from Ω. When we go from a functional structure $\vec{G}$ to its 1-transitive fraternal augmentation, new function symbols are inserted, representing transitive and fraternal arcs. We will later argue that it is okay also to remove those formulas which are not satisfied by any 1-transitive fraternal augmentation or extension thereof. Since all 1-transitive fraternal augmentations have a certain structure, we can discard more formulas. The next definition formally captures these concepts.

Definition 3.3. *Let σ, ρ be signatures with $\sigma \subseteq \rho$ and $\vec{G}' \in \mathcal{G}(\rho)$. We say $\vec{G}'$ is a σ-ρ-*expansion *if there exists $\vec{G} \in \mathcal{G}(\sigma)$ such that $\vec{G}'$ is an expansion of the 1-transitive fraternal augmentation of $\vec{G}$. We also say $\vec{G}'$ is a σ-ρ-expansion of $\vec{G}$. A canonical conjunctive clause $\omega(y\bar{x}) \in \mathbf{C}(|\bar{x}|, \sigma, \rho)$ is σ-ρ-*unsatisfiable *if $\vec{G}' \not\models \omega(v\bar{u})$ holds for every σ-ρ-expansion $\vec{G}' \in \mathcal{G}(\rho)$ and every $v\bar{u} \in V(\vec{G}')^{|y\bar{x}|}$.*

In the next step we further simplify the formulas by reducing the number of mixed positive literals from an arbitrary number down to one.

Lemma 3.3. *Let σ, ρ be signatures with $\sigma \subseteq \rho$ and $\omega(y\bar{x}) = \tau(y) \wedge \psi(\bar{x}) \wedge \Delta^{=}(y\bar{x}) \wedge \Delta^{\neq}(y\bar{x}) \in \mathbf{C}(|\bar{x}|, \sigma, \rho)$. There exists an algorithm that either computes a literal $f(y) = g(x_i) \in \Delta^{=}(y\bar{x})$ such that*

$$\omega(y\bar{x}) \equiv \tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge \Delta^{\neq}(y\bar{x})$$

or concludes that $\omega(y\bar{x})$ is σ-ρ-unsatisfiable.

Proof. By definition, $\Delta^{=}(y\bar{x})$ is nonempty. If it contains only one literal, we do not need to do anything. Let us assume there are two literals $f(y) = g(x_i)$ and $f'(y) = g'(x_j)$ in $\Delta^{=}(y\bar{x})$.

Let $\vec{G}'$ be a σ-ρ-expansion and $v \in V(\vec{G})$. Since $f, f' \in \sigma_{fun}$ and by the fraternal rule, there exists a function $h \in \rho_{fun}$ such that either $h^{\vec{G}}(f^{\vec{G}}(v)) = f'^{\vec{G}}(v)$ or $f^{\vec{G}}(v) = h^{\vec{G}}(f'^{\vec{G}}(v))$. Thus, if $\omega(y\bar{x})$ is σ-ρ-satisfiable then $\tau(y)$ either contains $h(f(y)) = f'(y)$ or $f(y) = h(f'(y))$ for some function $h \in \rho_{fun}$. Let us assume it is $h(f(y)) = f'(y)$ because the other case is similar. Then there must be $h(g(x_i)) = g'(x_j)$ present in $\tau(y)$ (or the formula is unsatisfiable and in particular σ-ρ-unsatisfiable). From $f'(y) = h(f(y))$, $h(g(x_i)) = g'(x_j)$, and $f(y) = g(x_i)$ follows $f'(y) = g'(x_j)$. Therefore we can remove $f'(y) = g'(x_j)$ from $\Delta^{=}(y\bar{x})$. We repeat this procedure as long as $\Delta^{=}(y\bar{x})$ contains at least two literals. □

The next simplification gets rid of some of the negative mixed literals, however, not all of them. The remaining ones have a special relation to the rest of the conjunctive clause that will cause them later on to have only a small influence on the counting term.

Lemma 3.4. *Let σ, ρ be signatures with $\sigma \subseteq \rho$ and $\omega(y\bar{x}) = \tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge \Delta^{\neq}(y\bar{x}) \in \mathbf{C}(|\bar{x}|, \sigma, \rho)$. We define $\Delta^{\neq}_{\tau}(y\bar{x})$ to be the set of all literals of the form $f'(y) \neq g'(x_j)$ in $\Delta^{\neq}(y\bar{x})$ such that $\tau(y)$ contains $h(f(y)) \neq f'(y)$ for all $h \in \rho_{fun}$. There exists an algorithm that concludes either that*

$$\omega(y\bar{x}) \equiv \tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge \Delta^{\neq}_{\tau}(y\bar{x})$$

or that $\omega(y\bar{x})$ is σ-ρ-unsatisfiable.

Proof. Let $f'(y) \neq g'(x_j)$ be a literal that is contained in $\Delta^{\neq}(y\bar{x})$, but not in $\Delta^{\neq}_{\tau}(y\bar{x})$. We argue that we can either safely remove it or the formula is unsatisfiable. By our assumption there exists $h \in \rho_{fun}$ such that $h(f(y)) \neq f'(y) \notin \tau(y)$. Since $\tau(y)$ is complete, $h(f(y)) = f'(y) \in \tau(y)$. Since $\psi(\bar{x})$ is also complete, it either contains the literal $h(g(x_i)) = g'(x_j)$ or its negation $h(g(x_i)) \neq g'(x_j)$.

Assume $h(g(x_i)) \neq g'(x_j) \in \psi(\bar{x})$. The literal $f(y) = g(x_i)$ with $h(f(y)) = f'(y)$ and $h(g(x_i)) \neq g'(x_j)$ implies $f'(y) \neq g'(x_j)$, so we can safely remove it from $\Delta^{\neq}(y\bar{x})$.

Assume that $h(g(x_i)) = g'(x_j) \in \psi(\bar{x})$. The literal $f(y) = g(x_i)$ with $h(f(y)) = f'(y)$ implies $h(g(x_i)) = f'(y)$. On the other hand, $f'(y) \neq g'(x_j)$ together with $h(g(x_i)) = g'(x_j)$ implies $h(g(x_i)) \neq f'(y)$. Henceforth, $\omega(y\bar{x})$ is unsatisfiable. ∎

The following lemma aggregates the results from Lemmas 3.2, 3.3, and 3.4.

LEMMA 3.5. *For two signatures σ, ρ with $\sigma \subseteq \rho$ and a given quantifier-free formula $\varphi(y\bar{x}) \wedge f_{\text{apx}}(y) = f_{\text{apx}}(x_1) \in \text{FO}[1, \sigma]$ one can compute a set of canonical conjunctive clauses $\Omega \subseteq \mathbf{C}(|\bar{x}|, \sigma, \rho)$ with the following properties:*

1. *Every formula $\omega \in \Omega$ has the form $\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge \Delta_\tau^{\neq}(y\bar{x})$. We require for every literal of the form $f'(y) \neq g'(x_j)$ in $\Delta_\tau^{\neq}(y\bar{x})$ and every $h \in \rho_{fun}$ that $h(f(y)) \neq f'(y) \in \tau(y)$.*
2. *For every σ-ρ-expansion $\vec{G}$ and every tuple $v\bar{u} \in V(\vec{G})^{|y\bar{x}|}$ holds*

$$\vec{G} \models \varphi(v\bar{u}) \wedge f_{\text{apx}}(v) = f_{\text{apx}}(u_1) \quad \textit{iff} \quad \vec{G} \models \omega(v\bar{u}) \textit{ for some } \omega \in \Omega.$$

3. *Ω is mutually exclusive in the sense that for every σ-ρ-expansion $\vec{G}$ and every tuple $v\bar{u} \in V(\vec{G})^{|y\bar{x}|}$ there is at most one $\omega \in \Omega$ with $\vec{G} \models \omega(v\bar{u})$.*

Proof. Let $\Omega \subseteq \mathbf{C}(|\bar{x}|, \sigma, \rho)$ be the set computed by Lemma 3.2. This set already satisfies properties 2 and 3. We will modify it such that it also satisfies the first property. Let $\omega(y\bar{x}) \in \Omega$. We first apply the algorithm from Lemma 3.3 and then (assuming it was not concluded that $\omega(y\bar{x})$ is σ-ρ-unsatisfiable) the one from 3.4. This either yields a formula

$$\omega(y\bar{x}) \equiv \tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge \Delta_\tau^{\neq}(y\bar{x})$$

that meets the requirements of the first property or concludes that $\omega(y\bar{x})$ is σ-ρ-unsatisfiable. If the formula is σ-ρ-unsatisfiable, we can remove if from Ω and properties 2 and 3 remain true. We apply this procedure for every formula in Ω. Then property 1 is also satisfied. ∎

The following lemma shows that we can evaluate a simple counting term of the form $\#y\,\tau(y) \wedge f(y) = u \wedge f'(y) = u'$ for all values of u and u' in linear time. Since there are a quadratic number of tuples u, u', we only write down those tuples where the counting term is non-zero. Later we will need these numbers for an algorithm that can identify places where we need to add arcs to the functional structure in order to be able to eliminate certain negative mixed literals that have a non-negligible contribution to the value of a counting term. We will need the same lemma also in the proof that *exact* counting is possible for formulas of a certain shape.

LEMMA 3.6. *Let $\mathcal{C} \subseteq \mathcal{G}(\sigma)$ be a class with bounded expansion, $\tau(y) \in \text{FO}[2, \sigma]$ and $f, f' \in \sigma_{fun}$. For an input $\vec{G} \in \mathcal{C}$ the list of triples*

$$\{\, (u, u', c) \mid u, u' \in V(\vec{G}), \; c = [\![\#y\, \tau(y) \wedge f(y) = u \wedge f'(y) = u']\!]^{\vec{G}}, c > 0 \,\}$$

can be computed in time $O(\|\vec{G}\|)$.

