The Logic of Counting Query Answers: A Study via Existential Positive Queries

Hubie Chen * Stefan Mengel†

January 28, 2015

We consider the computational complexity of counting the number of answers to a logical formula on a finite structure. In the setting of parameterized complexity, we present a trichotomy theorem on classes of existential positive queries. We then proceed to study an extension of first-order logic in which algorithms for the counting problem at hand can be naturally and conveniently expressed.

1. Introduction

1.1. Background

The computational problem of evaluating a formula (of some logic) on a finite relational structure is of central interest in database theory and logic. In the context of database theory, this problem is often referred to as *query evaluation*, as it models the posing of a query to a database, in a well-acknowledged way: the formula is the query, and the structure represents the database. We will refer to the results of such an evaluation as *answers*; logically, these are the satisfying assignments of the formula on the structure. The particular case of this problem where the formula is a sentence is often referred to as *model checking*, and even in just the case of first-order sentences, can capture a variety of well-known decision problems from all throughout computer science [FG06].

In this article, we study the counting version of this problem, namely, given a formula and a structure, output the *number* of answers (see for example [PS11, GS, DM13, CM14] for previous studies). This counting problem generalizes model checking, which can be viewed as the particular case thereof where one is given a sentence and structure, and wants to decide if the number of answers is 1 or 0, corresponding to whether or not the empty assignment is satisfying. In addition to the counting problem's fundamental

*Universidad del Pas Vasco, E-20018 San Sebastin, Spain, *and* IKERBASQUE, Basque Foundation for Science, E-48011 Bilbao, Spain

†LIX UMR 7161, École Polytechnique, Université Paris Saclay, France

interest, it can be pointed out that all practical query languages supported by database management systems have a counting operator. Indeed, it has been argued that database queries with counting are at the basis of decision support systems that handle large data volume [GS].

As has been articulated [PY99, FG06], a typical situation in the database setting is the evaluation of a relatively short formula on a relatively large structure. Consequently, it has been argued that, in measuring the time complexity of query evaluation tasks, one could reasonably allow a slow (non-polynomial-time) preprocessing of the formula, so long as the desired evaluation can be performed in polynomial time following the preprocessing. Relaxing polynomial-time computation to allow arbitrary preprocessing of a *parameter* of a problem instance yields, in essence, the notion of *fixed-parameter tractability*. This notion of tractability is at the core of *parameterized complexity theory*, which provides a taxonomy for classifying problems where each instance has an associated parameter. We utilize this paradigm in this article (here, the formula is the parameter).

1.2. Contribution: complexity classification of existential positive queries

Existential positive queries are the first-order formulas built from the two binary connectives ($\wedge$, $\vee$) and existential quantification. They are semantically equivalent to so-called *unions of conjunctive queries* (also known as *select-project-join-union queries*), which have been argued to be the most common database queries [AHV95].

We study the problem of counting query answers on existential positive queries. In particular, we study the complexity of this problem relative to sets of existential positive queries: each such set Φ gives a restricted version of the general problem, namely, count the number of answers of a given formula $\phi \in \Phi$ on a given finite structure $\mathbf{B}$. We hence have a family of problems, one problem for each such set Φ.

We prove a trichotomy theorem (Theorem 5.1) on the parameterized complexity of the described family of problems, which shows essentially that each such problem is fixed-parameter tractable, equivalent to the clique problem, or as hard as the counting clique problem. This theorem is in fact proved by showing that for each set of existential positive queries, there exists a set of primitive positive queries such that the two sets exhibit the same complexity behavior (see Theorem 4.1). The trichotomy at hand is then derived from a known trichotomy on primitive positive queries [CM14]; recall that primitive positive queries are first-order formulas built from the connective $\wedge$ and existential quantification. (Let us also remark that here and throughout, we focus on sets of formulas of *bounded arity*, that is, for which there is a constant upper bound on the arity of used relation symbols.)

Our new trichotomy theorem generalizes and unifies a number of existing complexity classification results in the literature, namely those on model checking primitive positive formulas [DKV02, Gro07], model checking existential positive formulas [Che14a], counting answers to quantifier-free primitive positive formulas [DJ04], and counting answers to primitive positive formulas [CM14].

Let us remark that we are not aware of any fragment of first-order logic extending existential positive queries for which even model checking is understood, from the viewpoint of classifying the complexity of all sets of queries (for more information, see the discussion in [Che14b]). Hence, the research project of extending our complexity classification beyond existential positive queries would first require an advance in the study of model checking in first-order logic.

1.3. Contribution: ♯-logic

The width of a first-order formula ϕ is defined as the maximum number of free variables over all subformulas of ϕ. It is well-known that bounded width sentences are computationally desirable for evaluation; this is made precise as follows.

Observation 1.1 *[Var95] For each $k \geqslant 1$, the problem of evaluating a first-order sentence with width at most k on a given finite structure is polynomial-time computable, via the natural bottom-up evaluation algorithm.*

The following consequence is immediate.

Observation 1.2 *[Che14a] The following condition, which we will call the* classical condition*, is sufficient for fixed-parameter tractable model checking on a sentence set Φ: there exist $k \geqslant 1$ and an algorithm f that computes, for each $\phi \in \Phi$, a logically equivalent sentence $f(\phi)$ of width at most k.*

Interestingly, it is known that for existential positive sentences, this condition is the exclusive explanation for tractability: if a set Φ of such sentences is fixed-parameter tractable at all, then it satisfies the classical condition (see [Che14a] and the discussion therein). Let us remark that the query processing algorithm f here is related to and akin to the database notion of a query optimizer that computes a query execution plan.

The conceptual point that we wish to highlight here is that for the computational problem of model checking first-order sentences, *first-order logic itself can be used as a model of computation in which desirable, efficient algorithms can be expressed.* Note that here, logic can be seen as playing two complementary roles: one the one hand, the computational problem of interest is phrased directly in terms of logical sentences; on the other hand, the desired algorithmic solutions to this problem are themselves describable using logical sentences.

Inspired by this view of *logic as a useful model of computation*, we introduce an extension of first-order logic, called *♯-logic*, in which one can directly express numerical functions of structures.[1] In ♯-logic, the evaluation of a *♯-sentence* (a type of formula in ♯-logic) on a structure returns an integer value, as opposed to a propositional value as for usual sentences. Hence, it is possible for a ♯-sentence ψ to *represent* a first-order formula ϕ in the sense that, on any structure $\mathbf{B}$, the value of ψ is equal to the number of query answers for ϕ.

[1] Note that one can readily construct a set Φ of formulas on which counting query answers is tractable but which has unbounded width; consider for instance $\Phi = \{\phi_1, \phi_2, \ldots\}$ where $\phi_n(v_1, \ldots, v_n) = S_1(v_1) \wedge \cdots \wedge S_n(v_n)$.

From the discussed viewpoint of bounded width as an explanation for tractability, the relationship of the counting query answers problem to $\sharp$-logic is strongly analogous to the relationship of model checking to usual first-order logic. Indeed, we have the following parallels of the above observations.

Observation 1.3 *For each $k \geqslant 1$, the problem of evaluating a $\sharp$-sentence having width at most k on a given finite structure is polynomial-time computable.*

The following consequence is entailed.

Observation 1.4 *The following condition, which we call the* new condition*, is sufficient for fixed-parameter tractability of counting query answers on a formula set Φ: there exist $k \geqslant 1$ and an algorithm f that computes, for each $\phi \in \Phi$, a $\sharp$-sentence representation $f(\phi)$ of width at most k.*

We further establish the result that, in a parallel to the classical condition being the exclusive explanation for the tractability of model checking, the new condition is the exclusive explanation for tractability for the problem of counting answers to existential positive formulas. On a conceptual level, we view this result as evidence that, for the problem of counting query answers, $\sharp$-logic is a useful model of computation in which desirable, efficient algorithms can be expressed. We obtain this result by proving two theorems:

- First, we show that if counting answers on a set Φ of such formulas is tractable at all, then the $\phi \in \Phi$ have representations of bounded width (Theorem 7.4).
- Second, we prove that there is an algorithm that, given an existential positive formula, computes a representation of minimum width (Theorem 7.5).

This latter theorem can be read as stating that $\sharp$-logic is well-characterized and well-understood as a model of computation: conceiving of a $\sharp$-sentence representation of an existential positive formula as a computational procedure for counting query answers, this theorem provides an algorithm that always outputs an *optimal* procedure for a given existential positive formula, where optimality here is measured with respect to width. This theorem is technically non-trivial; it uses in a key fashion a theorem which, intuitively speaking, shows the linear independence of pp-formulas that are pairwise inequivalent in a counting sense (Theorem 9.6). This independence-type theorem introduces and makes use of a number of novel notions that may be of future utility, including a notion of a polynomial acting on a relational structure, and a notion of multivariate polynomial associated to a pp-formula (see Section 9.2).

In short, our presentation and study of $\sharp$-logic forwards the discussed use of logic as a means for expressing computationally desirable procedures; in particular, $\sharp$-logic allows one to directly express procedures for counting query answers.

Finally, let us remark that $\sharp$-logic allows for the expression of a well-known algorithm for the problem of counting the number of homomorphisms from a given *source* structure $\mathbf{A}$ to a given *target* structure $\mathbf{B}$. (It is well-known and straightforward to verify that this problem is equivalent to counting the number of answers to a quantifier-free primitive positive formula on a given structure **B**.) In particular, this problem is known to be polynomial-time tractable when there is a constant treewidth bound on the permitted source structures; indeed, the corresponding algorithm, which performs dynamic programming over a tree decomposition of **A**, has received a textbook treatment [FK10, Section 5.3]. It is readily verified that from a tree decomposition for a structure **A**, one can compute (in polynomial time, using the algorithm of Observation 1.4) a $\sharp$-sentence representing the problem of counting homomorphisms from **A**, which sentence has width precisely equal to the width of the given decomposition (plus one). Indeed, we believe that the resulting $\sharp$-sentences accurately, faithfully, and cleanly describes the execution of this tractability result's algorithm.

2. Preliminaries

Note that $\cdot$ is sometimes used for multiplication of real numbers.

Polynomials. We remind the reader of some basic facts about polynomials which we will use throughout the paper. Here, a univariate polynomial p in a variable x is a function $p(x) = \sum_{i=0}^{d} a_i x^i$ where $d \geqslant 1$, each $a_i \in \mathbb{R}$ and $a_d \neq 0$, or the *zero polynomial* $p(x) = 0$. The a_i are called *coefficients* of p. The degree of a polynomial is defined as $-\infty$ in the case of the zero polynomial, and as d otherwise. Let $(x_0, y_0), \ldots, (x_n, y_n)$ be $n+1$ pairs of real numbers. Then there is a uniquely determined polynomial of degree at most n such that $p(x_i) = y_i$ for each i; consequently, a polynomial p of degree n that has at least $n+1$ zeroes (where a *zero* is a value x such that $p(x) = 0$) is the zero polynomial. If all x_i and y_i are rational numbers, then the coefficients a_i of this polynomial are rational numbers as well; moreover, the a_i can be computed in polynomial time.

Logic. We assume basic familiarity with the syntax and semantics of first-order logic. In this article, we focus on relational first-order logic where equality is not built-in to the logic. Hence, each *vocabulary* under discussion consists only of relation symbols. We assume structures under discussion to be *finite* (that is, have finite universe); nonetheless, we sometimes describe structures as *finite* for emphasis. We assume that the relations of structures are represented as lists of tuples. We use the letters **A**, **B**, ... to denote structures, and the corresponding letters A, B, ... to denote their respective universes. When τ is a signature, we use $\mathbf{I}_\tau$ to denote the τ-structure with universe $\{a\}$ and where each relation symbol $R \in \tau$ has $R^{\mathbf{I}} = \{(a, \ldots, a)\}$.

We use the term *fo-formula* to refer to a first-order formula. An *ep-formula* (short for *existential positive formula*) is a fo-formula built from *atoms* (by which we refer to predicate applications of the form $R(v_1, \ldots, v_k)$), conjunction ($\wedge$), disjunction ($\vee$), and existential quantification ($\exists$). A *pp-formula* (short for *primitive positive formula*) is an ep-formula where disjunction does not occur.

The set of free variables of a formula ϕ is denoted by free(ϕ) and is defined as usual. A formula ϕ is a *sentence* if free(ϕ) $= \varnothing$. A primary concern in this article is in counting satisfying assignments of fo-formulas on a finite structure. The count is sensitive to the set of variables over which assignments are considered; and, we will sometimes want to count relative to a set of variables that is strictly larger than the set of free variables. Hence, we will often associate with each fo-formula ϕ a set V of variables called the *liberal variables*, denoted by $\mathrm{lib}(\phi)$, which is required to be a superset of $\mathrm{free}(\phi)$. To indicate that V is the set of liberal variables of ϕ, we often use the notation $\phi(V)$ (we also use $\phi(v_1, \ldots, v_n)$, where the v_i are a listing of the elements of V). Relative to a formula $\phi(V)$, when $\mathbf{B}$ is a structure, we will use $\phi(\mathbf{B})$ to denote the set of assignments $f : V \to B$ such that $\mathbf{B}, f \models \phi$.