Proof. We define a "counter" $c(u, u')$ for $u, u' \in V(\vec{G})$ as

$$c(u, u') = \big|\{\, v \in V(\vec{G}) \mid \vec{G} \models \tau(v) \wedge f(v) = u \wedge f'(v) = u' \,\}\big|.$$

The following algorithm obviously computes $c(u, u')$.

for $v \in V(\vec{G})$ with $\vec{G} \models \tau(v)$ **do**
  $c(f(v), f'(v)) \leftarrow c(f(v), f'(v)) + 1$
end for

Computing $c(u, u')$ for all $u, u' \in V(\vec{G})$ would be easy in quadratic time, but is also possible in linear time. Kazana and Segoufin showed that we can enumerate all vertices v with $G \models \tau(v)$ in time linear in $\|\vec{G}\|$ [17]. The only issue with this short piece of code is how to store the counters $c(u, u')$. There is a quadratic number of them, although most of them are left to be zero and we are only interested in those with a positive count. One possibility is delaying the increment of the counters to the end. Instead of carrying out the commands $c(f(v), f'(v)) \leftarrow c(f(v), f'(v)) + 1$ immediately we store them in an array of linear length. At the end we can sort this array in linear time (e.g., by a combination of radix- and bucket-sort) and then combine blocks of identical commands while counting their sizes. What remains is a list of the positive counters together with their respective values. ∎

The next step is to actually compute an expansion that is prepared in such a way that every mixed negative literal from every possible relevant conjunctive clause that influences the underlying counting term significantly can be removed. While Lemma 3.7 does the preparation by adding flip arcs, Lemma 3.8 shows that the resulting expansion has the desired property.

LEMMA 3.7. *Let $\mathcal{C} \subseteq \mathcal{G}(\sigma)$ be a class with bounded expansion and $\varepsilon > 0$. There exists a signature $\rho \supseteq \sigma$ and*

a class $\mathcal{C}' \subseteq \mathcal{G}(\rho)$ with bounded expansion such that for every $\vec{G} \in \mathcal{C}$ one can compute a σ-ρ-expansion $\vec{G}' \in \mathcal{C}'$ of $\vec{G}$ with the following property in time $O(\|\vec{G}\|)$: If there exist $u, u' \in V(\vec{G}')$, a quantifier-free formula $\tau(y) \in \mathrm{FO}[2,\sigma]$, and f, $f' \in \sigma_{fun}$ such that $[\![\#y\, \tau(y) \wedge f(y) = u \wedge f'(y) = u']\!]^{\vec{G}'} > \varepsilon [\![\#y\, \tau(y) \wedge f(y) = u]\!]^{\vec{G}'}$ then there exists $h \in \rho_{fun}$ with $h^{\vec{G}'}(u) = u'$ (i.e., $\vec{G}'$ contains an arc $u'u$).

Proof. We have to show that we can identify all pairs u', u that need a new arc in linear time and that the resulting functional representation belongs to a class with bounded expansion. We start with the second task.

We will express $\vec{G}'$ by first doing a 1-transitive fraternal augmentation on $\vec{G}$ and then adding all necessary remaining arcs using a bounded number of flip augmentations. By Corollary 3.1, $\vec{G}'$ then belongs to a class with bounded expansion.

Let $u, u' \in V(\vec{G})$ such that there needs to be an arc $u'u$ in $\vec{G}'$. This means in particular that $[\![\#y\, \tau(y) \wedge f(y) = u \wedge f'(y) = u']\!]^{\vec{G}'} > 0$, i.e., there is a vertex $v \in V(\vec{G})$ such that the arcs $u'v$ and uv are present in $\vec{G}$. By the fraternal rule, either the arc $u'u$ or uu' exists in the 1-transitive fraternal augmentation of $\vec{G}$.

If the arc $u'u$ is already present in the 1-transitive fraternal augmentation, we do not need to do anything. But if only the arc uu' is there, we need to flip it. Since $\mathcal{C}$ is a class of bounded expansion, there must be a constant bound on the indegree of all graphs in $\mathcal{C}$. Let us say this bound is d. We only stay within a class with bounded expansion if the indegree remains bounded after flipping the arcs. We fix a vertex $u \in V(\vec{G})$ and show that only a constant number of edges need to be oriented towards u. The set $U' = \{\, u' \in V(\vec{G}) \mid [\![\#y\, \tau(y) \wedge f(y) = u \wedge f'(y) = u']\!]^{\vec{G}} > \varepsilon [\![\#y\, \tau(y) \wedge f(y) = u]\!]^{\vec{G}} \,\}$ contains all vertices u' for which an arc $u'u$ needs to be present in $\vec{G}'$. Let $A = \{\, v \in V(\vec{G}) \mid \vec{G} \models \tau(v) \wedge f(v) = u \,\}$. For every $u' \in U'$ there are at least $\varepsilon|A|$ arcs going from u' to A in $\vec{G}$. Therefore, there are at least $\varepsilon|A| \cdot |U'|$ many arcs going into the set A and there must be one vertex in A that receives at least $\varepsilon|U'|$ of them. As this number must be smaller than d, we can conclude that $|U'| \leq d/\varepsilon$. Hence, $\vec{G}'$ can be obtained using a 1-transitive fraternal augmentation and at most d/ε many flip operations.

It remains to be shown that this construction can be carried out in linear time. There is only a constant number of combinations of f, f', τ because $|\sigma|$ and the number of non-equivalent quantifier-free formulas in $\mathrm{FO}[2,\sigma]$ are constant. For each such combination we can compute the lists

$$l_1 = \{\, (u, u', c) \mid u, u' \in V(\vec{G}), \\ c = [\![\#y\, \tau(y) \wedge f(y) = u \wedge f'(y) = u']\!]^{\vec{G}}, c > 0 \,\},$$

$$l_2 = \{\, (u, c) \mid u \in V(\vec{G}), \\ c = [\![\#y\, \tau(y) \wedge f(y) = u]\!]^{\vec{G}}, c > 0 \,\}$$

in linear time by Lemma 3.6. Using l_1 and l_2 it is easy to determine all u', u with $[\![\#y\, \tau(y) \wedge f(y) = u \wedge f'(y) = u']\!]^{\vec{G}} > \varepsilon [\![\#y\, \tau(y) \wedge f(y) = u]\!]^{\vec{G}}$. ∎

LEMMA 3.8. *Let $\mathcal{C} \subseteq \mathcal{G}(\sigma)$ be a class with bounded expansion and $\varepsilon > 0$. There exists a signature $\rho \supseteq \sigma$ and a class $\mathcal{C}' \subseteq \mathcal{G}(\rho)$ with bounded expansion such that for every $\vec{G} \in \mathcal{C}$ one can compute a σ-ρ-expansion $\vec{G}' \in \mathcal{C}'$ of $\vec{G}$ with the following property in time $O(\|\vec{G}\|)$:*

For every formula $\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge f'(y) = g'(x_j) \in \mathbf{C}(|\bar{x}|, \sigma, \rho)$ with $h(f(y)) \neq f'(y) \in \tau(y)$ for all $h \in \rho_{fun}$ and for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$ holds

$$[\![\#y\, \tau(y) \wedge \psi(\bar{u}) \wedge f(y) = g(u_i) \wedge f'(y) = g'(u_j)]\!]^{\vec{G}'} \leq \\ \varepsilon [\![\#y\, \tau(y) \wedge \psi(\bar{u}) \wedge f(y) = g(u_i)]\!]^{\vec{G}'}.$$

Proof. For a given input $\vec{G}$, we construct $\vec{G}' \in \mathcal{G}(\rho)$ according to Lemma 3.7. Consider a formula $\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge f'(y) = g'(x_j)$ and a tuple $\bar{u} \in V(\vec{G})^{|\bar{x}|}$ as specified above. For convenience we define $u = g(u_i)$ and $u' = g'(u_j)$. We assume there exists $v \in V(\vec{G}')$ such that $\vec{G}' \models \tau(v) \wedge \psi(\bar{u}) \wedge f(v) = u \wedge f'(v) = u'$, since otherwise the left-hand side of the equation is zero and the statement is trivially true. For every $h \in \rho_{fun}$ we have $h(f(y)) \neq f'(y) \in \tau(y)$. This implies $h^{\vec{G}'}(u) = h^{\vec{G}'}(f^{\vec{G}'}(v)) \neq f'^{\vec{G}'}(v) = u'$ for every $h \in \rho_{fun}$. According to Lemma 3.7,

$$[\![\#y\, \tau(y) \wedge f(y) = u \wedge f'(y) = u']\!]^{\vec{G}'} \leq \\ \varepsilon [\![\#y\, \tau(y) \wedge f(y) = u]\!]^{\vec{G}'},$$

since otherwise $h^{\vec{G}'}(u) = u'$ for some $h \in \rho_{fun}$. The result follows since $\vec{G}' \models \psi(\bar{u})$. ∎

The following lemma presents a way to reduce the functional depth of a formula. It is taken from [17] with the notation changed to ours. While [17] does not mention that the formula with reduced depth can be computed, it clearly follows from their construction.

PROPOSITION 3.1. (LEMMA 4, [17]) *For a quantifier-free formula $\varphi(\bar{x})$ with signature σ and class $\mathcal{C} \in \mathcal{G}(\sigma)$ with bounded expansion there exists a signature $\rho \supseteq \sigma$, a class $\mathcal{C}' \subseteq \mathcal{G}(\rho)$ with bounded expansion and a quantifier-free formula $\varphi'(\bar{x}) \in \mathrm{FO}[1,\rho]$ such that the following properties are satisfied:*

- *The formula φ' can be computed from φ.*
- *For every $\vec{G} \in \mathcal{C}$ we can compute in time $O(\|\vec{G}\|)$ an expansion $\vec{G}' \in \mathcal{C}'$ of $\vec{G}$ such that for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$ holds $\vec{G} \models \varphi(\bar{u})$ iff $\vec{G}' \models \varphi'(\bar{u})$.*

While ignoring a single negative mixed literal is now guaranteed to change the value of a counting term only by a little, the next Lemma helps to estimate the influence of removing all negative mixed literals at once.

LEMMA 3.9. *Let $\omega(y), l_1(y), \dots, l_t(y)$ be formulas, $\vec{G}$ be a functional representation, and $\varepsilon > 0$. If for every $i \in \{1, \dots, t\}$ with $\varepsilon' = \min(1, \varepsilon)/2t$*

$$[\![\#y\, \omega(y) \wedge l_i(y)]\!]^{\vec{G}} \leq \varepsilon' [\![\#y\, \omega(y)]\!]^{\vec{G}}$$

then

$$[\![\#y\, \omega(y) \wedge \bigwedge_{i=1}^{t} \neg l_i(y)]\!]^{\vec{G}} \leq [\![\#y\, \omega(y)]\!]^{\vec{G}} \leq (1+\varepsilon)[\![\#y\, \omega(y) \wedge \bigwedge_{i=1}^{t} \neg l_i(y)]\!]^{\vec{G}}.$$

Proof. Every assignment $v \in V(\vec{G})$ for y satisfies either $\bigwedge_{i=1}^{t} \neg l_i(v)$ or $l_i(v)$ for at least one $i \in \{1, \dots, t\}$. Therefore

$$[\![\#y\, \omega(y)]\!]^{\vec{G}} \leq [\![\#y\, \omega(y) \wedge \bigwedge_{i=1}^{t} \neg l_i(y)]\!]^{\vec{G}} + \sum_{i=1}^{t} [\![\#y\, \omega(y) \wedge l_i(y)]\!]^{\vec{G}}.$$

Using our initial assumption, this means

$$[\![\#y\, \omega(y)]\!]^{\vec{G}} \leq [\![\#y\, \omega(y) \wedge \bigwedge_{i=1}^{t} \neg l_i(y)]\!]^{\vec{G}} + t\varepsilon' [\![\#y\, \omega(y)]\!]^{\vec{G}},$$

and thus

$$(1 - t\varepsilon')[\![\#y\, \omega(y)]\!]^{\vec{G}} \leq [\![\#y\, \omega(y) \wedge \bigwedge_{i=1}^{t} \neg l_i(y)]\!]^{\vec{G}} \leq [\![\#y\, \omega(y)]\!]^{\vec{G}}.$$

Let $a, b > 0$ with $(1 - t\varepsilon')a \leq b \leq a$. Dividing by a yields $1 - t\varepsilon' \leq b/a \leq 1$, and then taking the reciprocal gives us $1 \leq a/b \leq 1/(1 - t\varepsilon') = 1/(1 - \min(1, \varepsilon)/2) \leq 1 + \varepsilon$. We multiply with b and obtain $b \leq a \leq (1+\varepsilon)b$. This yields the statement of this lemma. ∎

We finally arrived at the point where we can approximate a counting term with a sum of simple terms of the form $\#y\, \tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i)$. While they still have many free variables, $\psi(\bar{x})$ does not depend on y and can therefore be pulled outside the counting quantifier. This leaves a counting term with a single free variable that we can evaluate using Lemma 3.6 in linear time.

LEMMA 3.10. *Let $\mathcal{C} \subseteq \mathcal{G}(\sigma)$ be a graph class with bounded expansion and $\varepsilon > 0$. For every quantifier-free formula $\varphi(y\bar{x})$ with signature σ one can compute a signature $\rho \supseteq \sigma$ and a set of conjunctive clauses $\Omega \subseteq \mathrm{FO}[2, \rho]$ of the form $\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i)$ with the following property:*

There exists a class $\mathcal{C}' \subseteq \mathcal{G}(\rho)$ with bounded expansion such that for every $\vec{G} \in \mathcal{C}$ one can compute in time $O(\|\vec{G}\|)$ an expansion $\vec{G}' \in \mathcal{C}'$ of $\vec{G}$ such that for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$

$$[\![\#y\, \varphi(y\bar{u})]\!]^{\vec{G}} \leq \sum_{\omega \in \Omega} [\![\#y\, \omega(y\bar{u})]\!]^{\vec{G}'} \leq (1+\varepsilon)[\![\#y\, \varphi(y\bar{u})]\!]^{\vec{G}}.$$

Proof. Assume we are given a formula $\varphi(y\bar{x})$ and a functional representation $\vec{G} \in \mathcal{C}$. We pick a vertex $w \in V(\vec{G})$ and add a new function symbol f_{apx} to our signature σ and input structure with $f_{\mathrm{apx}}^{\vec{G}}(v) = w$ for every $v \in V(\vec{G})$. For the underlying directed graph of $\vec{G}$, this amounts to making w an apex vertex. Thus by Definition 3.1, $\vec{G}$ still comes from a class with bounded expansion.

Using Proposition 3.1 we can replace $\varphi(y\bar{x})$ with another quantifier-free formula with functional depth one. The price we have to pay is replacing $\vec{G}$ with an expanded functional representation, which is still from a class with bounded expansion. Both the new formula and the new functional representation can be computed in the desired time. Therefore, from now on, we can assume without loss of generality that $\varphi(y\bar{x})$ is a quantifier-free formula with signature σ and functional depth one and that there exists a function symbol $f_{\mathrm{apx}} \in \sigma_{fun}$ with $f_{\mathrm{apx}}^{\vec{G}}(v) = f_{\mathrm{apx}}^{\vec{G}}(v')$ for every $v, v' \in V(\vec{G})$.

Lemma 3.8 with $\varepsilon' = \min(1, \varepsilon)/2|\sigma|^2|\bar{x}|$ gives us a signature ρ and a bounded expansion class $\mathcal{C}' \subseteq \mathcal{G}(\rho)$. We compute in time $O(\|\vec{G}\|)$ the σ-ρ-expansion $\vec{G}' \in \mathcal{C}'$ of $\vec{G}$ with properties as in Lemma 3.8. Let $\Omega' \subseteq \mathbf{C}(|\bar{x}|, \sigma, \rho)$ be the set of canonical conjunctive clauses obtained from $\varphi(y\bar{x}) \wedge f_{\mathrm{apx}}(y) = f_{\mathrm{apx}}(x_1)$ with the algorithm from Lemma 3.5. Property 2 and 3 of Lemma 3.5 together imply for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$

$$[\![\#y\, \varphi(y\bar{u})]\!]^{\vec{G}} = [\![\#y\, \varphi(y\bar{u}) \wedge f_{\mathrm{apx}}^{\vec{G}}(y) = f_{\mathrm{apx}}^{\vec{G}}(u_1)]\!]^{\vec{G}} = \sum_{\omega' \in \Omega'} [\![\#y\, \omega'(y\bar{u})]\!]^{\vec{G}'}. \tag{3.3}$$

We consider a clause $\omega'(y\bar{x}) = \tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \wedge \Delta_\tau^{\neq}(y\bar{x}) \in \Omega'$, where $\Delta_\tau^{\neq}(y\bar{x})$ is of the

form $\bigwedge_{i=1}^{t} \neg l_i(y\bar{x})$. Lemma 3.8 states (using the first property of Lemma 3.5) for $1 \leq i \leq t$ and $\bar{u} \in V(\vec{G})^{|\bar{x}|}$

$$[\![\#y\, \tau(y) \wedge \psi(\bar{u}) \wedge f(y) = g(u_i) \wedge l_i(y\bar{u})]\!]^{\vec{G}'} \leq \varepsilon' [\![\#y\, \tau(y) \wedge \psi(\bar{u}) \wedge f(y) = g(u_i)]\!]^{\vec{G}'}.$$

Since the literals of $\Delta_\tau^{\neq}(y\bar{x})$ have the form $f(y) \neq g(x_i)$ with $f, g \in \sigma_{fun}$, we have $t \leq |\sigma|^2|\bar{x}|$. Hence, by Lemma 3.9,

(3.4)
$$[\![\#y\, \omega'(y\bar{u})]\!]^{\vec{G}'} \leq [\![\#y\, \tau(y) \wedge \psi(\bar{u}) \wedge f(y) = g(u_i)]\!]^{\vec{G}'} \leq (1+\varepsilon)[\![\#y\, \omega'(y\bar{u})]\!]^{\vec{G}'}.$$

Combining (3.3) and (3.4) yields our result. ∎

The last step to reach this section's goal is to actually evaluate the simple counting terms in the sum of the last lemma and store the results as "weights" at the individual vertices. Finally, this allows us to approximate a counting term $\#y\, \varphi(y\bar{u})$ using combinations of quantifier-free first-order formulas and the calculated weights. We consider the following theorem the main technical contribution of this paper, as the whole upcoming quantifier elimination procedure builds upon it. Since negative summands lead to the problem of cancellation and therefore bad approximations, a lot of effort has been spent in this section to make sure that none of the summands $c_{\omega,i}(u_i)$ are negative. A similar result with negative summands is easier to prove and can be found in the full version of this paper [5].