An fo-formula is *prenex* if it has the form $Q_1 v_1 \ldots Q_n v_n \theta$ where θ is quantifier-free, that is, if all quantifiers occur in the front of the formula. *We assume that, in each prenex formula with liberal variables associated with it, no variable is both liberal and quantified.*

An fo-formula ϕ is *free* if $\mathrm{free}(\phi) \neq \emptyset$. An fo-formula ϕ is *liberal* if $\mathrm{lib}(\phi)$ is defined and $\mathrm{lib}(\phi) \neq \emptyset$.

pp-formulas. Each prenex pp-formula $\phi(S)$ may be viewed as a pair $(\mathbf{A}, S)$ where the universe A of $\mathbf{A}$ is the union of S with the variables appearing in ϕ, and a tuple $(a_1, \ldots, a_k) \in A^k$ is in $R^{\mathbf{A}}$ if and only if $R(a_1, \ldots, a_k)$ appears in ϕ. *We will freely interchange between this structure view and the usual notion of a prenex pp-formula.* For such a pair $(\mathbf{A}, S)$, we generally assume that $S \subseteq A$.

Two structures are *homomorphically equivalent* if each has a homomorphism to the other. A structure is a *core* if it is not homomorphically equivalent to a proper substructure of itself. A structure $\mathbf{B}$ is a *core* of a structure $\mathbf{A}$ if $\mathbf{B}$ is a substructure of $\mathbf{A}$ that is a core and is homomorphically equivalent to $\mathbf{A}$. It is known that all cores of a structure are isomorphic and hence one sometimes speaks of *the* core of a structure.

Definition 2.1 *For a prenex pp-formula $(\mathbf{A}, S)$ where $\mathbf{A}$ is defined on vocabulary τ, we define the* augmented structure*, denoted by $\mathsf{aug}(\mathbf{A}, S)$, to be the structure over the expanded vocabulary $\tau \cup \{R_a \mid a \in S\}$ (understood to be a disjoint union) where $R_a^{\mathsf{aug}(\mathbf{A},S)} := \{a\}$. We define the* core *of the pp-formula $(\mathbf{A}, S)$ to be the core of $\mathsf{aug}(\mathbf{A}, S)$.*

We present fundamental facts on pp-formulas, which will be used throughout.

Theorem 2.2 (follows from [CM77]) *Let $(\mathbf{A}, V)$, $(\mathbf{B}, V)$ be prenex pp-formulas. The formula $(\mathbf{B}, V)$ logically entails the $(\mathbf{A}, V)$ if and only if there exists a homomorphism from $\mathsf{aug}(\mathbf{A}, V)$ to $\mathsf{aug}(\mathbf{B}, V)$. The formulas $(\mathbf{A}, V)$, $(\mathbf{B}, V)$ are logically equivalent if and only if they have isomorphic cores, or equivalently, when $\mathsf{aug}(\mathbf{A}, V)$ and $\mathsf{aug}(\mathbf{B}, V)$ are homomorphically equivalent.*

We will also make use of the basic known fact [CM77] that, when $\mathbf{B}$ is a structure, $f : S \to B$ is a map, and $\phi(S)$ is a pp-formula with pair $(\mathbf{A}, S)$, $\mathbf{B}, f \models \phi(S)$ if and only if there is an extension f' of f that is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.

ep-formulas. In order to discuss ep-formulas, we will employ the following terminology. An ep-formula is *disjunctive* if it is the disjunction of prenex pp-formulas. An ep-formula is *all-free* if it is disjunctive and each pp-formula appearing as a disjunct is free. An ep-formula is *normalized* if it is disjunctive and for each sentence disjunct $(\mathbf{A}, S)$ and any other disjunct $(\mathbf{A}', S)$, there is no homomorphism from $\mathsf{aug}(\mathbf{A}, S)$ to $\mathsf{aug}(\mathbf{A}', S)$ (equivalently, there is no homomorphism from $\mathbf{A}$ to $\mathbf{A}'$). It is straightforward to verify that there is an algorithm that, given an ep-formula, outputs a logically equivalent normalized ep-formula.

Graphs We assume the reader to be familiar with the definition and basic facts on treewidth, see e.g. [FG06]. For the convenience of the reader we introduce the notions we will use in Appendix A.

To every prenex pp-formula $(\mathbf{A}, S)$ we assign a graph whose vertex set is $A = A \cup S$ and where two vertices are connected by an edge if they appear together in a tuple of a relation of $\mathbf{A}$. We define the treewidth of a prenex pp-formula to be that of its graph. A prenex pp-formula $(\mathbf{A}, S)$ is called *connected* if its graph is connected. When $(\mathbf{A}, S)$, $(\mathbf{A}', S')$ are two pp-formulas, we write $(\mathbf{A}, S) \subseteq (\mathbf{A}', S')$ and say that $(\mathbf{A}, S)$ is a *preformula* of $(\mathbf{A}', S')$ if $A \subseteq A'$, $S \subseteq S' \cap A$, and $R^{\mathbf{A}} \subseteq R^{\mathbf{A}'}$ for each relation symbol R. A *component* of a prenex pp-formula $(\mathbf{A}', S')$ is a connected preformula $(\mathbf{A}, S)$ of $(\mathbf{A}', S')$ that is maximal (in the $\subseteq$ order just defined) among all connected preformulas of $(\mathbf{A}', S')$. It is straightforward to verify that two components $(\mathbf{A}_1, S)$, $(\mathbf{A}_2, S)$ of a prenex pp-formula are either equal or have $A_1 \cap A_2 = \emptyset$. Suppose that $\phi(V)$ is a prenex pp-formula and $v \in V$ is a liberal variable. Setting $\mathbf{C}$ to be the structure with universe $\{v\}$ and having only empty relations, we have that $(\mathbf{C}, \{v\})$ is a connected preformula of $\phi(V)$; if $v \notin \mathrm{free}(\phi)$, then it is a component of $\phi(V)$.

Complexity theory. Throughout, we use Σ to denote an alphabet over which strings are formed. All problems to be considered are viewed as counting problems. So, a *problem* is a mapping $Q : \Sigma^* \to \mathbb{N}$. We view decision problems as problems where, for each $x \in \Sigma^*$, it holds that $Q(x)$ is equal to 0 or 1. A *parameterization* is a mapping $\kappa : \Sigma^* \to \Sigma^*$. A parameterized problem is a pair (Q, κ) consisting of a problem Q and a parameterization κ.

We define count to be the problem that maps a pair $(\phi(V), \mathbf{B})$ consisting of a fo-formula and a finite structure to the value $|\phi(\mathbf{B})|$. We define p-count to be the parameterized problem (count, π_1). Here, by π_i we denote the operator that projects a tuple onto its ith coordinate.

A partial function $T : \Sigma^* \to \mathbb{N}$ is *polynomial-multiplied* with respect to a parameterization κ if there exists a computable function $f : \Sigma^* \to \mathbb{N}$ and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that, for each $x \in \mathrm{dom}(T)$, it holds that $T(x) \leqslant f(\kappa(x))p(|x|)$.

Definition 2.3 *Let $\kappa : \Sigma^* \to \Sigma^*$ be a parameterization. A partial mapping $r : \Sigma^* \to \Sigma^*$ is* FPT-computable *with respect to κ if there exist a polynomial-multiplied function $T : \Sigma^* \to \mathbb{N}$ (with respect to κ) with* $\mathrm{dom}(T) = \mathrm{dom}(r)$ *and an algorithm A such that,* *for each string $x \in \mathrm{dom}(r)$, the algorithm A computes $r(x)$ within time $T(x)$; when this holds, we also say that r is* FPT-computable *with respect to κ* via *A.*

As is standard, we may and do freely interchange among elements of Σ^*, $\Sigma^* \times \Sigma^*$, and $\mathbb{N}$. We define FPT to be the class that contains a parameterized problem (Q, κ) if and only if Q is FPT-computable with respect to κ.

We now introduce a notion of reduction for counting problems, which is a form of Turing reduction. We use $\wp_{\mathsf{fin}}(A)$ to denote the set containing all finite subsets of A.

Definition 2.4 *A* counting FPT-reduction *from a parameterized problem (Q, κ) to another (Q', κ') consists of a computable function $h : \Sigma^* \to \wp_{\mathsf{fin}}(\Sigma^*)$, and an algorithm A such that:*

- *on an input x, A may make oracle queries of the form $Q'(y)$ with $\kappa'(y) \in h(\kappa(x))$, and*
- *Q is FPT-computable with respect to κ via A.*

We use Clique to denote the decision problem where (k, G) is a yes-instance when G is a graph that contains a clique of size $k \in \mathbb{N}$. By #Clique we denote the problem of counting, given (k, G), the number of k-cliques in the graph G. The parameterized versions of these problems, denoted by p-Clique and p-#Clique, are defined via the parameterization $\pi_1(k, G) = k$.

3. Counting case complexity

3.1. Framework

We employ the framework of *case complexity* to develop some of our complexity results. We present the needed elements of this framework for counting problems. The definitions and results here are due to [CM14] and are based on the theory of [Che14b]; see those articles for motivation of the framework.

A *case problem* consists of a problem $Q : \Sigma^* \times \Sigma^* \to \mathbb{N}$ and a subset $S \subseteq \Sigma^*$, and is denoted $Q[S]$; for each case problem $Q[S]$, we define $\mathsf{param}\text{-}Q[S]$ as the parameterized problem (P, π_1) where $P(s, x)$ is defined as equal to $Q(s, x)$ if $s \in S$, and as 0 otherwise.

We now introduce a reduction notion for case problems.

Definition 3.1 *A* counting slice reduction *from a case problem $Q[S]$ to a second case problem $Q'[S']$ consists of*

- *a computably enumerable language $U \subseteq \Sigma^* \times \wp_{\mathsf{fin}}(\Sigma^*)$, and*
- *a partial function $r : \Sigma^* \times \wp_{\mathsf{fin}}(\Sigma^*) \times \Sigma^* \to \Sigma^*$ that has domain $U \times \Sigma^*$ and is FPT-computable with respect to (π_1, π_2) via an algorithm A that, on input (s, T, y), may make queries of the form $Q'(t, z)$ where $t \in T$,*

such that the following conditions hold:

- *(coverage) for each $s \in S$, there exists $T \subseteq S'$ such that $(s, T) \in U$, and*
- *(correctness) for each $(s, T) \in U$, it holds (for each $y \in \Sigma^*$) that $Q(s, y) = r(s, T, y)$.*

Theorem 3.2 *Counting slice reducibility is transitive.*

The following theorem shows that, from a counting slice reduction, one can obtain complexity results for the corresponding parameterized problems.

Theorem 3.3 *Let $Q[S]$ and $Q'[S']$ be case problems. Suppose that $Q[S]$ counting slice reduces to $Q'[S']$, and that both S and S' are computable. Then* param-$Q[S]$ *counting FPT-reduces to* param-$Q'[S']$.

3.2. Classification of pp-formulas

We present the complexity classification of pp-formulas presented in [CM14]. The following definitions are adapted from that article. Let $(\mathbf{A}, S)$ be a prenex pp-formula, let $\mathbf{D}$ be a core thereof, and let $G = (D, E)$ be the graph of $\mathbf{D}$. A *free component* of $(\mathbf{A}, S)$ is a graph of the form $G[V']$ where there exists $V \subseteq D$ that is the vertex set of a component of $G[D \backslash S]$ and V' is the union of V with all vertices in S having an edge to V. Define contract$(\mathbf{A}, S)$ to be the graph obtained by starting from $G[S]$ and adding an edge between two vertices that appear together in a free component of $(\mathbf{A}, S)$.

Let Φ be a set of prenex pp-formulas. Let us say that Φ satisfies the *contraction condition* if the graphs in contract(Φ) are of bounded treewidth. Let us say that Φ satisfies the *tractability condition* if it satisfies the contraction condition and, in addition, the cores of Φ are of bounded treewidth.

Theorem 3.4 *[CM14] Let Φ be a set of prenex pp-formulas that satisfies the tractability condition. Then, the restriction of p-*count *to $\Phi \times \Sigma^*$ is an FPT-computable partial function.*

Theorem 3.5 *[CM14] Let Φ be a set of prenex pp-formulas of bounded arity that does not satisfy the tractability condition.*

1. *If Φ satisfies the contraction condition, then* count$[\Phi]$ *and* Clique$[\mathbb{N}]$ *are equivalent under counting slice reductions.*
2. *Otherwise, there is a counting slice reduction from* #Clique$[\mathbb{N}]$ *to* count$[\Phi]$.

We say that a set of formulas Φ has *bounded arity* if there exists a constant $k \geqslant 1$ that upper bounds the arity of each relation symbol appearing in a formula in Φ.

4. Equivalence theorem statement

The following theorem, which we call the *equivalence theorem*, is our first main result; it is used to derive our complexity trichotomy on ep-formulas from the known complexity trichotomy on pp-formulas (which was presented in Section 3.2).