THEOREM 3.1. *Let $\mathcal{C} \subseteq \mathcal{G}(\sigma)$ be a class with bounded expansion and $\varepsilon > 0$. One can compute for every quantifier-free formula $\varphi(y\bar{x})$ with signature σ a set of conjunctive clauses Ω with free variables $\bar{x}$ and signature $\rho \supseteq \sigma$ that satisfies the following property:*

There exists a class $\mathcal{C}' \subseteq \mathcal{G}(\rho)$ with bounded expansion such that for every $\vec{G} \in \mathcal{C}$ one can compute in time $O(\|\vec{G}\|)$ an expansion $\vec{G}' \in \mathcal{C}'$ of $\vec{G}$ and functions $c_{\omega,i}(v) \colon V(\vec{G}) \to \mathbf{N}$ for $\omega \in \Omega$ and $i \in \{1, \ldots, |\bar{x}|\}$ such that for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$ there exists exactly one formula $\omega \in \Omega$ with $\vec{G}' \models \omega(\bar{u})$. For this formula

$$[\![\#y\, \varphi(y\bar{u})]\!]^{\vec{G}} \leq \sum_{i=1}^{|\bar{x}|} c_{\omega,i}(u_i) \leq (1+\varepsilon)[\![\#y\, \varphi(y\bar{u})]\!]^{\vec{G}}.$$

Proof. We use Lemma 3.10 to construct $\Omega' \subseteq \mathrm{FO}[2, \rho]$ and $\vec{G}' \in \mathcal{C}' \subseteq \mathcal{G}(\rho)$ such that for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$

$$[\![\#y\, \varphi(y\bar{u})]\!]^{\vec{G}} \leq \sum_{\omega' \in \Omega'} [\![\#y\, \omega'(y\bar{u})]\!]^{\vec{G}'} \leq (1+\varepsilon)[\![\#y\, \varphi(y\bar{u})]\!]^{\vec{G}}. \tag{3.5}$$

Let $\Omega \subseteq \mathrm{FO}[2, \rho]$ be the set of all complete conjunctive clauses with functional depth two, signature ρ and free variables $\bar{x}$. This set has two important properties: First, for every $\bar{u} \in V(G)^{|\bar{x}|}$ there exists exactly one $\omega \in \Omega$ with $\vec{G}' \models \omega(\bar{u})$. Secondly, for every $\omega \in \Omega$ and conjunctive clause $\psi(\bar{x}) \in \mathrm{FO}[2, \rho]$ either $\omega \models \psi$ or $\omega \models \neg\psi$.

Let now $\bar{u} \in V(\vec{G})^{|\bar{x}|}$, $\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \in \Omega'$, and $\omega \in \Omega$ such that $\vec{G}' \models \omega(\bar{u})$. If $\omega \models \neg\psi$ then $[\![\#y\, \tau(y) \wedge \psi(\bar{u}) \wedge f(y) = g(u_i)]\!]^{\vec{G}'} = 0$. If $\omega \models \psi$ then $[\![\#y\, \tau(y) \wedge \psi(\bar{u}) \wedge f(y) = g(u_i)]\!]^{\vec{G}'} = [\![\#y\, \tau(y) \wedge f(y) = g(u_i)]\!]^{\vec{G}'}$. Using this observation, we define for every $\omega \in \Omega$ and $i \in \{1, \ldots, |\bar{x}|\}$ a set $\Gamma_{\omega,i}$ by iterating over all formulas $\omega \in \Omega$ and $\tau(y) \wedge \psi(\bar{x}) \wedge f(y) = g(x_i) \in \Omega'$ and adding $\tau(y) \wedge f(y) = g(x_i)$ to $\Gamma_{\omega,i}$ if $\omega \models \psi$. Now for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$ there exists exactly one formula $\omega \in \Omega$ with $\vec{G}' \models \omega(\bar{u})$, and for this formula

$$\sum_{\omega' \in \Omega'} [\![\#y\, \omega'(y\bar{u})]\!]^{\vec{G}'} = \sum_{i=1}^{|\bar{x}|} \sum_{\tau(y) \wedge f(y) = g(x_i) \in \Gamma_{\omega,i}} [\![\#y\, \tau(y) \wedge f(y) = g(u_i)]\!]^{\vec{G}'}. \tag{3.6}$$

Let us fix one set $\Gamma_{\omega,i}$. For every formula $\tau(y) \wedge f(y) = g(x_i) \in \Gamma_{\omega,i}$ we use Lemma 3.6 to construct a function c with $c(v) = [\![\#y\, \tau(y) \wedge f(y) = g(v)]\!]^{\vec{G}'}$. Let $c_{\omega,i}$ be the sum over all such functions for formulas in $\Gamma_{\omega,i}$. Then

$$c_{\omega,i}(u_i) = \sum_{\tau(y) \wedge f(y) = g(x_i) \in \Gamma_{\omega,i}} [\![\#y\, \tau(y) \wedge f(y) = g(u_i)]\!]^{\vec{G}'}. \tag{3.7}$$

Combining (3.5), (3.6) and (3.7) yields our statement. ∎

3.4 Constructing an Approximate Quantifier-Free Formula.

Theorem 3.1 approximates a term $\#y\, \varphi$ using functions $c_{\omega,i}(v)$ that assign each vertex a number. In the following lemma, we round $c_{\omega,i}(v)$ into a finite number of intervals (with a granularity depending on ε) and extend the underlying structure with unary predicates that encode for each vertex v what interval $c_{\omega,i}(v)$ lies in. Using these predicates, we build a quantifier-free formula φ, approximating $\#y\, \varphi > N$. This starts the quantifier elimination step we will later use to build our approximation scheme.

LEMMA 3.11. *Let $\mathcal{C} \subseteq \mathcal{G}(\sigma)$ be a class with bounded expansion and $\varepsilon > 0$. For every quantifier-free formula $\varphi(y\bar{x})$ with signature σ one can compute a quantifier-free formula $\varphi'(\bar{x})$ with signature $\rho \supseteq \sigma$ and the following property:*

There exists another class $\mathcal{C}' \subseteq \mathcal{G}(\rho)$ with bounded expansion and for every $\vec{G} \in \mathcal{C}$ and $N \in \mathbf{Z}$ one can compute in time $O(\|\vec{G}\|)$ an expansion $\vec{G}' \in \mathcal{C}'$ of $\vec{G}$ such that for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$:

- *If $\vec{G}' \models \varphi'(\bar{u})$ then $\vec{G} \models \#y\,\varphi(y\bar{u}) > N$.*
- *If $\vec{G}' \not\models \varphi'(\bar{u})$ then $\vec{G} \models \#y\,\varphi(y\bar{u}). \leq (1+\varepsilon)N$.*

Proof. Let $\vec{G} \in \mathcal{C} \subseteq \mathcal{G}(\sigma)$ and $\varphi(y\bar{x})$ be a quantifier-free formula with signature σ. We can assume that $-1 \leq N \leq |\vec{G}|$ because we are counting the size of a vertex set. By Theorem 3.1 we can compute a signature $\rho^* \supseteq \sigma$, a set of conjunctive clauses Ω with signature ρ^*, an expansion $\vec{G}^*$ of $\vec{G}$ and functions $c_{\omega,i}$ such that for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$

$$\begin{aligned}(3.8)\quad [\![\#y\,\varphi(y\bar{u})]\!]^{\vec{G}} &\leq \sum_{\omega\in\Omega} [\![\omega(\bar{u})]\!]^{\vec{G}^*} \sum_{i=1}^{|\bar{x}|} c_{\omega,i}(u_i) \\ &\leq (1+\varepsilon/2)[\![\#y\,\varphi(y\bar{u})]\!]^{\vec{G}}.\end{aligned}$$

The set Ω is such that for every $\bar{u} \in \vec{G}^{|\bar{x}|}$ there is at most one $\omega \in \Omega$ with $\vec{G} \models \omega(\bar{u})$. Furthermore $\vec{G}^*$ comes from a class $\mathcal{C}^* \subseteq \mathcal{G}(\rho^*)$ with bounded expansion, and $\vec{G}^*$ as well as the functions $c_{\omega,i}$ can be computed in time linear in $\|\vec{G}\|$. We define a step size $s = N\varepsilon/2|\bar{x}|$ and $c'_{\omega,i}(v) = s\lfloor c_{\omega,i}(v)/s \rfloor$. We get $c'_{\omega,i}(v)$ if we round $c_{\omega,i}(v)$ down to the next multiple of s. Therefore $c'_{\omega,i}(v) \leq c_{\omega,i}(v) \leq s + c'_{\omega,i}(v)$. For every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$ we get

$$\begin{aligned}[\![\#y\,\varphi(y\bar{u})]\!]^{\vec{G}} - |\bar{x}|s &\leq \sum_{\omega\in\Omega} [\![\omega(\bar{u})]\!]^{\vec{G}^*} \sum_{i=1}^{|\bar{x}|} c'_{\omega,i}(u_i) \\ &\leq (1+\varepsilon/2)[\![\#y\,\varphi(y\bar{u})]\!]^{\vec{G}}\end{aligned}$$

and because of $|\bar{x}|s = N\varepsilon/2$ also

$$(3.9)\quad [\![\#y\,\varphi(y\bar{u})]\!]^{\vec{G}} \leq N \implies \sum_{\omega\in\Omega}\sum_{i=1}^{|\bar{x}|} c'_{\omega,i}(u_i) \leq (1+\varepsilon/2)N,$$

$$(3.10)\quad [\![\#y\,\varphi(y\bar{u})]\!]^{\vec{G}} > (1+\varepsilon)N \implies \sum_{\omega\in\Omega} [\![\omega(\bar{u})]\!]^{\vec{G}^*} \sum_{i=1}^{|\bar{x}|} c'_{\omega,t}(u_i) > (1+\varepsilon/2)N.$$

From now on, we will construct an expansion $\vec{G}'$ of $\vec{G}^*$ and a quantifier-free first-order formula $\varphi'(\bar{x})$ such that for $N' = (1+\varepsilon/2)N$ and every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$

$$(3.11)\quad \vec{G}' \models \varphi'(\bar{u}) \text{ iff } \sum_{\omega\in\Omega} [\![\omega(\bar{u})]\!]^{\vec{G}^*} \sum_{i=1}^{|\bar{x}|} c'_{\omega,i}(u_i) > N'.$$

Using (3.9), (3.10) we see that this is sufficient to prove this lemma. We choose $l_{\max} \in \mathbf{N}$ independently of N such that $sl_{\max} \geq N'$. We define $\vec{G}'$ to be the structure obtained from $\vec{G}^*$ by adding the following unary relations. For $\omega \in \Omega$, $i \in \{1,\dots,|\bar{x}|\}$, $l \in \{0,\dots,l_{\max}\}$ we add the relations

$$\begin{aligned}R^l_{\omega,i} &= \{\, v \in V(\vec{G}) \mid c'_{\omega,i}(v) = sl \,\}, \\ R^{\max}_{\omega,i} &= \{\, v \in V(\vec{G}) \mid c'_{\omega,i}(v) > sl_{\max} \,\}.\end{aligned}$$

(For these relations we will not distinguish between the relations themselves in the structure $\vec{G}'$ and the corresponding relational symbols in the signature.) The number of such relations is independent of N and therefore $\vec{G}' \in \mathcal{G}(\rho)$ for a signature $\rho \supseteq \rho^*$ whose size depends only on ρ^* and ε. Since $\vec{G}'$ and $\vec{G}^*$ have the same underlying graph, $\vec{G}'$ also belongs to a class $\mathcal{C}' \subseteq \mathcal{G}(\rho)$ with bounded expansion. Furthermore, the relations $R^l_{\omega,i}$ and $R^{\max}_{\omega,i}$ can be computed in time $O(\|\vec{G}^*\|)$ from $\vec{G}^*$ by evaluating and comparing each value $c'_{\omega,i}(v)$, $v \in V(\vec{G})$ in constant time.

The values of all functions $c'_{\omega,i}$ are always multiples of s. Therefore, knowing the relations $R^l_{\omega,t}$ for $0 \leq l \leq l_{\max}$ and $R^{\max}_{\omega,t}$ for a vertex $v \in V(\vec{G})$ gives us either the exact value of $c'_{\omega,t}(v)$ or indicates that $c'_{\omega,t}(v) > l_{\max}$. With the help of those predicates we now define formulas. Let

$$\varphi'_1(\bar{x}) = \bigvee_{\omega\in\Omega, i\in\{1,\dots,|\bar{x}|\}} \omega(\bar{x}) \wedge R^{\max}_{\omega,i}(x_i).$$

Observe that $\vec{G}' \models \varphi'_1(\bar{u})$ iff there exists $\omega \in \Omega$, $i \in \{1,\dots,|\bar{x}|\}$ such that $[\![\omega(\bar{u})]\!]^{\vec{G}^*} c'_{\omega,i}(u_i) > sl_{\max} \geq N'$. Let further

$$L = \Big\{\, (l_1,\dots,l_{|\bar{x}|}) \in \{0,\dots,l_{\max}\}^{|\bar{x}|} \;\Big|\; \sum_{i=1}^{|\bar{x}|} l_i s > N \,\Big\}$$

be the set of tuples whose sum is larger than N when multiplied with s. We define

$$\varphi'_2(\bar{x}) = \bigvee_{\omega\in\Omega} \Big(\omega(\bar{x}) \wedge \bigvee_{(l_1,\dots,l_{|\bar{x}|})\in L} \bigwedge_{i=1}^{|\bar{x}|} R^{l_i}_{\omega,i}(x_i)\Big)$$

and $\varphi'(\bar{x}) = \varphi'_1(\bar{x}) \vee \varphi'_2(\bar{x})$. We prove (3.11) by a case distinction. First, assume $\vec{G}' \models \varphi'_1(\bar{u})$. Then also $\vec{G}' \models \varphi'(\bar{u})$ and, as shown above, there exists $\omega \in \Omega$, $i \in \{1,\dots,|\bar{x}|\}$ such that $[\![\omega(\bar{u})]\!]^{\vec{G}^*} c'_{\gamma,i}(u_i) > N'$. This implies the right hand side of (3.11). Next assume that $\vec{G}' \not\models \varphi'_1(\bar{u})$. Then $[\![\omega(\bar{u})]\!]^{\vec{G}^*} c'_{\omega,i}(u_i) \in \{0,\dots,l_{\max}\}$ for all $\omega \in \Omega$, $i \in \{1,\dots,|\bar{x}|\}$. Since Ω is such that $\vec{G}' \models \omega$ for at most one $\omega \in \Omega$,

$$\vec{G}' \models \varphi'_2(\bar{u}) \text{ iff } \sum_{\omega\in\Omega} [\![\omega(\bar{u})]\!]^{\vec{G}^*} \sum_{i=1}^{|\bar{x}|} c'_{\omega,i}(u_i) > N'.$$

□

For the special case $N = 0$, Lemma 3.11 ammounts to the elimination of an existential quantifier.

3.5 Iterated Quantifier Elimination. In this section we finally construct the linear fpt model-checking approximation scheme for graph classes with bounded expansion. For some $\varepsilon > 0$, such a scheme either returns 1 or 0 or $\perp$. The symbol $\perp$ stands for "I do not know" and may only be returned if the input sentence is $(1+\varepsilon)$-unstable on the input graph. We obtain such an algorithm by means of quantifier elimination: For a given FO($\{>0\}$) sentence in prenex normal form and input graph, we iteratively remove the innermost quantifier of the sentence (while simultaneously computing expansions of the input graph) until no quantifiers are left. Then we evaluate the remaining quantifier-free formula on the last expansion.

Using the previous Lemma 3.11, we can construct for a formula φ of the form $\#y\,\psi > N$ (where ψ is quantifier-free) a new quantifier-free formula φ' that approximates φ. This effectively removes one counting quantifier. We will use this as the main building block of our quantifier elimination procedure. But since φ' only approximates φ, we need to be careful. If the counting term $\#y\,\psi$ evaluates to something greater than N but not greater than $(1+\varepsilon)N$ then φ and φ' may give a different answer. Since an approximation scheme is not allowed to give the wrong answer, we cannot simply replace φ with φ'.

Looking at Lemma 3.11, we notice that φ' is, in a sense, an "underapproximation" of φ: If φ' says yes, then we know for sure that φ is satisfied, but sometimes φ' may say no, even though φ is still satisfied. As we will see soon, we could also construct φ' to be an "overapproximation": If φ' says no, then we know for sure that φ is not satisfied, but sometimes φ' may say yes, even though φ is unsatisfied.

In the following, we will define over- and underapproximations of a FO($\{>0\}$)-formula φ, denoted by φ'_+, φ'_-, respectively. If the over- and underapproximation agree then we know for certain whether φ is satisfied. But if they do not agree, then φ is $(1+\varepsilon)$-unstable and our approximation scheme is allowed to return "I do not know." We call a tuple consisting of an over- and underapproximation together with an expansion of the input graph an *ε-approximation* (see Definition 3.4). Then in Lemma 3.12 we show how to obtain an ε-approximation from the results of the previous sections and how to chain ε-approximations in a meaningful way. The proof is quite technical, but ultimately unexciting. At last, in Theorem 3.1 we perform the quantifier elimination procedure. We will iteratively remove the innermost quantifier while expanding the input graph and maintaining an over- and underapproximation of the original FO($\{>0\}$) sentence. In the end, we evaluate both approximations and only return $\perp$ if they disagree.

Definition 3.4. *Let $\varphi(\bar{x})$ be an* FO($\{>0\}$)*-formula, $\varepsilon > 0$, $\vec{G}$ be a functional structure, $\vec{G}'$ be an expansion of $\vec{G}$, and $\varphi'_+(\bar{x}), \varphi'_-(\bar{x})$ be quantifier-free formulas. We say $(\vec{G}', \varphi'_+, \varphi'_-)$ is an* ε-approximation *of $(\vec{G}, \varphi)$ if there are formulas $\tilde{\varphi}_+$, $\tilde{\varphi}_-$ that are $(1+\varepsilon)$-similar to φ and for every $\bar{u} \in V(\vec{G})^{|\bar{x}|}$ the following holds.*

1. *If $\vec{G}' \models \varphi'_-(\bar{u})$ then $\vec{G} \models \varphi(\bar{u})$.*
2. *If $\vec{G}' \not\models \varphi'_-(\bar{u})$ then $\vec{G} \not\models \tilde{\varphi}_-(\bar{u})$.*
3. *If $\vec{G}' \not\models \varphi'_+(\bar{u})$ then $\vec{G} \not\models \varphi(\bar{u})$.*
4. *If $\vec{G}' \models \varphi'_+(\bar{u})$ then $\vec{G} \models \tilde{\varphi}_+(\bar{u})$.*

The next lemma describes our main quantifier elimination step. Assuming we already have an approximation of a formula φ, it gives us an approximation of $\#y\varphi > N$.