Theorem 4.1 *Let Φ be a set of ep-formulas. There exists a set Φ^+ of prenex pp-formulas such that the counting case problems $\mathsf{count}[\Phi]$ and $\mathsf{count}[\Phi^+]$ are equivalent under counting slice reductions.*

5. Classification theorem

In this section, we state our trichotomy theorem on the complexity of counting answers to ep-formulas, and show how to prove it using the equivalence theorem (Theorem 4.1). For a set Φ of ep-formulas, let us use $p\text{-}\mathsf{count} \upharpoonright \Phi$ to denote the parameterized problem which is equal to $p\text{-}\mathsf{count}$ on $\Phi \times \Sigma^*$, and is equal to 0 elsewhere (and has the parameterization of $p\text{-}\mathsf{count}$); note that this is equivalent to $\mathsf{param\text{-}count}[\Phi]$.

Theorem 5.1 *Let Φ be a computable set of ep-formulas of bounded arity, and let Φ^+ be the set of pp-formulas given by Theorem 4.1.*

1. *If Φ^+ satisfies the tractability condition, then $(p\text{-}\mathsf{count} \upharpoonright \Phi)$ is in FPT.*
2. *If Φ^+ does not satisfy the tractability condition but satisfies the contraction condition, then $(p\text{-}\mathsf{count} \upharpoonright \Phi)$ is equivalent to p-Clique under counting FPT-reduction.*
3. *Otherwise, there is a counting FPT-reduction from p-#Clique to $(p\text{-}\mathsf{count} \upharpoonright \Phi)$.*

Proof. For (1), we use the counting slice reduction (U, r) from $\mathsf{count}[\Phi]$ to $\mathsf{count}[\Phi^+]$ given by Theorem 4.1. In particular, given an instance $(\phi, \mathbf{B})$ of $(p\text{-}\mathsf{count} \upharpoonright \Phi)$, the algorithm for r is invoked on $(\phi, \phi^+, \mathbf{B})$, where ϕ^+ is as defined in Section 6.4; queries to $\mathsf{count}(\psi, \mathbf{B})$ where $\psi \in \Phi^+$ are resolved according to the algorithm of Theorem 3.4.

For (2) and (3), we make use of the result (Theorem 4.1) that the problems $\mathsf{count}[\Phi]$ and $\mathsf{count}[\Phi^+]$ are equivalent under counting slice reductions. For (2), we have from Theorem 3.5 that $\mathsf{count}[\Phi^+]$ and Clique$[\mathbb{N}]$ are equivalent under counting slice reductions. Hence, Clique$[\mathbb{N}]$ and $\mathsf{count}[\Phi]$ are equivalent under counting slice reductions, and the result follows from Theorem 3.3. For (3), we have from Theorem 3.5 that there is a counting slice reduction from #Clique$[\mathbb{N}]$ to $\mathsf{count}[\Phi^+]$, and hence from #Clique$[\mathbb{N}]$ to $\mathsf{count}[\Phi]$; the result then follows from Theorem 3.3. □

Let us remark that when case (2) applies, a consequence of this theorem is that the problem $(p\text{-}\mathsf{count} \upharpoonright \Phi)$ is not in FPT unless W[1] is in FPT, since p-Clique is W[1]-complete; similarly, when case (3) applies, the problem $(p\text{-}\mathsf{count} \upharpoonright \Phi)$ is not in FPT unless ♯W[1] is in FPT, since p-#Clique is ♯W[1]-complete.

6. Proof of equivalence theorem

In this section, we generally assume pp-formulas to be prenex.

6.1. Counting equivalence

Definition 6.1 *Define two fo-formulas* $\phi(V), \phi'(V')$ *to be* counting equivalent *if they are over the same vocabulary* τ *and for each finite* τ*-structure* $\mathbf{B}$ *it holds that* $|\phi(\mathbf{B})| = |\phi'(\mathbf{B})|$.

In this subsection, we characterize counting equivalence for pp-formulas, using the following notion.

Definition 6.2 *Define two fo-formulas* $\phi(V), \phi'(V')$ *to be* renaming equivalent *if one can be made logically equivalent to the other by a sequence of renaming steps. By a* renaming step*, we mean that either a liberal variable* u *or an occurrence of a variable* v *appearing right after a quantifier is changed to a variable* w*. In the first case, we understand that all free occurrences of* u *are changed to* w*; in the second case, we understand that all occurrences of* v *that are free at the quantifier location are changed to* w*, and also that* w *is not free at the quantifier location.*

The following is the main theorem of this subsection.

Theorem 6.3 *Two pp-formulas* $\phi_1(S_1), \phi_2(S_2)$ *are counting equivalent if and only if they are renaming equivalent.*

Before we prove Theorem 6.3, we show some results that will be helpful in the proof. We start off with an simple observation.

Observation 6.4 *Let* ϕ *and* ϕ' *be counting equivalent pp-formulas. Then* $|\mathrm{lib}(\phi)| = |\mathrm{lib}(\phi')|$.

Proposition 6.5 *Let* ϕ *and* ϕ' *be two pp-formulas with corresponding pairs* $(\mathbf{A}, S)$ *and* $(\mathbf{A}', S')$*, respectively. If there are surjections* $h : \mathrm{lib}(\phi) \to \mathrm{lib}(\phi')$ *and* $h' : \mathrm{lib}(\phi') \to \mathrm{lib}(\phi)$ *that can be extended to homomorphisms* $\bar{h} : \mathbf{A} \to \mathbf{A}'$ *and* $\bar{h}' : \mathbf{A}' \to \mathbf{A}$*, respectively, then* ϕ *and* ϕ' *are renaming equivalent.*

Proof.[of Theorem 6.3] Renaming equivalent formulas are trivially counting equivalent, because renaming variables does not change the number of satisfying assignments.

For the other direction, let $\phi_1(S_1)$ and $\phi_2(S_2)$ be two pp-formulas over a common vocabulary τ that are not renaming equivalent. Let $(\mathbf{A}_1, S_1)$ and $(\mathbf{A}_2, S_2)$ be the corresponding pairs. By way of contradiction, assume that ϕ_1 and ϕ_2 are counting equivalent. If $|\mathrm{lib}(\phi_1)| \neq |\mathrm{lib}(\phi_2)|$ we are done by Proposition 6.4. So we may assume, after potentially renaming some variables, that $\mathrm{lib}(\phi_1) = \mathrm{lib}(\phi_2) =: S$.

When $\mathbf{C}$, $\mathbf{D}$ are structures with $S \subseteq C \cap D$, define $\hom(\mathbf{C}, \mathbf{D}, S)$ to be the set of mappings from S to D that can be extended to a homomorphism from $\mathbf{C}$ to $\mathbf{D}$; denote by $\mathrm{surj}(\mathbf{C}, \mathbf{D}, S)$ the surjections $h : S \to S$ that lie in $\hom(\mathbf{C}, \mathbf{D}, S)$. With Proposition 6.5 we may w.l.o.g. assume that $\mathrm{surj}(\mathbf{A}_1, \mathbf{A}_2, S) = \varnothing$. For $T \subseteq S$ let $\hom_T(\mathbf{A}_1, \mathbf{A}_2, S)$ be the set of mappings $h \in \hom(\mathbf{A}_1, \mathbf{A}_2, S)$ such that $h(S) \subseteq T$. By inclusion-exclusion we get

$$|\mathrm{surj}(\phi_1, \phi_2, S)| = \sum_{T \subseteq S} (-1)^{|S|-|T|} |\hom_T(\mathbf{A}_1, \mathbf{A}_2, S)|.$$

For $i \geqslant 0$ let $\hom_{i,T}(\mathbf{A}_1, \mathbf{A}_2, S)$ be the set of mappings $h \in \hom(\mathbf{A}_1, \mathbf{A}_2, S)$ such that h maps exactly i variables from S into T. Now for each $j = 1, \ldots, |S|$ we construct a new structure $\mathbf{D}_{j,T}$ over the domain $D_{j,T}$. To this end, let $a^{(1)}, \ldots, a^{(j)}$ be copies of $a \in T$ that are not in A_2. Then we set

$$D_{j,T} := \{a^{(k)} \mid a \in A_2, a \in T, k \in [j]\} \cup (A_2 \backslash T).$$

We define a mapping $B : A_2 \to \mathcal{P}(D_{j,T})$, where $\mathcal{P}(D_{j,T})$ is the power set of $D_{j,T}$, by

$$B(a) := \begin{cases} \{a^{(k)} \mid k \in [j]\}\}, & \text{if } a \in T \\ \{a\}, & \text{otherwise.} \end{cases}$$

For every relation symbol $R \in \tau$ we define

$$R^{\mathbf{D}_{T,j}} := \bigcup_{(d_1,\ldots,d_s) \in R^{\mathbf{A}_2}} B(d_1) \times \ldots \times B(d_s).$$

Then every $h \in \hom_{i,T}(\mathbf{A}_1, \mathbf{A}_2, S)$ corresponds to j^i mappings in $\hom(\mathbf{A}_1, \mathbf{D}_{j,T}, S)$. Thus for each j we get

$$\sum_{i=1}^{|S|} j^i |\hom_{i,T}(\mathbf{A}_1, \mathbf{A}_2, S)| = |\hom(\mathbf{A}_1, \mathbf{D}_{j,T}, S)|.$$

This is a linear system of equations and the corresponding matrix is a Vandermonde matrix, so $\hom_T(\mathbf{A}_1, \mathbf{A}_2, S) = \hom_{|S|,T}(\mathbf{A}_1, \mathbf{A}_2, S)$ can efficiently be computed from $|\hom(\mathbf{A}_1, \mathbf{D}, S)| = |\phi_1(\mathbf{D})|$ for some structures $\mathbf{D}$. We can similarly determine the values $|\hom_T(\mathbf{A}_2, \mathbf{A}_2, S)|$ as a function of $|\phi_2(\mathbf{D})|$ for the same structures $\mathbf{D}$. Since $|\phi_1(\mathbf{D})| = |\phi_2(\mathbf{D})|$ for every structure $\mathbf{D}$ by assumption, it follows that for every $T \subseteq S$ we have $|\hom_T(\mathbf{A}_1, \mathbf{A}_2, S)| = |\hom_T(\mathbf{A}_2, \mathbf{A}_2, S)|$. But then we have $|\mathrm{surj}(\mathbf{A}_1, \mathbf{A}_2, S)| = |\mathrm{surj}(\mathbf{A}_2, \mathbf{A}_2, S)|$. Since we have $\mathrm{surj}(\mathbf{A}_1, \mathbf{A}_2, S) = \varnothing$ and $\mathrm{id} \in \mathrm{surj}(\mathbf{A}_2, \mathbf{A}_2, S)$, this is a contradiction. Consequently, ϕ_1 and ϕ_2 are not counting equivalent. □

6.2. Semi-counting equivalence

In this subsection, we study the following relaxation of the notion of *counting equivalence*. Define two prenex pp-formulas $\phi_1(V_1)$, $\phi_2(V_2)$ to be *semi-counting equivalent* if for each finite structure $\mathbf{B}$ such that $|\phi_1(\mathbf{B})| > 0$ and $|\phi_2(\mathbf{B})| > 0$, it holds that $|\phi_1(\mathbf{B})| = |\phi_2(\mathbf{B})|$.

For each free prenex pp-formula $\phi(V)$, define $\widehat{\phi}(V)$ to be the pp-formula obtained from ϕ by removing components not having liberal variables, or put equivalently, by restricting to liberal components. The following characterization of semi-counting equivalence is the main theorem of this subsection.

Theorem 6.6 *Two free prenex pp-formulas $\phi_1(V_1), \phi_2(V_2)$ are semi-counting equivalent if and only if $\widehat{\phi_1}(V_1)$ and $\widehat{\phi_2}(V_2)$ are counting equivalent.*

We will use the following proposition in the proof of Theorem 6.6.

Proposition 6.7 *Let $\phi(V)$ be a free prenex pp-formula. Then for every structure $\mathbf{B}$ we have $\phi(\mathbf{B}) = \varnothing$ or $\phi(\mathbf{B}) = \widehat{\phi}(\mathbf{B})$.*

Proof.[of Theorem 6.6] Assume first that $\widehat{\phi_1}$ and $\widehat{\phi_2}$ are counting equivalent. Let $\mathbf{B}$ be a structure. Then if $|\phi_1(\mathbf{B})| > 0$ and $|\phi_2(\mathbf{B})| > 0$, we have by Proposition 6.7 and counting equivalence of $\widehat{\phi_1}$ and $\widehat{\phi_2}$ that $|\phi_1(\mathbf{B})| = |\widehat{\phi_1}(\mathbf{B})| = |\widehat{\phi_2}(\mathbf{B})| = |\phi_2(\mathbf{B})|$, so ϕ_1 and ϕ_2 are semi-counting equivalent.

For the other direction let now ϕ_1 and ϕ_2 be semi-counting equivalent. By way of contradiction, we assume that $\widehat{\phi_1}$ and $\widehat{\phi_2}$ are not counting equivalent. Then by definition there is a structure $\mathbf{B}$ such that $|\widehat{\phi_1}(\mathbf{B})| \neq |\widehat{\phi_2}(\mathbf{B})|$. Note that each component of $\widehat{\phi_1}$ and $\widehat{\phi_2}$ has a free variable.