Lemma 3.12. *Let $\mathcal{C}' \subseteq \mathcal{G}(\sigma)$ be a class with bounded expansion and $\varepsilon > 0$. For given quantifier-free formulas $\varphi'_+(y\bar{x})$, $\varphi'_-(y\bar{x})$ with signature σ one can compute two quantifier-free formulas $\varphi''_+(\bar{x})$, $\varphi''_-(\bar{x})$ with signature $\rho \supseteq \sigma$ and the following property:*

There exists another class $\mathcal{C}'' \subseteq \mathcal{G}(\rho)$ with bounded expansion and for every $\vec{G}' \in \mathcal{C}'$ and $N \in \mathbf{Z}$ one can compute in time $O(\|\vec{G}'\|)$ an expansion $\vec{G}'' \in \mathcal{C}''$ of $\vec{G}'$ such that for every pair $(\vec{G}, \varphi(y\bar{x}))$:

If $(\vec{G}', \varphi'_+, \varphi'_-)$ is an ε-approximation of $(\vec{G}, \varphi)$ then $(\vec{G}'', \varphi''_+, \varphi''_-)$ is an ε-approximation of $(\vec{G}, \#y\,\varphi > N)$.

Essentially, the formulas φ''_+ and φ''_- are obtained by applying Lemma 3.11 to φ'_+ and φ'_-, respectively. The complete proof of Lemma 3.12 is quite technical and excluded from this paper due to space limitations. It can be found in the full version of this paper [5]. It will be convenient to convert FO($\{>0\}$)-formulas into the following normal form. This can be done in the same way as for plain first-order sentences.

Definition 3.5. *We say a* FO($\{>0\}$) *sentence φ is in* counting prenex normal form *if it contains no $\exists$-quantifiers, no $\forall$-quantifiers and all subformulas of the form $\varphi_1 \vee \varphi_2$ or $\varphi_1 \wedge \varphi_2$ are such that φ_1 and φ_2 are quantifier-free.*

Fact 3.1. *For every* FO($\{>0\}$) *sentence φ one can compute a* FO($\{>0\}$) *sentence φ' in counting prenex normal form such that $\varphi \equiv \varphi'$ and for structure $\vec{G}$ and $\lambda > 1$, φ is λ-stable on $\vec{G}$ iff φ' is λ-stable on $\vec{G}$.*

A formula in counting prenex normal form with at least one quantifier is either of the form $\#y\,\varphi > N$ or $\#y\,\varphi \leq N$, where φ is again in counting prenex normal form. Using Lemma 3.12, we can approximate formulas of the former form. The following simple observation additionally gives us approximations of the latter.

FACT 3.2. *If the triple $(\vec{G}', \varphi_+, \varphi_-)$ is an ε-approximation of $(\vec{G}, \varphi)$ then $(\vec{G}', \neg\varphi_-, \neg\varphi_+)$ is an ε-approximation of $(\vec{G}, \neg\varphi)$.*

We are now ready to prove our main result: an approximation scheme as described in Definition 2.4.

THEOREM 1.1. *There is a linear fpt model-checking approximation scheme for* FO($\{>0\}$) *on labeled graph classes with bounded expansion.*

Proof. As discussed in Section 3.2, we can assume that the input to our approximation scheme is a functional structure $\vec{G}$, taken from a class $\mathcal{C}$ with bounded expansion and a functional sentence φ (as well as $\varepsilon > 0$). By Fact 3.1, we can further assume that our input sentence φ is given in counting prenex form.

The main idea behind this proof is to iteratively remove the innermost counting quantifier via Lemma 3.12 until none are left. Then we can easily evaluate the remaining quantifier-free formula. For each quantifier, we will replace the input structure with an expansion. For technical reasons (Lemma 3.12 requires a non-empty tuple $\bar{x}$ of free variables), we add an unused free variable x to φ, and call the formula $\varphi(x)$. The fact whether $\varphi(x)$ is satisfied in $\vec{G}$ is independent of the assignment to x.

We define a sequence of increasing subformulas $\varphi^0, \ldots, \varphi^l$ of φ with $\varphi^l = \varphi$ as follows: φ^0 is the maximal quantifier-free subformula of φ and φ^{i+1} is either of the form $\#\,y\varphi^i > N$ or $\#\,y\varphi^i \leq N$ for some $N \in \mathbf{Z}$.

In the following, we will construct an ε-approximation $(\vec{G}^i, \varphi^i_+, \varphi^i_-)$ of $(\vec{G}, \varphi^i)$ for every $0 \leq i \leq l$. Since φ^0 is quantifier-free, by Definition 3.4 $(\vec{G}, \varphi^0, \varphi^0)$ is an ε-approximation of $(\vec{G}, \varphi^0)$ and we set set $(\vec{G}^0, \varphi^0_+, \varphi^0_-) = (\vec{G}, \varphi^0, \varphi^0)$. Assume we already have an ε-approximation $(\vec{G}^i, \varphi^i_+, \varphi^i_-)$ of $(\vec{G}, \varphi^i)$. Then Lemma 3.12 and Fact 3.2 give us an ε-approximation $(\vec{G}^{i+1}, \varphi^{i+1}_+, \varphi^{i+1}_-)$ of $(\vec{G}, \varphi^{i+1})$ as well. Each formula $\varphi^{i+1}_+, \varphi^{i+1}_-$ is constructed independently of the structure and depends only on ε and φ. Furthermore, each expansion $\vec{G}^{i+1}$ can be constructed from $\vec{G}^i$ in linear time. In the end, we have an ε-approximation $(\vec{G}^l, \varphi^l_+, \varphi^l_-)$ of $(\vec{G}, \varphi)$. Since the formulas $\varphi^l_+(x), \varphi^l_-(x)$ are quantifier-free, we can easily evaluate them on $\vec{G}^l$ (since x has no purpose in φ, we assign an arbitrary vertex to x). We can distinguish three outcomes:

- If $\vec{G}^l \models \varphi^l_-$ then by Definition 3.4, $\vec{G} \models \varphi$. We return 1.
- If $\vec{G}^l \not\models \varphi^l_+$ then by Definition 3.4, $\vec{G} \not\models \varphi$. We return 0.
- If $\vec{G}^l \not\models \varphi^l_-$ and $\vec{G}^l \models \varphi^l_+$ then by Definition 3.4 and 2.3, φ is $(1+\varepsilon)$-unstable on $\vec{G}$. We return $\perp$.

Thus, for every $\varepsilon > 0$ we have given an algorithm which takes as input a sentence φ and a structure $\vec{G} \in \mathcal{C}$, runs in time $f(|\varphi|, \varepsilon)\|\vec{G}\|$ for some function f, and whose output satisfies the criteria of a linear fpt model-checking approximation scheme, as formulated in Definition 2.4.

But Definition 2.4 further requires that there is a *single* algorithm (independent of ε) taking ε, φ, and $\vec{G}$ as input. We only presented one algorithm *for each* $\varepsilon > 0$. Our proofs are structured such that for each $\varepsilon > 0$, we use the algorithms in Lemma 3.7, 3.8, 3.10, 3.11, 3.12 and Theorem 3.1 as subroutines. After further inspection we notice that these algorithms itself can be easily computed from ε. This gives us a single algorithm running in time $f(|\varphi|, \varepsilon)\|\vec{G}\|$ and therefore a linear fpt model-checking approximation scheme. ∎

4 Hardness Results

In this section we show the linear fpt model-checking approximation scheme for FO($\{>0\}$) presented in this work to be optimal in some sense: We show that the (exact) model-checking problem for FO($\{>0\}$) is already AW[$*$]-hard on trees of depth 4, and therefore most likely not in FPT (Lemma 4.1). We further show that (unless AW[$*$] $\subseteq$ FPT) certain other fragments of FOC($\{>\}$) that are slightly stronger than FO($\{>0\}$) do not admit a model-checking approximation algorithm on the class of all trees with depth 9 – even if we only want an "approximation ratio" of $2^{\sqrt{\log(n)}}$ (Lemma 4.3). This means we cannot hope for model-checking approximation schemes on graph classes with bounded expansion if we allow for example comparison of non-atomic counting terms (e.g., $1 \cdot \#y\,\varphi_1 > 1 \cdot \#z\,\varphi_2$) or multiplication (e.g., $\#y\,\varphi_1 \cdot \#z\,\varphi_2 > N$).

The ideas behind the proofs of this section are considerably simpler than those of the remaining paper, but unfortunately they involve very technical constructions. We reduce from the AW[$*$]-complete parameterized first-order model-checking problem on the class of all graphs. The reduction showing hardness of the model-checking problem for FO($\{>0\}$) is very similar to [15, Theorem 4.1]. We encode an arbitrarily graph as a tree of depth 4 such that the underlying graph can be recovered using formulas in FO($\{>0\}$).

As this reduction is quite unstable (slight changes in the number-constants of the formula lead to the wrong answer), we have to use a more sophisticated reduction in Section 4.2 to show hardness of approximation. For

a graph with n vertices, the information "u is adjacent to v" can be written down using $O(\log(n))$ bits by assuming that the vertices of the graph are the numbers from 1 to n and then giving a binary encoding of u and v. Since this encoding has length $O(\log(n))$, we only need $O(\log\log(n))$ bits to write down the information "the ith bit of some adjacency-encoding is 1." We store this information (for every i and every edge) in a tree of bounded depth. We can then recover the original graph in a very robust manner. The counting terms will only need to distinguish between $O(\log\log(n))$ different possible values and therefore even larger perturbations in the constants of the formula will still lead to the correct answer.

4.1 Hardness of FO($\{>0\}$) Model-Checking.

LEMMA 4.1. *The* FO($\{>0\}$)*-model-checking problem is* AW[$*$]*-complete on the class of all trees of depth 4.*

The proof of Lemma 4.1 is conceptually very similar to [15, Theorem 4.1] and can also be found in the full version of this paper [5].