Let $\mathbf{I} = \mathbf{I}_\tau$. For each $k \in \mathbb{N}$ we denote by $\mathbf{B} + k\mathbf{I}$ the structure we get from $\mathbf{B}$ by disjoint union with k copies of $\mathbf{I}$. Note that for $k > 0$ we have $|\phi(\mathbf{B} + k\mathbf{I})| > 0$ for every pp-formula ϕ. Consequently, for every $k > 0$ we have $|\phi_1(\mathbf{B} + k\mathbf{I})| = |\widehat{\phi_1}(\mathbf{B} + k\mathbf{I})|$ and $|\phi_2(\mathbf{B} + k\mathbf{I})| = |\widehat{\phi_2}(\mathbf{B} + k\mathbf{I})|$ by Proposition 6.7. By the semi-counting equivalence of ϕ_1 and ϕ_2 we also have $|\phi_1(\mathbf{B} + k\mathbf{I})| = |\phi_2(\mathbf{B} + k\mathbf{I})|$ for all $k > 0$. It follows that $|\widehat{\phi_1}(\mathbf{B} + k\mathbf{I})| = |\widehat{\phi_2}(\mathbf{B} + k\mathbf{I})|$ for $k > 0$.

Let $\phi_{1,1}, \ldots, \phi_{1,n}$ be the components of $\widehat{\phi_1}$ and let $\phi_{2,1}, \ldots, \phi_{2,m}$ be the components of $\widehat{\phi_2}$. Because every component of ϕ_1 has a free variable, we have

$$\begin{aligned} |\widehat{\phi_1}(\mathbf{B} + k\mathbf{I})| &= \sum_{J \subseteq [n]} k^{n-|J|} \prod_{j \in J} |\phi_{1,j}(\mathbf{B})| \\ &= \sum_{\ell=0}^{n} k^{n-\ell} \sum_{J \subseteq [n], |J| = \ell} \prod_{j \in J} |\phi_{1,j}(\mathbf{B})|. \end{aligned}$$

We can express $|\widehat{\phi_2}(\mathbf{B} + k\mathbf{I})|$ analogously. The expressions are polynomials in k and they are equal for every positive integer k by the observations above; thus the coefficients of the polynomials must coincide. The coefficients of k^0, namely $\prod_{j \in [n]} |\phi_{1,j}(\mathbf{B})|$ and $\prod_{j \in [m]} |\phi_{2,j}(\mathbf{B})|$, are thus equal. But then we get

$$|\widehat{\phi_1}(\mathbf{B})| = \prod_{j \in [n]} |\phi_{1,j}(\mathbf{B})| = \prod_{j \in [m]} |\phi_{2,j}(\mathbf{B})| = |\widehat{\phi_2}(\mathbf{B})|,$$

which is a contradiction to our assumption. □

Corollary 6.8 *Semi-counting equivalence is an equivalence relation (on pp-formulas).*

We now present a lemma that will be of utility.

Lemma 6.9 *Let $\phi_1(S_1), \ldots, \phi_n(S_n)$ be pp-formulas over the same vocabulary τ, with each $|S_i| > 0$. Then there is a structure $\mathbf{C}$ such that for all pp-formulas ϕ over τ we have that $|\phi(\mathbf{C})| > 0$ and for all $i, j \in [n]$ such that ϕ_i and ϕ_j are not semi-counting equivalent we have $|\phi_i(\mathbf{C})| \neq |\phi_j(\mathbf{C})|$.*

6.3. The all-free case

The aim of this subsection is the proof of Theorem 4.1 in the special case of all-free ep-formulas. For every $\phi \in \Phi$ we define a set ϕ^* of free pp-formulas; then, we define $\Phi^* = \cup_{\phi \in \Phi} \phi^*$. Let $\phi = \phi_1 \vee \ldots \vee \phi_s$ where the ϕ_i are free pp-formulas. By inclusion-exclusion we have for every structure $\mathbf{B}$ that

$$\begin{aligned} |\phi(\mathbf{B})| &= \sum_{J \in [s]} (-1)^{|J|+1} |(\bigwedge_{j \in J} \phi_j)(\mathbf{B})| \\ &= \sum_{J \in [s]} (-1)^{|J|+1} |\phi_J(\mathbf{B})|, \end{aligned} \tag{1}$$

where $\phi_J = \bigwedge_{j \in J} \phi_j$ are pp-formulas. Now iteratively do the following: If there are two summands $c|\phi_J(\mathbf{B})|$ and $c'|\phi_{J'}(\mathbf{B})|$ such that ϕ_J and $\phi_{J'}$ are counting equivalent, delete both summands and add $(c + c')|\phi_J|$ to the sum. When this operation can no longer be applied, delete all summands with coefficient zero. The pp-formulas that remain in the sum form the set ϕ^*. We obtain the following proposition.

Proposition 6.10 *There exists an algorithm that, given an all-free ep-formula ϕ, outputs the set $\phi^* := \{\phi_1^*, \ldots, \phi_\ell^*\}$ of free pp-formulas as defined above and coefficients $c_1, \ldots, c_\ell \in \mathbb{Z} \backslash \{0\}$ such that for every structure $\mathbf{B}$, $|\phi(\mathbf{B})| = \sum_{i=1}^{\ell} c_i |\phi_i^*(\mathbf{B})|$.*

We will also require the following two facts for our proof.

Proposition 6.11 *Let $\phi(S)$ and $\phi'(S')$ be two semi-counting equivalent free pp-formulas that are not counting equivalent and let $(\mathbf{A}, S)$ and $(\mathbf{A}', S')$ be the structures of ϕ and ϕ', respectively. Then $\mathbf{A}$ and $\mathbf{A}'$ are not homomorphically equivalent.*

Lemma 6.12 *There is an oracle* FPT*-algorithm for the following problem: Given a set $\phi_1, \ldots, \phi_s$ of semi-counting equivalent free pp-formulas that are pairwise not counting equivalent, a sequence $c_1, \ldots, c_s \in \mathbb{Z} \backslash \{0\}$, and a structure $\mathbf{B}$, the algorithm computes $|\phi_i(\mathbf{B})|$ for every $i \in [s]$, and may make calls to an oracle that provides $\sum_{i=1}^{s} c_i \cdot |\phi_i(\mathbf{B}')|$ upon being given a structure $\mathbf{B}'$. Here, the ϕ_i with the c_i constitute the parameter.*

We can now prove Theorem 4.1 for all-free ep-formulas.

Theorem 6.13 *Let Φ be a set of all-free ep-formulas. There exists a set Φ^* of free prenex pp-formulas such that the counting case problems* $\mathsf{count}[\Phi]$ *and* $\mathsf{count}[\Phi^*]$ *are equivalent under counting slice reductions.*

Proof. The reduction from $\mathsf{count}[\Phi]$ to $\mathsf{count}[\Phi^*]$ is straightforward: The relation U is given by $\{(\phi, \phi^*) \mid \phi \in \Phi\}$. Obviously, this satisfies the coverage condition. Then the oracle-FPT-algorithm to compute $\phi(\mathbf{B})$ given ϕ, ϕ^* and $\mathbf{B}$ first computes all of the $|\phi_i^*(\mathbf{B})|$ by oracle calls and then uses Proposition 6.10. This completes the reduction.

For the other direction, let $\phi' \in \Phi^*$. We set $U := \{(\phi', \{\phi\}) \mid \phi \in \Phi, \phi' \in \phi^*\}$. Given ϕ', ϕ and $\mathbf{B}$, we compute $|\phi'(\mathbf{B})| := r(\phi', \{\phi\}, \mathbf{B})$ as follows: Let $\phi_1^*, \ldots, \phi_s^*$ be the equivalence classes of ϕ^* with respect to semi-counting equivalence. Now choose a strucuture $\mathbf{C}$ as in Lemma 6.9. Then for $\psi, \psi' \in \phi^*$ we have $|\psi(\mathbf{C})| \neq |\psi'(\mathbf{C})|$ if ψ and ψ' are from different equivalence classes with respect to semi-counting equivalence, and otherwise $|\psi(\mathbf{C})| = |\psi'(\mathbf{C})| > 0$.

Fix for each $j \in [s]$ a formula in ϕ_j^* and call it ψ_j. Moreover, denote by c_ψ the coefficiencent of ψ in Proposition 6.10. Using this notation and Proposition 6.10 we get for every $\ell \in \mathbb{N}$ that $|\phi(\mathbf{B} \times \mathbf{C}^\ell)| = \sum_{j=1}^s |\psi_j(\mathbf{C})|^\ell (\sum_{\psi \in \phi_j^*} c_\psi |\psi(\mathbf{B})|)$. Note that this is a linear equation with coefficients $|\psi_j(\mathbf{C})|^\ell$ which can be computed by brute force. Letting ℓ range from 0 to $s-1$ thus yields a system of linear equations whose coefficient matrix is a Vandermonde matrix. Consequently, with s oracle calls we can compute the $\sum_{\psi \in \phi_j^*} c_\psi |\psi(\mathbf{B})|$ for each j. Now we use Lemma 6.12 to compute $\phi'(\mathbf{B})$. □

6.4. The general case

We may assume that each ep-formula $\phi \in \Phi$ is normalized. For each ep-formula ϕ, define ϕ_{af} to be the all-free part of ϕ, that is, the disjunction of the ϕ-disjuncts that are free; define Φ_{af} to be $\{\phi_{\text{af}} \mid \phi \in \Phi\}$; and, define ϕ_{af}^- to be the formulas in ϕ_{af}^* that do not logically entail a sentence disjunct of ϕ. We define ϕ^+ to be the union of ϕ_{af}^- and the set containing each pp-sentence disjunct of ϕ; and, we define Φ^+ to be $\cup_{\phi \in \Phi} \phi^+$.

The idea of the proof of Theorem 4.1 is as follows. (See Section H for the full proof.) The counting slice reduction from $\mathsf{count}[\Phi]$ to $\mathsf{count}[\Phi^+]$ has U as the set of pairs (ϕ, ϕ^+) where ϕ is a normalized ep-formula; r on $(\phi(V), \phi^+, \mathbf{B})$ behaves as follows. First, it checks if there is a sentence disjunct θ of ϕ that is true on $\mathbf{B}$; if so, it outputs $|V|^{|B|}$; otherwise, it makes use of the counting slice reduction from $\mathsf{count}[\Phi_{\text{af}}]$ to $\mathsf{count}[\Phi_{\text{af}}^*]$. The counting slice reduction from $\mathsf{count}[\Phi^+]$ to $\mathsf{count}[\Phi]$ has U as the set $\{(\psi, \{\phi\}) \mid \psi \in \phi^+\}$; r on $(\psi(V), \phi(V), \mathbf{B})$ is defined as follows. When $\psi \in \phi_{\text{af}}^-$, the counting slice reduction (U', r') from $\mathsf{count}[\Phi_{\text{af}}^*]$ to $\mathsf{count}[\Phi_{\text{af}}]$ is used to determine $|\psi(\mathbf{B})|$; this is performed by passing to r' a treated version of $\mathbf{B}$, on which no sentence disjunct of ϕ may hold. When ψ is a sentence disjunct of ϕ, an oracle query is made to obtain the count of ϕ on a treated version of $\mathbf{B}$; on this treated version, it is proved that all assignments satisfy ϕ if and only if $\mathbf{B} \models \psi$.

7. $\sharp$-logic

7.1. Syntax

Syntactically, $\sharp$-logic consists of $\sharp$-*formulas*, each of which has an associated set of free variables as well as an associated set of *closed* variables. When ϕ is a $\sharp$-formula with free variables V, evaluating ϕ with respect to a structure $\mathbf{B}$ and an assignment $f : V \to B$ returns an integer value, as opposed to a propositional value (as for a fo-formula). We define $\sharp$-formulas inductively, as follows.

- $C(\phi, L)$ is a $\sharp$-formula if ϕ is a fo-formula and $L \supseteq \text{free}(\phi)$.

  Define $\text{free}(C\phi) = L$ and $\text{closed}(C\phi) = \varnothing$.

- $PV\phi$ is a $\sharp$-formula if ϕ is a $\sharp$-formula and V is a set of variables with $V \cap \text{closed}(\phi) = \varnothing$.

  Define $\text{free}(PV\phi) = \text{free}(\phi)\backslash V$ and $\text{closed}(PV\phi) = V \cup \text{closed}(\phi)$.

- $EV\phi$ is a $\sharp$-formula if ϕ is a $\sharp$-formula and V is a set of variables with $V \cap (\text{free}(\phi) \cup \text{closed}(\phi)) = \varnothing$.

  Define $\text{free}(EV\phi) = V \cup \text{free}(\phi)$ and $\text{closed}(EV\phi) = \text{closed}(\phi)$.

- $\phi \times \phi'$ is a $\sharp$-formula if ϕ and ϕ' are $\sharp$-formulas with $\text{free}(\phi) = \text{free}(\phi')$ and $\text{closed}(\phi) \cap \text{closed}(\phi') = \varnothing$.

  Define $\text{free}(\phi \times \phi') = \text{free}(\phi)$ and $\text{closed}(\phi \times \phi') = \text{closed}(\phi) \cup \text{closed}(\phi')$.

- $\phi + \phi'$ is a $\sharp$-formula if ϕ and ϕ' are $\sharp$-formulas with $\text{free}(\phi) = \text{free}(\phi')$.

  Define $\text{free}(\phi + \phi') = \text{free}(\phi)$ and $\text{closed}(\phi + \phi') = \text{closed}(\phi) \cup \text{closed}(\phi')$.

- n is a $\sharp$-formula if $n \in \mathbb{Z}$.

  Define $\text{free}(n) = \varnothing$ and $\text{closed}(n) = \varnothing$.

A formula $C(\phi, L)$ can be thought of as the casting of a fo-formula ϕ into a $\sharp$-formula; the P quantifier can be thought of as projecting or closing variables; and the E quantifier can be thought of as expanding the set of free variables. The connectives $\times$ and $+$ perform the usual arithmetic operations.

Let ψ be a $\sharp$-formula. We define a subformula of ψ in the natural fashion; note that when $C(\phi, L)$ is a $\sharp$-formula, ϕ and its subformulas are considered to be subformulas of

$C(\phi, L)$. We define $\mathsf{width}(\psi)$ to be the maximum of $|\text{free}(\theta)|$ over all subformulas θ of ψ, and $\sharp$-$\mathsf{width}(\psi)$ to be the maximum of $|\text{free}(\theta)|$ over all $\sharp$-subformulas θ of ψ. We say that ψ is a $\sharp$-sentence if $\text{free}(\psi) = \emptyset$.

We define a $\sharp\mathsf{PP}$-*formula* to be a $\sharp$-formula where, in each subformula of the form $C(\phi, L)$, ϕ is a pp-formula; the notion of $\sharp\mathsf{EP}$-*formula* is defined analogously.

7.2. Semantics

Let $\mathbf{B}$ be a structure, let ϕ be a $\sharp$-formula over the signature of $\mathbf{B}$, let $h : \text{free}(\phi) \to B$ be an assignment, and let $c \in \mathbb{Z}$.

We define the relation $\mathbf{B}, h, c \models \phi$ as follows. For each such structure $\mathbf{B}$, $\sharp$-formula ϕ, and assignment $h : \text{free}(\phi) \to B$ there is a unique c such that $\mathbf{B}, h, c \models \phi$; we will use $[\mathbf{B}, \phi]$ as notation for the mapping taking an assignment $h : \text{free}(\phi) \to B$ to this unique c.

- When $C(\phi, L)$ is a $\sharp$-formula,

  $[\mathbf{B}, \phi](h) = 1$ if $\mathbf{B}, h \models \phi$, and $[\mathbf{B}, \phi](h) = 0$ otherwise.

  That is, $\mathbf{B}, h, 1 \models C(\phi, L)$ if $\mathbf{B}, h \models \phi$; and, $\mathbf{B}, h, 0 \models C(\phi, L)$ otherwise.

- When $PV\phi$ is a $\sharp$-formula,

  $[\mathbf{B}, PV\phi](h) = \sum_{h'} [\mathbf{B}, \phi](h')$, where the sum is over all extensions $h' : \text{free}(\phi) \cup V \to B$ of h.

- When $EV\phi$ is a $\sharp$-formula,

  $[\mathbf{B}, EV\phi](h) = [\mathbf{B}, \phi](h \upharpoonright \text{free}(\phi))$.

- When $\phi \times \phi'$ is a $\sharp$-formula,

  $[\mathbf{B}, \phi \times \phi'](h) = [\mathbf{B}, \phi](h) \cdot [\mathbf{B}, \phi'](h)$.

- When $\phi + \phi'$ is a $\sharp$-formula,

  $[\mathbf{B}, \phi + \phi'](h) = [\mathbf{B}, \phi](h) + [\mathbf{B}, \phi'](h)$.

- When n is a $\sharp$-formula,

  $[\mathbf{B}, n](h) = n$.

We consider two $\sharp$-formulas ϕ, ϕ' with $\text{free}(\phi) = \text{free}(\phi')$ to be *logically equivalent* if for each structure $\mathbf{B}$, it holds that $[\mathbf{B}, \phi] = [\mathbf{B}, \phi']$. A $\sharp$-sentence ψ *represents* or is a *representation* of a fo-formula $\phi(V)$ if for each finite structure $\mathbf{B}$, it holds that $|\phi(\mathbf{B})| = [\mathbf{B}, \psi](\epsilon)$, where ϵ is the empty assignment. For simplicity, when ψ is a $\sharp$-sentence, we will typically write $[\mathbf{B}, \psi]$ in place of $[\mathbf{B}, \psi](\epsilon)$. Observe that, for each fo-formula $\phi(V)$, the $\sharp$-sentence $PVC(\phi, V)$ is a representation of $\phi(V)$.

Example 7.1 Define the formula $\phi(x_0, x_1, x_2, y_0, y_1, y_2)$ as $\phi_0 \wedge \phi_1 \wedge \phi_2$, where $\phi_i = \exists z_i T_i(x_i, x_{i+1}, y_i, z_i)$ for each $i \in \{0, 1, 2\}$; here, the quantity $i + 1$ appearing in x_{i+1} is computed modulo 3. These formulas are over the vocabulary $\{T_0, T_1, T_2\}$ having three relation symbols, each of arity 4.

Define $\psi_i = P\{y_i\}C(\phi_i, \{x_0, x_1, x_2, y_i\})$ for each $i \in \{0, 1, 2\}$. Observe that when $\mathbf{B}$ is a structure and $h : \{x_0, x_1, x_2\} \to B$ is a map, $[\mathbf{B}, \psi_i](h)$ gives the number of extensions $h' : \{x_0, x_1, x_2, y_i\} \to B$ of h satisfying ϕ_i on $\mathbf{B}$. We have $\text{free}(\psi_0) = \text{free}(\psi_1) = \text{free}(\psi_2) = \{x_0, x_1, x_2\}$ and $\text{closed}(\psi_i) = \{y_i\}$. Now consider $\psi = (\psi_0 \times \psi_1) \times \psi_2$. It can be verified that, for a structure $\mathbf{B}$ and a map $h : \{x_0, x_1, x_2\} \to B$, $[\mathbf{B}, \psi](h)$ gives the number of extensions $h' : \{x_0, x_1, x_2, y_0, y_1, y_2\} \to B$ of h satisfying ϕ on $\mathbf{B}$. It follows that the $\sharp$-sentence $P\{x_0, x_1, x_2\}\psi$ is a representation of ϕ. Note that this representation of ϕ has width equal to $\max(3, \mathsf{width}(\psi_0), \mathsf{width}(\psi_1), \mathsf{width}(\psi_2)) = 4$.

Consider (as an example) the subformula $C(\phi_0, \{x_0, x_1, x_2, y_0\})$ of ψ_0. It holds that $x_2 \notin \text{free}(\phi_0)$, and so $E\{x_2\}C(\phi_0, \{x_0, x_1, y_0\})$ is a $\sharp$-formula and is logically equivalent to $C(\phi_0, \{x_0, x_1, x_2, y_0\})$. □

7.3. Results

We first study representations of pp-formulas, obtaining the following theorems.

Theorem 7.2 *Let Φ be a class of prenex pp-formulas.*

- *The class Φ satisfies the tractability condition if and only if there exists $k \geqslant 1$ such that each formula in Φ has a $\sharp$PP-representation ϕ' having $\mathsf{width}(\phi') \leqslant k$.*
- *The class Φ satisfies the contraction condition if and only if there exists $k \geqslant 1$ such that each formula in Φ has a $\sharp$PP-representation ϕ' having $\sharp\text{-}\mathsf{width}(\phi') \leqslant k$.*

Theorem 7.3 *There exists an algorithm that, given a prenex pp-formula ϕ, outputs a $\sharp$PP-representation ψ of ϕ of minimum width.*

Building on this understanding of pp-formulas, we are then able to achieve general versions of these theorems for ep-formulas.

Theorem 7.4 *Let Φ be a class of ep-formulas.*

- *The class Φ^+ satisfies the tractability condition if and only if there exists $k \geqslant 1$ such that each formula in Φ has a $\sharp$EP-representation ϕ' having $\mathsf{width}(\phi') \leqslant k$.*
- *The class Φ^+ satisfies the contraction condition if and only if there exists $k \geqslant 1$ such that each formula in Φ has a $\sharp$EP-representation ϕ' having $\sharp\text{-}\mathsf{width}(\phi') \leqslant k$.*

Theorem 7.5 *There exists an algorithm that, given an ep-formula ϕ, outputs a $\sharp$EP-representation ψ of ϕ of minimum width.*

8. pp-formulas and ♯-logic

In this section, we introduce a new width measure of pp-formulas which we call quantifier aware width and show that it is related to the width of ♯PP-formulas.

We assume all tree decompositions of pp-formulas to be nice. So let $(T, (B_t)_{t \in T})$ be a nice tree decomposition of a pp-formula ϕ. For every variable x of ϕ let $\mathsf{top}(x)$ be the vertex t of T that is highest in T such that $x \in B_t$. We call a tree decomposition of ϕ *quantifier aware* if for every free-component C of ϕ and for all $x \in V(C) \setminus \text{free}(\phi)$ and all $y \in V(C) \cap \text{free}(\phi)$, we have that $\mathsf{top}(y)$ is on the path from $\mathsf{top}(x)$ to the root of T. We call the *quantifier aware width* of a pp-formula ϕ, denoted by $\mathsf{qaw}(\phi)$, the minimal treewidth of a quantifier aware tree decomposition of ϕ plus 1.

We first show that $\mathsf{qaw}(\phi)$ is essentially equivalent to the combination of $\mathsf{tw}(\phi)$ and $\mathsf{tw}(\mathsf{contract}(\phi))$ for every pp-formula ϕ. Consequently, quantifier aware treewidth allows us to characterize tractable classes of pp-formulas for counting.

Lemma 8.1 *For every pp-formula ϕ we have*

$$\begin{aligned}&\max\{\mathsf{tw}(\phi), \mathsf{tw}(\mathsf{contract}(\phi))\} + 1\\ \leqslant &\mathsf{qaw}(\phi) \leqslant \mathsf{tw}(\phi) + \mathsf{tw}(\mathsf{contract}(\phi)) + 1.\end{aligned}$$

The main idea of the proof is that forcing the free variables of a free component to appear above the quantified variables has a very similar effect as connecting them to a clique in the construction of the $\mathsf{contract}(\phi)$. So from a quantifier aware decomposition we get a decomposition of $\mathsf{contract}(\phi)$ by restricting to the tree decomposition to the free variables. For the other direction, we can add the quantified variables to a tree decomposition of $\mathsf{contract}(\phi)$ in a straightforward way.

We now show how to compute quantifier aware treewidth which will allow us to compute ♯PP-formulas of optimal width.

Lemma 8.2 *Computing quantifier aware tree decompositions of minimal width of pp-formulas is fixed-parameter tractable, parameterized by the quantifier aware width.*

The idea of the proof is to add some edges to the graph of ϕ in such a way that the treewidth of the resulting graph is exactly $\mathsf{qaw}(\phi)$. Then we apply standard algorithms for computing treewidth.

We call a ♯PP-formula ϕ *basic* if it does not contain + nor subformulas of the form n, where $n \in \mathbb{Z}$. We now show that basic ♯PP-formulas correspond very closely to pp-formulas.

Lemma 8.3 *a) For every basic ♯PP-sentence ϕ' there exists a pp-formula ϕ that ϕ' represents such that $\mathsf{width}(\phi') \geqslant \mathsf{qaw}(\phi)$, and*
$\sharp\text{-}\mathsf{width}(\phi) \geqslant \mathsf{tw}(\mathsf{contract}(\phi)) + 1$.

b) For every pp-formula ϕ there exists a basic ♯PP-sentence ϕ' that represents ϕ with $\mathsf{width}(\phi') \leqslant \mathsf{qaw}(\phi)$, and
$\sharp\text{-}\mathsf{width}(\phi) \leqslant \mathsf{tw}(\mathsf{contract}(\phi)) + 1$.

The proof relies on the observation that the condition on $\mathsf{top}(x)$ in quantifier aware tree decompositions corresponds closely to the fact that free variables can only be closed by a P-quantifier in $\sharp$PP-formulas after the contained pp-formula has been casted by a C-quantifier. With this in mind, a $\sharp$PP-formula is transformed into a pp-formula by making use of and inducting on the $\sharp$PP-formula's structure (viewed as a tree). The other direction is similar.

9. ep-formulas and $\sharp$-logic

9.1. Normal forms of $\sharp$-formulas

We call a $\sharp$PP-formula *constant* if it is only constructed from constants in $\mathbb{Z}$, $\times$, and P- and E-quantifiers. We call a $\sharp$PP-formula *flat* if it is of the form $\sum_{i\in[\ell]} \psi_i \times \phi_i$ where where the $\psi_1, \ldots, \psi_n$ are constant and $\phi_1, \ldots, \phi_\ell$ are basic $\sharp$PP-formulas.

We establish the following normalization result.