4.2 Hardness of Approximation for Extensions of FO($\{>0\}$).

In preparation for Lemma 4.3, we represent the edge relationship of an arbitrary graph by a binary encoding that needs only logarithmic number of bits per edge. This means that model-checking is already hard on bipartite graphs where one side of the graph has only logarithmic size. In this work, log() is the logarithm to the base two.

LEMMA 4.2. *The model-checking problem for* FO *on the class of all bipartite graphs with vertex sets U, V such that $|U| = \log(|V|)$ is* AW[$*$]*-complete.*

The proof of Lemma 4.2 can be found in the full version of this paper [5]. We now show that certain fragments of FOC($\{>\}$) similar to FO($\{>0\}$) do not admit an fpt model-checking approximation scheme on the class of all trees with depth 9. Using Lemma 4.2, we only need to encode the edge relationship of a bipartite graph where one side has only logarithmic size. By again using a binary encoding, we need roughly $\log(\log(n))$ bits to identify a vertex on the small side. This very small number of bits allows our encoding to be stable in the approximation setting.

We do not show hardness for formulas constructed by the rule $\#y\,\varphi_1 + \#z\,\varphi_2 > N$. In fact, with some additional effort, our approximation scheme can be extended to formulas of this form as well. We refrained from doing so because it does not offer much more additional expressive power.

Note that the multiplication with one in the term $1 \cdot \#y\,\varphi_1$ of rule 1 seems redundant, but is needed to comply with our definition of λ-similarity. In a λ-similar formula the constant 1 can be replaced with an arbitrary number between 1 and λ.

LEMMA 4.3. *Let* L *be a fragment of* FOC($\{>\}$) *obtained by extending* FO *by one of the following four rules (with semantics as expected, see [19] for a rigorous definition): Let y and z stand for arbitrary variables.*

1. *If φ_1, φ_2 are formulas, then $1 \cdot \#y\,\varphi_1 > 1 \cdot \#z\,\varphi_2$ is a formula.*
2. *If φ_1, φ_2 are formulas and $N \in \mathbf{Z}$, then $\#y\,\varphi_1 - \#z\,\varphi_2 > N$ is a formula.*
3. *If φ_1, φ_2 are formulas and $N \in \mathbf{Z}$, then $\#y\,\varphi_1 \cdot \#z\,\varphi_2 > N$ is a formula.*
4. *If φ, is a formula and $N \in \mathbf{Z}$, then $\#yz\,\varphi > N$ is a formula.*

Unless AW[$*$] $\subseteq$ FPT*, there is no algorithm with the following properties: It gets as input a sentence $\varphi \in$ L, a tree T of depth 9, runs in time at most $f(|\varphi|)|T|^c$ for some function f and constant c and returns either 1, 0, or $\perp$.*

- *If the algorithm returns 1 then $T \models \varphi$.*
- *If the algorithm returns 0 then $T \not\models \varphi$.*
- *If the algorithm returns $\perp$ then φ is $2^{\sqrt{\log(|T|)}}$-unstable on T.*

Proof. Since $\#y\,\varphi_1 \cdot \#z\,\varphi_2 > N$ is equivalent to $\#yz\,\varphi_1 \wedge \varphi_2 > N$ after possibly renaming variables, rule 3 is more expressive than rule 4. Also $1 \cdot \#y\,\varphi_1 > 1 \cdot \#z\,\varphi_2$ is equivalent to $\#y\,\varphi_1 - \#z\,\varphi_2 > 0$ and therefore rule 2 is more expressive than rule 1. Thus from now on, we only have to consider rule 1 and 3.

For each rule, we will provide a polynomial-time procedure which takes as input a bipartite graph G with sides U, V as well as $|U| = \log(|V|)$ and a FO sentence χ and constructs a tree T and an L-sentence φ that is $2^{\sqrt{\log(|T|)}}$-stable on T and whose size is bounded by a function of χ, such that $G \models \chi$ if and only if $T \models \varphi$. Since φ is $2^{\sqrt{\log(|T|)}}$-stable on T, a model-checking approximation algorithm for L as presented in this lemma would never yield $\perp$ on input (T, φ) and therefore decides in fpt time whether $T \models \varphi$. This would then decide whether $G \models \chi$, which by Lemma 4.2 implies AW[$*$] $\subseteq$ FPT.

Let χ be a first-order sentence and G be a bipartite graph with vertex sets U, V and $|U| = \log(|V|)$.

We can assume without loss of generality that $U = \{1, \dots, \log(n)\}$, $V = \{1, \dots, n\}$ for some sufficiently large n. We will first define some constructions that are independent of whether L is defined using rule 1 or 3 and later distinguish between the two cases. Let $\lambda = \lfloor 2^{\log(n)^{2/3}} \rfloor$. The tree T will contain different "gadget" trees as subtrees. We construct a "gadget" tree T_i, for $i \in \{1, \dots, \log(n)\}$ using the following procedure: We start with inserting a root q. Then for every $l \in \{1, \dots, \lfloor \log(\log(n)) \rfloor + 1\}$ we

- insert a vertex a_l and an edge qa_l,
- if the lth bit of the binary encoding of i is one insert a vertex b_l and an edge $a_l b_l$,
- insert λ^{3l} many so-called "c-vertices", for each new c-vertex c also add a new vertex c' and edges $a_l c$, cc',
- insert $\lambda^{4\lfloor \log(\log(n)) \rfloor - 3l}$ many so-called "d-vertices", for each new d-vertex d also add new vertices d', d'' and edges $a_l d$, dd', $d'd''$,
- insert λ^{3l+1} many so-called "c^+-vertices", for each new c^+-vertex c also add new vertices c', c'', c''' and edges $a_l c$, cc', $c'c''$, $c''c'''$,
- insert λ^{3l-1} many so-called "c^--vertices", for each new c^--vertex c also add new vertices c', c'', c''', c'''' and edges $a_l c$, cc', $c'c''$, $c''c'''$, $c'''c''''$.

The number of c-, c^+-, c^-- and d-children are all powers of λ. This will later help us build λ-stable L-formulas that can identify the different a-vertices based on their number of children. Since $\lambda^{\lfloor \log(\log(n)) \rfloor} = n^{O(1)}$, each gadget tree T_i has polynomial size and can be constructed in polynomial time. We construct T of depth 9 with the following polynomial-time procedure:

- insert a root r,
- for $i \in U$ insert a copy of T_i, rename its root u_i and add an edge $u_i r$,
- for $j \in V$ insert vertices v_j, e_j, f_j and edges rv_j, $v_j e_j$, $v_j f_j$,
- for $ij \in E(G)$ with $i \in U$ and $j \in V$ insert a copy of T_i, name its root $w_{i,j}$ and add an edge $w_{i,j} v_j$.

In order to build φ, we first need formulas $\varphi_a(x)$, $\varphi_b(x)$, $\varphi_c(x)$, $\varphi_{c^+}(x)$, $\varphi_{c^-}(x)$, $\varphi_d(x)$, $\varphi_u(x)$, $\varphi_v(x)$, $\varphi_w(x)$ defining the set of corresponding vertices. They can be build using these observations:

- a-vertices are those vertices with at least three neighbors of degree two,
- the b-, c-, d-, c^+-, c^--vertices are the neighbors of a-vertices with degree one or two and "tails" of length 0, 1, 2, 3, 4 respectively,
- w-vertices are those with more than one a-neighbor and distance two to a leaf,
- u-vertices are those with more than one a-neighbor and not distance two to a leaf,
- v-vertices are those adjacent to a w-vertex and exactly two leaves.

There is a bijection between U and the u-vertices in T and V and the v-vertices in T. We want to check whether two vertices $i \in U$ and $j \in V$ are adjacent in G by evaluating a formula with two free variables on u_i and v_j in T. We know that u_i is the root of a gadget tree isomorphic to T_i. Furthermore i and j are adjacent in G if and only if v_j has a w-child that is the root of a gadget tree isomorphic to T_i. In order to test that, we need an L-formula $\psi_{\text{gadget}}(x, y)$ such that for every u-vertex u and w-vertex w in T holds $T \models \psi_{\text{gadget}}(u, w)$ if and only if u and w are the roots of two isomorphic gadget trees. We further require $\psi_{\text{gadget}}(u, w)$ to be λ-stable on T. If we have such a formula, the remaining proof is similar to the ones of Lemma 4.1 and 4.2.

Let u, w be the roots of two gadget trees isomorphic to $T_i, T_{i'}$ in T. For each $l \in \{1, \dots, \lfloor \log(\log(n)) \rfloor + 1\}$ let a_l^u and a_l^w be the a-child of u and w, respectively, with λ^{3l} c-children. We first construct a formula $\psi_{\text{pair}}(x, y)$ such that $T \models \psi_{\text{pair}}(a_l^u, a_{l'}^w)$ iff $l = l'$. For this, we have to distinguish whether L extends FO using rule 1 or 3.