Lemma 9.1 *There exists an algorithm that computes, for a given $\sharp$EP-formula ϕ, a logically equivalent flat $\sharp$PP-formula ϕ' such that $\mathsf{width}(\phi') \leqslant \mathsf{width}(\phi)$.*

Lemma 9.2 *There exists an algorithm that computes, for a given $\sharp$EP-formula ϕ of the form $C(\psi, L)$, a logically equivalent $\sharp$PP-formula ϕ' such that $\mathsf{width}(\phi') \leqslant \mathsf{width}(\phi)$.*

Proof. In a first step we transform ψ into a logically equivalent disjunctive ep-formula ψ^d; this can be done without increasing width [Che14a]. So let ψ^d be the disjunction $\bigvee_{i=1}^{s} \psi_i$ where the ψ_i are pp-formulas. Then we claim that ϕ is logically equivalent to $\phi' = \sum_{J\subseteq[s], J\neq\varnothing}(EL(-1)^{|J|+1}) \prod_{i\in J} C(\psi_i, L)$. First note that this is a well-formed $\sharp$-formula, because for all additions and multiplications the free variables of all operands are L. It remains to show that ϕ' is logically equivalent to ϕ. So fix $\mathbf{B}$ and $h : L \to B$.

If h does not satisfy ψ, then $[\mathbf{B}, \phi](h) = 0$. Since h does not satisfy any ψ_i, it is easy to see that $[\mathbf{B}, \phi](h) = 0$ as well.

Let now h satisfy ψ, say it satisfies the disjuncts $\psi_1, \ldots, \psi_\ell$. By definition $[\mathbf{B}, \phi](h) = 1$. Moreover,

$$\begin{aligned}
[\mathbf{B}, \phi'](h) = & \sum_{J\subseteq[s], J\neq\varnothing} (-1)^{|J|+1} \prod_{i\in J} [B, C(\psi_i, L)] \\
= & \sum_{J\subseteq[\ell], J\neq\varnothing} (-1)^{|J|+1} \\
= & \, \ell - \binom{\ell}{2} + \binom{\ell}{3} - \ldots \pm \binom{\ell}{\ell} \\
& - (1-1)^{\ell} + 1 \\
= & \, 1.
\end{aligned}$$

It is readily seen that the width of ϕ' is not bigger than that of ϕ. □

We call an $\sharp$PP-formula $+$-*free* if it does not contain $+$.

Lemma 9.3 *There exists an algorithm that computes, for a given $\sharp$EP-formula ϕ, a logically equivalent $\sharp$PP-formula ϕ' of the form $\sum_{i=1}^{s} \phi_i$ where the ϕ_i are $+$-free such that* $\mathsf{width}(\phi') \leqslant \mathsf{width}(\phi)$.

Proof. (idea) The proof is by straightforward induction on the structure of ϕ, pushing all occurences of $+$ up in the formula; for instance, one proves that $PV(\psi_1 + \psi_2)$ is logically equivalent to $PV\psi_1 + PV\psi_2$. The base case $\phi = C(\psi, L)$ is Lemma 9.2. □

The proof of the following lemma is by a straightforward induction.

Lemma 9.4 *There exists an algorithm that computes, for a given constant $\sharp$PP-formula ϕ, a logically equivalent $\sharp$PP-formula $\phi' = EV_1PV_2n$ with $n \in \mathbb{Z}$ such that* $\mathsf{width}(\phi') \leqslant \mathsf{width}(\phi)$.

Lemma 9.5 *There exists an algorithm that computes, for a given $+$-free $\sharp$PP-formula ϕ, a logically equivalent $\sharp$PP-formula $\phi' = \psi_1 \times \psi_2$ where ψ_1 is constant and ψ_2 is basic such that* $\mathsf{width}(\phi') \leqslant \mathsf{width}(\phi)$.

Proof. (sketch) The proof is again straightforward induction in the style of Lemma 9.3. We only the the case of P-quantifiers which is the only case that is not completely clear from the definition.

So let $\phi = PV\phi'$ where $\phi' = \phi_1 \times \phi_2$ such that ϕ_1 is constant and ϕ_2 is basic. Note that by Lemma 9.4 we may assume that $\phi_1 = PV_1EV_2n$ for some $n \in \mathbb{Z}$. We claim that ϕ is logically equivalent to $\phi_1' \times PV\phi_2$ where $\phi_1' = PV_1E(V_2 \backslash V)$. To see this, consider a structure $\mathbf{B}$ and an assignment to ϕ. Then $[\mathbf{B}, PV\phi'](h) = \sum_{h'} [\mathbf{B}, \phi_1 \times \phi_2](h') = \sum_{h'} ([\mathbf{B}, \phi_1](h') \cdot [\mathbf{B}, \phi_2](h'))$ where the h' are as in the definition. Now choose an arbitrary assignment h'' to ϕ', then $[\mathbf{B}, \phi_1](h'') = [\mathbf{B}, \phi_1](h') = [\mathbf{B}, \phi_1'](h)$ for all h'. Consequently, $[\mathbf{B}, PV\phi'](h) = [\mathbf{B}, \phi_1](h'') \cdot \sum_{h'} [\mathbf{B}, \phi_2](h') = [\mathbf{B}, \phi_1'](h) \cdot \sum_{h'} [\mathbf{B}, \phi_2](h') = [\mathbf{B}, \phi_1'](h) \cdot [\mathbf{B}, PV\phi_2](h) = [\mathbf{B}, \phi_1' \times PV\phi_2](h)$. □

Proof.[of Lemma 9.1] First use Lemma 9.3 to turn ϕ into a sum of $+$-free $\sharp$PP-formulas. Then apply Lemma 9.5 to each of the summands. □

9.2. Independence of pp-formulas

In the scope of this subsection, define a *linear combination* to be an expression of the form $\sum_{i=1}^{m} c_i |\phi_i(V_i)|$, where each c_i is a non-zero integer and the $\phi_i(V_i)$ are pp-formulas that are pairwise not counting equivalent. Each linear combination ℓ naturally induces a mapping $\ell(\cdot)$ from finite structures to $\mathbb{Z}$. The following theorem will be key for our understanding of equivalence of $\sharp$EP-formulas.

Theorem 9.6 *For any non-empty linear combination ℓ, there exists a finite structure $\mathbf{D}$ such that $\ell(\mathbf{D}) \neq 0$.*

We begin by defining a notion of applying a univariate polynomial to a structure. Let p be a univariate polynomial with positive integer coefficients and variable X. Fix a representation of p as a term with 1 and X as the inputs and where addition and multiplication are the operations. For each structure $\mathbf{B}$ over vocabulary τ, we define $p(\mathbf{B})$ as the τ-structure obtained by evaluating the representation of p by interpreting 1 as $\mathbf{I}_\tau$, X as $\mathbf{B}$, addition as the disjoint union $\uplus$ of two structures, and multiplication as the product of two structures. We have the following commutativity property.

Lemma 9.7 *For each univariate polynomial p with positive integer coefficients and each finite structure $\mathbf{B}$, it holds that $|\phi(p(\mathbf{B}))| = p(|\phi(\mathbf{B})|)$.*

Proof. This can be proved by a straightforward induction on the structure of p, using the observations that $|\phi(\mathbf{I}_\tau)| = 1$, $|\phi(\mathbf{D} \uplus \mathbf{D}')| = |\phi(\mathbf{D})| + |\phi(\mathbf{D}')|$, and $|\phi(\mathbf{D} \times \mathbf{D}')| = |\phi(\mathbf{D})| \cdot |\phi(\mathbf{D}')|$. □

Lemma 9.8 *(orthogonality) Let $\phi_1(S_1), \ldots, \phi_n(S_n)$ be connected liberal pp-formulas that are pairwise not counting equivalent. Then for every $m \geqslant 2$, there exist structures $(\mathbf{B}_{(a_1,\ldots,a_n)} \mid (a_1, \ldots, a_n) \in [m]^n)$ and injective functions $f_1, \ldots, f_n : [m] \to \mathbb{N}$ such that for each $(a_1, \ldots, a_n) \in [m]^n$ and each $i \in [n]$, it holds that $|\phi_i(\mathbf{B}_{(a_1,\ldots,a_n)})| = f_i(a_i)$. Moreover, when $\mathbf{A}$ is any structure on which $|\phi_i(\mathbf{A})| > 0$ for each $i \in [n]$, all of the structures $\mathbf{B}_{(a_1,\ldots,a_n)}$ can be chosen to be of the form $\mathbf{A} \times \cdot$ (that is, the product of $\mathbf{A}$ with another structure).*

Proof. For each $\phi_i(S_i)$, we have $\phi_i(S_i) = \widehat{\phi}_i(S_i)$ since these formulas are connected and liberal; hence, by Lemma 6.6, they are pairwise not semi-counting equivalent. It follows from Lemma 6.9 that there exists a structure $\mathbf{C}'$ such that the values $|\phi_i(\mathbf{C}')|$ are pairwise different. By taking a sufficiently large power P of $\mathbf{C}'$, we may obtain that for the structure $\mathbf{C} = \mathbf{C}'^P \times \mathbf{A}$, the values $c_i = |\phi_i(\mathbf{C})|$ are pairwise different. For each $(a_1, \ldots, a_n) \in [m]^n$, define $p_{(a_1,\ldots,a_n)}$ to be a univariate polynomial over the rationals that evaluates to 0 at 0, and to a_i at c_i (for each $i \in [n]$). Define D to be the absolute value of the product of all denominators of coefficients in the defined polynomials. Set $p'_{(a_1,\ldots,a_n)} = D \cdot p_{(a_1,\ldots,a_n)}$; each such polynomial has integer coefficients. Next, set $p^-_{(a_1,\ldots,a_n)}$ to be the restriction of $p'_{(a_1,\ldots,a_n)}$ to summands with negative coefficients. Define $p''_{(a_1,\ldots,a_n)}$ to be $p'_{(a_1,\ldots,a_n)} + 2\sum_{(a_1,\ldots,a_n)\in[m]^n}(-p^-_{(a_1,\ldots,a_n)})$. Now, for each $(a_1, \ldots, a_n) \in [m]^n$, define the structure $\mathbf{B}_{(a_1,\ldots,a_n)}$ as $p''_{(a_1,\ldots,a_n)}(\mathbf{C})$; for each $i \in [n]$, we have $|\phi_i(\mathbf{B}_{(a_1,\ldots,a_n)})| = |\phi_i(p''_{(a_1,\ldots,a_n)}(\mathbf{C}))| = p''_{(a_1,\ldots,a_n)}(c_i)$; the second equality here holds by the previous lemma. From these equalities and the definitions of $p_{(a_1,\ldots,a_n)}$ and $p''_{(a_1,\ldots,a_n)}$, it is straightforward to verify that the defined structures have the desired property. Our claim concerning the structure $\mathbf{A}$ holds as $p''_{(a_1,\ldots,a_n)}(0) = 0$ holds (for each $(a_1, \ldots, a_n) \in [m]^n$), implying that the structures $\mathbf{B}_{(a_1,\ldots,a_n)}$ provided can be obtained in the form $\mathbf{C} \times \cdot$, which has the form $\mathbf{A} \times \cdot$. □

We now introduce a highly useful notion, that of *component polynomial*. Fix a set V of liberal variables. Denote by $\mathcal{E}$ the set of counting equivalence classes of liberal connected pp-formulas (with liberal variables from V). A *component polynomial* q is a multivariate polynomial with integer coefficients over variables $\{X_e \mid e \in \mathcal{E}\}$. For any finite structure $\mathbf{B}$, we define the value of q evaluated on $\mathbf{B}$, denoted by $q[\![\mathbf{B}]\!]$, as the integer value obtained by evaluating q when each X_e is given the value $|\phi_e(\mathbf{B})|$, for a formula $\phi_e \in e$. The following is our main theorem on component polynomials.

Theorem 9.9 *When q is a component polynomial q that is a non-zero polynomial, there exists a finite structure $\mathbf{B}$ such that $q[\![\mathbf{B}]\!] \neq 0$. Moreover, when $\phi_1(S_1), \ldots, \phi_n(S_n)$ are representatives of the equivalence classes $e_1, \ldots, e_n \in \mathcal{E}$ whose corresponding variables X_{e_i} appears in q, the structure $\mathbf{B}$ may be picked as a structures provided by Lemma 9.8.*

In order to establish this theorem, we will make use of the following known fact concerning multivariate polynomials.

Proposition 9.10 *Let $p(x_1, \ldots, x_n)$ be a multivariate polynomial in n variables over a field F. For each $i \in [n]$, let d_i denote the degree of p as a polynomial in x_i, and suppose that $T_i \subseteq F$ is a set of size $d_i + 1$ or greater. Then, if p is not the zero polynomial, there exists a point $(t_1, \ldots, t_n) \in T_1 \times \cdots \times T_n$ such that $p(t_1, \ldots, t_n) \neq 0$.*

Proof. (Theorem 9.9) Let $\phi_1(S_1), \ldots, \phi_n(S_n)$ be as described in the theorem statement. Let $m \geqslant 2$ be a value that exceeds the degree of each of the variables $X_{e_1}, \ldots, X_{e_n}$ in q, and apply Lemma 9.8 to obtain structures $(\mathbf{B}_{(a_1,\ldots,a_n)} \mid (a_1, \ldots, a_n) \in [m]^n)$ and the corresponding functions $f_1, \ldots, f_n : [m] \to \mathbb{N}$. Evaluating q on these structures amounts to evaluating q when the variables $(X_{e_1}, \ldots, X_{e_n})$ are given values in $f_1([m]) \times \cdots \times f_n([m])$. By Proposition 9.10, q must evaluate to a non-zero value on one of these structures. □

We now prove Theorem 9.6.