1. This rule can compare two non-atomic counting terms. With this ability, we can pair a_l^u with $a_{l'}^w$ if the number of c-children of a_l^u lies between the number of c^-- and c^+-children of $a_{l'}^w$. The pairing is achieved by the formula

$$\begin{aligned}\psi_{\text{pair}}(x, y) = \big(1 \cdot \#z\, \varphi_{c^+}(z) \wedge E(z, x) \\ \geq 1 \cdot \#z\, \varphi_c(z) \wedge E(z, y)\big) \\ \wedge \big(1 \cdot \#z\, \varphi_{c^-}(z) \wedge E(z, x) \\ \leq 1 \cdot \#z\, \varphi_c(z) \wedge E(z, y)\big).\end{aligned}$$

3. This rule can compare the product of two non-constant counting terms. With this ability, we can pair a_l^u with $a_{l'}^w$ if the product of the number c-children of a_l^u and the the number of d-children of $a_{l'}^w$ equals $\lambda^{4\lfloor \log(\log(n)) \rfloor}$. This pairing can be

modeled by the formula

$$\psi_{\text{pair}}(x,y) = \big(\#z\,\varphi_c(z) \wedge E(z,x) \cdot \#z\,\varphi_d(z) \wedge E(z,y) \geq \lambda^{4\lfloor \log(\log(n))\rfloor - 1}\big) \wedge \big(\#z\,\varphi_c(z) \wedge E(z,x) \cdot \#z\,\varphi_d(z) \wedge E(z,y) \leq \lambda^{4\lfloor \log(\log(n))\rfloor + 1}\big).$$

For two different a-children, their number of c-, c^+-, c^--, d-children differ at least by a factor of λ^3. Therefore, we still have $T \models \psi'_{\text{pair}}(a_l^u, a_{l'}^w)$ iff $l = l'$ for every λ-similar formula ψ'_{pair}. This makes $\psi_{\text{pair}}(a_l^u, a_{l'}^w)$ λ-stable on T. We can now compare the binary encoding of i and i': It holds $i = i'$ if and only if for each pair of a-vertices either both or none have a b-child. We define

$$\psi_{\text{gadget}}(x,y) = \forall a \forall a' \Big(\big(\varphi_a(a) \wedge \varphi_a(a') \wedge E(a,x) \wedge E(a',y) \wedge \psi_{\text{pair}}(a,a')\big) \rightarrow \big(\big(\exists b \varphi_b(b) \wedge E(a,b)\big) \leftrightarrow \big(\exists b \varphi_b(b) \wedge E(a',b)\big)\big)\Big).$$

Now $i = i'$ iff $T \models \psi_{\text{gadget}}(u,w)$. Since $\psi_{\text{pair}}(a_l^u, a_{l'}^w)$ is λ-stable, $\psi_{\text{gadget}}(u,w)$ also is λ-stable on T. Using

$$\psi_E(x,y) = \exists z \big(\varphi_w(z) \wedge E(y,z) \wedge \psi_{\text{gadget}}(x,z)\big),$$

we have for every $i \in U$, $j \in V$ that i and j are adjacent in G iff $T \models \psi_E(u_i, v_j)$. We can then construct φ from χ by replacing every occurrence of $E(x,y)$ with

$$\big(\varphi_u(x) \wedge \varphi_v(y) \wedge \psi_E(x,y)\big) \vee \big(\varphi_u(y) \wedge \varphi_v(x) \wedge \psi_E(y,x)\big)$$

and relativizing all quantifiers to u- and v-vertices by replacing subformulas $\exists x \psi$ with $\exists x((\varphi_u(x) \vee \varphi_v(x)) \wedge \psi)$. Then $G \models \chi$ iff $T \models \varphi$. Since ψ_{gadget} is λ-stable on T for all u- and w-vertices, φ also is λ-stable on T. The tree T has polynomial size and therefore

$$2^{\sqrt{\log(|T|)}} = 2^{\sqrt{O(1)\log(n)}} = o(2^{\log(n)^{2/3}}) = o(\lambda).$$

Without loss of generality we can assume n to be sufficiently large that $2^{\sqrt{\log(|T|)}} \leq \lambda$. This means that φ is $2^{\sqrt{\log(|T|)}}$-stable on T. ∎

5 Open Questions

We see the following open questions, sorted in descending order by estimated difficulty.

It should be possible to generalize our FO($\{>0\}$) model-checking approximation scheme from classes with bounded expansion to nowhere dense graph classes. Our approach, however, using functional representations and quantifier elimination will most likely not be applicable. Since a FO($\{>0\}$) model-checking approximation scheme also solves the model-checking problem for FO, we cannot hope to extend our results beyond nowhere dense graph classes (assuming monotonicity) [13].

Can our optimization result in Theorem 1.2 for counting-terms of the form $\#y\varphi(y\bar{x})$ with $\varphi(y\bar{x}) \in$ FO be extended to nowhere dense graph classes? At this point, we do not even know whether we can efficiently solve partial dominating set (special case of Theorem 1.2) on nowhere dense graph classes. We believe both to be the case. There is potential for further optimization results similar to Theorem 1.2 for different fragments of FOC($\mathbf{P}$) on bounded expansion or possibly nowhere dense graph classes.

Can our model-checking approximation scheme be generalized to query-counting or query-enumeration? For every FO($\{>0\}$)-formula $\varphi(\bar{x})$, structure G and $\varepsilon > 0$ we define:

$$\lceil \varphi, G, \varepsilon \rceil = \{\bar{u} \in V(G)^{|\bar{x}|} \mid \text{there exists } \varphi' \text{ that is } (1+\varepsilon)\text{-similar to } \varphi \text{ with } G \models \varphi'(\bar{u})\},$$
$$\lfloor \varphi, G, \varepsilon \rfloor = \{\bar{u} \in V(G)^{|\bar{x}|} \mid \text{for all } \varphi' \text{ that are } (1+\varepsilon)\text{-similar to } \varphi \text{ holds } G \models \varphi'(\bar{u})\}.$$

Can we compute in time $f(|\varphi|, \varepsilon)\|G\|$ a number $s \in \mathbf{N}$ with $|\lfloor \varphi, G, \varepsilon \rfloor| \leq s \leq |\lceil \varphi, G, \varepsilon \rceil|$? Is it possible to enumerate with constant-delay a set S with $\lfloor \varphi, G, \varepsilon \rfloor \subseteq S \subseteq \lceil \varphi, G, \varepsilon \rceil$? Using our approximate quantifier elimination procedure in Section 3.5, one probably can replace $\varphi(\bar{x})$ with a quantifier-free formula and then use existing query-counting and query-enumeration techniques.

Consider the logic FOC($\{=_p\}$), where $=_p$ is the equality relation modulo p. Then, for example, the FOC($\{=_2\}$)-formula $\forall x \, \#y \, E(x,y) =_2 0$ expresses if a graph has an Euler cycle. It is already known that the model-checking problem for FOC($\{=_p\}$) is fpt on graph classes with bounded degree [16]. For nowhere dense graph classes the proof might be considerably more difficult and require different techniques.

References

[1] O. Amini, F. V. Fomin, and S. Saurabh. Implicit branching and parameterized partial cover problems. *J. Comput. Syst. Sci.*, 77(6):1159–1171, 2011.

[2] B. Courcelle. The monadic second-order logic of graphs I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[3] E. D. Demaine, F. Reidl, P. Rossmanith, F. S. Villaamil, S. Sikdar, and B. D. Sullivan. Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *J. Comput. Syst. Sci.*, 105:199–241, 2019.

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

[5] J. Dreier and P. Rossmanith. Approximate evaluation of first-order counting queries. *CoRR*, abs/2010.14814, 2020.

[6] A. Durand and E. Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Trans. Comput. Log.*, 8(4):21, 2007.

[7] Z. Dvorák, D. Kráľ, and R. Thomas. Testing first-order properties for subclasses of sparse graphs. *J. ACM*, 60(5):36:1–36:24, 2013.

[8] H.-D. Ebbinghaus and J. Flum. *Finite model theory*. Springer Science & Business Media, 2005.

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

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

[11] M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Log.*, 130(1-3):3–31, 2004.

[12] M. Grohe. The parameterized complexity of database queries. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA*, pages 82–92. ACM, 2001.

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

[14] M. Grohe and W. Pakusa. Descriptive complexity of linear equation systems and applications to propositional proof complexity. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017.

[15] M. Grohe and N. Schweikardt. First-order query evaluation with cardinality conditions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 253–266. ACM, 2018.

[16] L. Heimberg, D. Kuske, and N. Schweikardt. Hanf normal form for first-order logic with unary counting quantifiers. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 277–286. ACM, 2016.

[17] W. Kazana and L. Segoufin. Enumeration of first-order queries on classes of structures with bounded expansion. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 297–308. ACM, 2013.

[18] S. Kreutzer. Algorithmic meta-theorems. In *Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings*, volume 5018 of *Lecture Notes in Computer Science*, pages 10–12. Springer, 2008.

[19] D. Kuske and N. Schweikardt. First-order logic with counting. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017.

[20] A. Langer, F. Reidl, P. Rossmanith, and S. Sikdar. Evaluation of an MSO-solver. In *Proceedings of the 14th Meeting on Algorithm Engineering & Experiments, ALENEX 2012, The Westin Miyako, Kyoto, Japan, January 16, 2012*, pages 55–63. SIAM / Omnipress, 2012.

[21] L. Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[22] J. Nešetřil and P. O. de Mendez. Grad and classes with bounded expansion I. Decompositions. *Eur. J. Comb.*, 29(3):760–776, 2008.

[23] J. Nešetřil and P. O. de Mendez. Grad and classes with bounded expansion II. Algorithmic aspects. *Eur. J. Comb.*, 29(3):777–791, 2008.

[24] J. Nešetřil and P. O. de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[25] J. Nurmonen. Counting modulo quantifiers on finite structures. *Inf. Comput.*, 160(1-2):62–87, 2000.

[26] N. Schweikardt, L. Segoufin, and A. Vigny. Enumeration for FO queries over nowhere dense graphs. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 151–163. ACM, 2018.

[27] D. Seese. Linear time computable problems and first-order descriptions. *Math. Struct. Comput. Sci.*, 6(6):505–526, 1996.

[28] S. Torunczyk. Aggregate queries on sparse databases. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 427–443. ACM, 2020.

[29] J. A. Väänänen. Unary quantifiers on finite models. *Journal of Logic, Language and Information*, 6(3):275–304, 1997.