Proof. (Theorem 9.6) Denote ℓ by $\sum_{i=1}^{m} c_i|\phi_i(V_i)|$ and let $(\mathbf{A}_1, V_1), \ldots, (\mathbf{A}_m, V_m)$ be the pairs corresponding to the formulas $\phi_1(V_1), \ldots, \phi_m(V_m)$. By rearranging the indices, we may assume for the sake of notation that $\mathbf{A}_1, \ldots, \mathbf{A}_k$ are homomorphically equivalent structures (where $k \in [m]$) and that for no i with $k < i \leqslant m$ does $\mathbf{A}_i$ have a homomorphism to $\mathbf{A}_1$.

For any structure $\mathbf{B}$, it holds that one of the values $|\phi_1(\mathbf{B})|, \ldots, |\phi_k(\mathbf{B})|$ is non-zero if and only if all of them are. From this and the fact that $\phi_1(V_1), \ldots, \phi_k(V_k)$ are pairwise not counting equivalent, it follows that these formulas are pairwise not semi-counting equivalent. Thus by Theorem 6.6, it holds that $\widehat{\phi_1}(V_1), \ldots, \widehat{\phi_k}(V_k)$ are pairwise not counting equivalent. For each formula $\widehat{\phi_i}(V_i)$ (with $i \in [k]$), by considering its liberal connected components, we may define r_i to be a component polynomial which is a product of variables from $\{X_e \mid e \in \mathcal{E}\}$ such that $|\widehat{\phi_i}(\mathbf{B})| = r_i[\![\mathbf{B}]\!]$ for all finite structures $\mathbf{B}$. The products $r_1, \ldots, r_k$ are pairwise distinct, so $r = c_1 r_1 + \cdots + c_k r_k$ is a non-zero component polynomial. By applying Lemma 9.8 with $\mathbf{A} = \mathbf{A}_1$ and then invoking Theorem 9.9, we obtain a finite structure $\mathbf{D}$ of the form $\mathbf{A} \times \cdot$ such that $r[\![\mathbf{D}]\!] \neq 0$. Since no structure $\mathbf{A}_i$ with $k < i \leqslant m$ maps homomorphically to $\mathbf{A}$, we have $|\phi_{k+1}(\mathbf{D})| = \cdots = |\phi_m(\mathbf{D})| = 0$ and hence $\ell(\mathbf{D}) = r[\![\mathbf{D}]\!] \neq 0$. □

10. Proofs of the Theorems of Section 7

Lemma 10.1 *Let ϕ be a pp-formula and let ϕ' be a $\sharp\mathsf{PP}$-representation of ϕ. Then there is a basic $\sharp\mathsf{PP}$-formula ϕ'' that is also a $\sharp\mathsf{PP}$-representation of ϕ such that $\mathsf{width}(\phi'') \leqslant \mathsf{width}(\phi')$ and $\sharp\text{-}\mathsf{width}(\phi'') \leqslant \sharp\text{-}\mathsf{width}(\phi')$.*

Proof. With Lemma 9.1 we may assume that ϕ' is flat, so let $\phi' = \sum_{i=1}^{\ell} \psi'_i \times \phi'_i$ where ψ'_i is constant and ϕ'_i is basic. Then using Lemma 9.4 and Lemma 8.3 we get that for every structure $\mathbf{B}$, $[\mathbf{B}, \phi'] = \sum_{i=1}^{\ell} c_i |\phi_i(\mathbf{B})|$, where $c_i \in \mathbb{Z}$ and ϕ_i is a pp-formula. Now combine the summands of counting equivalent pp-formulas to get a linear combination with

$$[\mathbf{B}, \phi'] = \sum_{i=1}^{\ell'} c'_i |\phi''_i(\mathbf{B})|, \tag{2}$$

where the ϕ''_i are pairwise not counting equivalent pp-formulas and $c'_i \in \mathbb{Z}\backslash\{0\}$. Note that for all ϕ''_i we have $\mathsf{qaw}(\phi''_i) \leqslant \mathsf{width}(\phi')$.

Since ϕ' is a $\sharp\mathsf{PP}$-representation of ϕ, we have $[\mathbf{B}, \phi'] = |\phi(\mathbf{B})|$ for all structures $\mathbf{B}$. With Theorem 9.6 it follows that the linear combination in (2) consists only of one summand with coefficient 1. Let $|\psi(\mathbf{B})|$ be that summand. We have $|\psi(\mathbf{B})| = |\phi(\mathbf{B})|$ and $\mathsf{qaw}(\psi) \leqslant \mathsf{width}(\phi')$. Now we apply Lemma 8.3 b) on input ψ to construct ϕ'' with the desired properties. □

Proof.[of Theorem 7.2] We start with the second statement. Let first Φ satisfy the contraction condition. Then there is a constant k such that for all cores ϕ of pp-formulas in Φ we have $\mathsf{tw}(\mathsf{contract}(\phi)) \leqslant k$. But then Lemma 8.3 yields basic $\sharp\mathsf{PP}$-representations ϕ' with $\sharp\text{-}\mathsf{width}(\phi') \leqslant k + 1$.

Now assume there is a constant k such that every formula ϕ in Φ has a $\sharp\mathsf{PP}$-representation ϕ' such that $\sharp\text{-}\mathsf{width}(\phi') \leqslant k$. By Lemma 10.1 we may assume that ϕ' is basic. Then by Lemma 8.3 we find a pp-formula ϕ'' that is counting equivalent to ϕ such that $\mathsf{tw}(\mathsf{contract}(\phi'')) \leqslant k$. Using Theorem 6.3 it follows that ϕ is logically equivalent to a formula ψ with $\mathsf{tw}(\mathsf{contract}(\phi)) \leqslant k$. Consequently, Φ satisfies the contraction condition.

For the first statement, let first Φ satisfy the tractability condition. Then there is a constant k such that for all cores ϕ of pp-formulas in Φ we have $\mathsf{tw}(\phi) \leqslant k$ and $\mathsf{tw}(\mathsf{contract}(\phi)) \leqslant k$. It follows that $\mathsf{qaw}(\phi) \leqslant 2k$ by Lemma 8.1. Then Lemma 8.3 yields a basic $\sharp\mathsf{PP}$-representations ϕ' with $\mathsf{width}(\phi') \leqslant 2k$.

Now assume there is a constant k such that every formula ϕ in Φ has a $\sharp\mathsf{PP}$-representation ϕ' such that $\mathsf{width}(\phi') \leqslant k$. We may again assume that ϕ' is basic. Then Lemma 8.3 gives a pp-formula ϕ'' that is counting equivalent to ϕ such that $\mathsf{qaw}(\phi) \leqslant k$. Using Theorem 6.3 it follows that ϕ is logically equivalent to a formula ψ with $\mathsf{qaw}(\phi)) \leqslant k$. Now applying Lemma 8.1 shows that Φ satisfies the tractability condition. □

Proof.[of Theorem 7.3] By Lemma 10.1 we may assume that the desired $\sharp\mathsf{PP}$-representation is basic. But then the correspondence of Lemma 8.3 in combination with Lemma 8.2 directly yields the result. □

Proof.[of Theorem 7.4] Let first Φ^* satisfy the tractability condition. Then there is a constant k such that for all cores ϕ of pp-formulas in Φ we have $\mathsf{tw}(\phi) \leqslant k$ and $\mathsf{tw}(\mathsf{contract}(\phi)) \leqslant k$. With Lemma 8.1 it follows that $\mathsf{qaw}(\phi) \leqslant 2k$. Let ϕ^-_{af} be defined as in Section 6.4. Remember that there are coefficients $c_1, \ldots, c_\ell$ and a set $\psi_1, \ldots, \psi_t$ of sentences such that the following holds for every structure $\mathbf{B}$: If $\mathbf{B}$ satisfies any sentence ψ_i, then $|\phi(\mathbf{B})| = |B|^{|\text{lib}(\phi)|}$. Otherwise $|\phi(\mathbf{B})| = \sum_{i=1}^{\ell} |\phi_i(\mathbf{B})|$, where the ϕ_i are formulas from ϕ^-_{af}. Let $|\text{lib}(\phi)| = r$. Then

$$
\begin{aligned}
|\phi(\mathbf{B})| =& (\prod_{i=1}^{t}(1 - |\psi_i(\mathbf{B})|))(\sum_{i=1}^{\ell} |\phi_i(\mathbf{B})|) \\
&+ |B|^r(1 - \prod_{i=1}^{t}(1 - |\psi_i(\mathbf{B})))|.
\end{aligned}
$$

. Now let $\phi'_1, \ldots, \phi'_\ell$ be the formulas we get by applying Lemma 8.3 on $\phi_1, \ldots, \phi_\ell$. Then set

$$
\begin{aligned}
\psi =& P\text{free}(\phi)(E\text{free}(\phi)(1 - C(\bigwedge_{i=1}^{t} \psi_i, \varnothing)) \times \prod_{i=1}^{\ell} \phi'_i) \\
&+ PV_r C(\bigwedge_{i=1}^{t} \psi_t, \varnothing),
\end{aligned}
$$

. Clearly, ψ is a $\sharp\mathsf{PP}$-representation of ϕ. Moreover, the width of ψ is the maximal width of the ψ_i and $C(\psi_i, \varnothing)$. But with Lemma 8.3, this maximum is bounded by $2k$.

For the other direction, assume that there is a constant k such that each ep-formula in Φ has a $\sharp\mathsf{EP}$-representation ϕ' with $\mathsf{width}(\phi) \leqslant k$. Let ψ be a pp-formula from Φ^+. Now choose ϕ such that $\psi \in \phi^+$. With Lemma 9.1 we may assume that ϕ is flat, i.e., it has the form $\sum_{i=1}^{\ell} \psi_i \times \phi_i$, where the ψ_i are constant and the ϕ_i are basic. As in the proof of Lemma 10.1, this yields for every $\mathbf{B}$

$$
|\phi(\mathbf{B})| = \sum_{i=1}^{\ell} c'_i |\phi'_i(\mathbf{B})|, \tag{3}
$$

where the ϕ_i are pairwise not counting equivalent. Moreover, $\mathsf{qaw}(\phi_i) \leqslant k$.

Let again $|\text{lib}(\phi)| = r$ and let $\psi_1, \ldots, \psi_t$ be the sentences in ϕ^-. Then as before, for every structure $|\phi(\mathbf{B})| = \prod_{i=1}^{t}(1-|\psi_i(\mathbf{B})|)(\sum_{i=1}^{\ell} |\phi_i(\mathbf{B})|)+|B|^r(1-\prod_{i=1}^{t}(1-|\psi_i(\mathbf{B})|))$. Now multiplying the righthand side out, we get that $|\phi(\mathbf{B})|$ can be expressed as a weighted sum of terms of the form $|\phi_i(\mathbf{B})| \cdot \prod_{j \subseteq J} |\psi_j(\mathbf{B})|$ and $|B|^r \prod_{j \subseteq J} |\psi_j(\mathbf{B})|$. These terms are equivalent to $|(\phi_i \wedge \bigwedge_{j \subseteq J} \psi_j)(\mathbf{B})$ and $\psi^r(\mathbf{B}) \ \prod_{j \subseteq J} |\psi_j(\mathbf{B})|$. Now combine counting equivalent summands as before to get a linear combination

$$
|\phi(\mathbf{B})| = \sum_{i=1}^{\ell} c'_i |\phi'_i(\mathbf{B})|. \tag{4}
$$

We claim that the linear combination in 4 contains $c \cdot |\psi(\mathbf{B})|$ for some $c \neq 0$ as a summand. To see this, note first that $c \cdot \psi(\mathbf{B})$ appears in the original weighted sum. Moreover, ψ is not counting equivalent to any other summand ψ' in this sum. To see this, note first that by construction of ϕ^+ there is no homomorphism from any of the ψ_j to ψ. Moreover, the ϕ_i are pairwise not counting equivalent.

It follows that (3) and (4) give two linear combinations that are equal for every structure $\mathbf{B}$. Using Theorem 9.6 shows that the linear combination of (3) contains a summand $c \cdot \psi'$ that is counting equivalent to ψ. Moreover, $\mathsf{qaw}(\psi') \leqslant k$. It follows that ψ is logically equivalent to a formula with quantifier aware width at most k. □

Proof.[of Theorem 7.5] (sketch) By Lemma 9.1 we may assume that ψ is flat, i.e., it has the for $\phi = \sum_{i=1}^{\ell} \psi_i \times \phi_i$ where the ψ_i are constant and the ϕ_i are basic. Note that $\mathsf{width}(\phi) = \max_{i \in \ell}(\mathsf{width}(\phi_i))$.

Note that every flat $\sharp\mathsf{EP}$-representation of ϕ can be turned into a linear combination as in (2). Moreover, starting with any such representation yields the same linear combination up to counting equivalence of the summands by Theorem 9.6. Now turning this linear combination into a flat $\sharp\mathsf{PP}$-formula yields a $\sharp\mathsf{PP}$-representation and minimizing the width of the summands with Theorem 7.3 gives a representation of optimal width. □

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[Bod96] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[Che14a] Hubie Chen. On the complexity of existential positive queries. *ACM Trans. Comput. Log.*, 15(1), 2014.

[Che14b] Hubie Chen. The tractability frontier of graph-like first-order query sets. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, page 31, 2014.

[CM77] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceddings of STOC'77*, pages 77–90, 1977.

[CM14] Hubie Chen and Stefan Mengel. A trichotomy in the complexity of counting answers to conjunctive queries. *CoRR*, abs/1408.0890, 2014. conference version to appear in ICDT 2015.

[DJ04] Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theor. Comput. Sci.*, 329(1-3):315–323, 2004.

[DKV02] Victor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In *Constraint Programming '02*, LNCS, 2002.

[DM13] A. Durand and S. Mengel. Structural tractability of counting of solutions to conjunctive queries. In *Proceedings of the 16th International Conference on Database Theory (ICDT 2013)*, 2013.

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[FK10] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[Gro07] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1), 2007.

[GS] Gianluigi Greco and Francesco Scarcello. Counting solutions to conjunctive queries: structural and hybrid tractability. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*.

[PS11] Reinhard Pichler and Sebastian Skritek. Tractable counting of the answers to conjunctive queries. In *Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management*, 2011.

[PY99] C. Papadimitriou and M. Yannakakis. On the Complexity of Database Queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999.

[Var95] Moshe Y. Vardi. On the complexity of bounded-variable queries. In *PODS'95*, pages 266–276, 1995.

A. Basic definitions and facts on treewidth

We give some basic facts about treewidth, see e.g. [FG06] for more details.

A tree decomposition of a graph $G = (V, E)$ is a pair $(T, (B_t)_{t \in V(T)})$ where T is a tree and $(B_t)_{t \in V(T)}$ is a family of subsets of V such that:

- For every $v \in V$, the set $\{t \in V(T) \mid v \in B_t\}$ is non-empty and connected in T.
- For every edge $uv \in E$, there is a $t \in V(T)$ such that $u, v \in B_t$.

The width of a tree decomposition $(T, (B_t)_{t \in V(T)})$ is $\max\{|B_t| \mid t \in V(T)\} - 1$. The treewidth $\mathsf{tw}(G)$ of G is the minimum of the widths of the tree decompositions of G. Computing tree decompositions of minimal width is fixed-parameter tractable parameterized by the treewidth[Bod96]. We sometimes use the well-known fact that for every clique K in G there must in every tree decomposition be a bag B_t that contains K. We often implicitly consider the tree T of a tree decomposition as rooted, with all edges directed from the leaves to the root.

A tree decomposition is called nice, if every $t \in V(T)$ is of one of the following types:

- t has no children and $|B_t| = 1$.
- t has one child t' and $B_t = B_{t'} \cup \{v\}$ for a vertex $v \in V \backslash B_{t'}$.
- t has one child t' and $B_t = B_{t'} \backslash \{v\}$ for a vertex $v \in B_{t'}$.
- t has two children t_1, t_2 with $B_t = B_{t_1} = B_{t_2}$.

It is well-known that a tree decomposition of width k of G can be turned into a nice tree decomposition of width k in polynomial time.

B. Proof of Observation 6.4

Proof. Let $\mathbf{C}$ be the structure that interprets every relation symbol in R of ϕ by $R^{\mathbf{C}} := \{0,1\}^{\mathsf{arity}(R)}$. Then $|\phi(\mathbf{C})| = 2^{|\mathrm{lib}(\phi)|}$ and $|\phi'(\mathbf{C})| = 2^{|\mathrm{lib}(\phi')|}$ and the claim follows directly. □

C. Proof of Proposition 6.5

Proof. Observe first that because h and h' are both surjective, we have that $|\mathrm{lib}(\phi)| = |\mathrm{lib}(\phi')|$. Then $\underline{\mathrm{h}}$ is a homomorphism from $\mathsf{aug}(\phi)$ to $\mathsf{aug}(\phi'')$, because it respects the new relations R_a.

Because $\mathsf{aug}(\phi)$ and $\mathsf{aug}(\phi'')$ are finite, there is a $k \in \mathbb{N}$ such that $(\underline{\mathrm{h}}'' \circ \underline{\mathrm{h}})^k|_S = \mathrm{id}_S$. It follows that id_S can be extended to a homomorphism $\mathsf{aug}(\phi'') \to \mathsf{aug}(\phi)$. Thus $\mathsf{aug}(\phi)$ and $\mathsf{aug}(\phi'')$ are homomorphically equivalent and by Theorem 2.2 the formulas ϕ and ϕ'' are logically equivalent. Consequently, ϕ and ϕ' are renaming equivalent. □

D. Proof of Proposition 6.7

Proof. Let $\mathbf{B}$ be a structure. Let ψ be the conjunction of the components deleted from ϕ to obtain $\widehat{\phi}$. If ψ is false on $\mathbf{B}$, then obviously $\phi(\mathbf{B}) = \emptyset$. Otherwise, ψ is true on $\mathbf{B}$, and for any assignment $f : V \to B$, it holds that $\mathbf{B}, f \models \phi$ if and only if $\mathbf{B}, f \models \widehat{\phi}$. □

E. Proof of Lemma 6.9

We first prove the following lemma.

Lemma E.1 *Let $\phi_1(S_1)$ and $\phi_2(S_2)$ be two pp-formulas over a vocabulary τ that are not semi-counting equivalent. Then there is a structure $\mathbf{D}$ such that for every primitive positive formula ϕ over τ we have $|\phi(\mathbf{D})| > 0$ and $|\phi_1(\mathbf{D})| \neq |\phi_2(\mathbf{D})|$.*

Proof. Let $\mathbf{B}$ be any structure on which ϕ_1 and ϕ_2 have a non-zero but different number of solutions. Such a structure exists by definition of semi-counting equivalence. We claim that we can choose $\mathbf{D} = \mathbf{B} + k\mathbf{I}$ for some $k \in \mathbb{N}, k > 0$ where $\mathbf{B} + k\mathbf{I}$ is defined as in the proof of Theorem 6.6. By way of contradiction, assume that $|\phi_1(\mathbf{B} + k\mathbf{I})| = |\phi_2(\mathbf{B} + k\mathbf{I})|$ for all $k \in \mathbb{N}, k > 0$. Then with the same argument as in the proof of Theorem 6.6 we get the contradiction that $|\phi_1(\mathbf{B})| = |\phi_2(\mathbf{B})|$. □

Proof. (Lemma 6.9) We make an induction on n; $n = 2$ is Lemma E.1.

For $n > 2$, let $\mathbf{D}$ be the structure we get by induction for $\phi_1, \ldots, \phi_{n-1}$. Since $|\phi(\mathbf{D})| > 0$ for every ϕ, we may w.l.o.g. assume that the ϕ_i are all pairwise not semi-counting equivalent. Therefore, we may assume w.l.o.g. that $|\phi_1(\mathbf{D})| < |\phi_2(\mathbf{D})| < \ldots < |\phi_{n-1}(\mathbf{D})|$. If $|\phi_n(\mathbf{D})| \neq |\phi_i(\mathbf{D})|$ for every $i \in [n-1]$ or ϕ_n is semi-counting equivalent to ϕ_i for some $i \in [n-1]$, then we are done. So we assume that there is an index i such that $|\phi_n(\mathbf{D})| = |\phi_i(\mathbf{D})|$ but ϕ_n is not semi-counting equivalent to ϕ_i.

Let $\mathbf{D}'$ be the structure we get by applying Lemma E.1 on ϕ_n and ϕ_i.

Now choose k such that for every j with $1 < j \leqslant i$ we have

$$\frac{|\phi_{j-1}(\mathbf{D})|^k}{|\phi_j(\mathbf{D})|^k} < \frac{1}{|\mathrm{lib}(\phi_{j-1})|^{|D'|}}.$$

Then we have for every $\ell \geqslant k$ and $1 < j < i$

$$\begin{aligned} |\phi_{j-1}(\mathbf{D}^\ell \times \mathbf{D}')| &= |\phi_{j-1}(\mathbf{D}^\ell)| \cdot |\phi_{j-1}(\mathbf{D}')| \\ &\leqslant |\phi_{j-1}(\mathbf{D}^\ell)| \cdot |\mathrm{lib}(\phi_{j-1})|^{|D'|} \\ &< |\phi_j(\mathbf{D}^\ell)| \\ &\leqslant |\phi_j(\mathbf{D}^\ell)| \cdot |\phi_j(\mathbf{D}')| \\ &= |\phi_j(\mathbf{D}^\ell \times \mathbf{D}')|. \end{aligned}$$

Analogously, we get for every $\ell > k$ that

$$|\phi_{i-1}(\mathbf{D}^\ell \times \mathbf{D}')| < |\phi_n(\mathbf{D}^\ell \times \mathbf{D}')|.$$

Now choose k' such that for every j with $i \leqslant j < n$ we have

$$\frac{|\phi_{j+1}(\mathbf{D})|^{k'}}{|\phi_j(\mathbf{D})|^{k'}} > |\mathrm{lib}(\phi_j)|^{|D'|}.$$

Then we have for every $\ell > k'$ and every $i \leqslant j < n$

$$\begin{aligned}|\phi_j(\mathbf{D}^\ell \times \mathbf{D'})| &= |\phi_j(\mathbf{D}^\ell)| \cdot |\phi_j(\mathbf{D'})| \\ &\leqslant |\phi_j(\mathbf{D}^\ell)| \cdot |\mathrm{lib}(\phi_j)|^{|D'|} \\ &< |\phi_{j+1}(\mathbf{D}^\ell)| \\ &\leqslant |\phi_{j+1}(\mathbf{D}^\ell)| \cdot |\phi_j(\mathbf{D'})| \\ &= |\phi_{j+1}(\mathbf{D}^\ell \times \mathbf{D'})|.\end{aligned}$$

Similarly, we get for every $\ell > k$ that

$$|\phi_{i+1}(\mathbf{D}^\ell \times \mathbf{D'})| > |\phi_n(\mathbf{D}^\ell \times \mathbf{D'})|.$$

Now choosing $\ell = \max(k, k')$ and noting that

$$\begin{aligned}|\phi_i(\mathbf{D}^\ell \times \mathbf{D'})| &= |\phi_i(\mathbf{D}^\ell)| \cdot |\phi_i(\mathbf{D'})| \\ &\neq |\phi_n(\mathbf{D}^\ell)| \cdot |\phi_n(\mathbf{D'})| \\ &= |\phi_n(\mathbf{D}^\ell \times \mathbf{D'})|\end{aligned}$$

completes the proof with $\mathbf{C} = \mathbf{D}^\ell \times \mathbf{D'}$. □

F. Proof of Proposition 6.11

Proof. Since ϕ and ϕ' are semi-counting equivalent, we have by Theorem 6.6 and Theorem 6.3 that $\widehat{\phi}$ and $\widehat{\phi'}$ are renaming equivalent. By Theorem 2.2 it follows that after renaming some variables $\mathsf{aug}(\widehat{\phi})$ and $\mathsf{aug}(\widehat{\phi'})$ are homomorphically equivalent via hommomorphisms h and h'. To simplify the argument, assume w.l.o.g. that ϕ and ϕ' are such that $h = h' = \mathrm{id}$.

Now if there is a homomorphism $\bar{h} : \mathbf{A} \to \mathbf{A'}$, then we can extend h on the components of ϕ deleted in the construction of $\widehat{\phi}$ to get a homomorphism $\mathsf{aug}(\mathbf{A}, S) \to \mathsf{aug}(\mathbf{A'}, S')$. Moreover, if there is a homomorphism $\bar{h}' : \mathbf{A'} \to \mathbf{A}$, we get a homomorphism $\mathsf{aug}(\mathbf{A'}, S') \to \mathsf{aug}(\mathbf{A}, S)$. Since ϕ and ϕ' are not counting equivalent and thus $\mathsf{aug}(\mathbf{A}, S)$ and $\mathsf{aug}(\mathbf{A'}, S')$ are not homomorphically equivalent, it follows that there is no homomorphism $\mathbf{A} \to \mathbf{A'}$ or no homomorphism $\mathbf{A'} \to \mathbf{A}$. □

G. Proof of Lemma 6.12

Proposition G.1 *Let $\phi_1, \ldots, \phi_s$ be a set of semi-counting equivalent pp-formulas that are pairwise not counting equivalent. Then there is a structure $\mathbf{C}$ and $i \in [s]$ such that $\mathbf{C} \models \phi_i$ but $\mathbf{C} \not\models \phi_j$ for all $j \in [s] \setminus \{i\}$.*

Proof. Let $\mathbf{A}_1, \ldots, \mathbf{A}_n$ be the structures of $\phi_1, \ldots, \phi_n$. By Proposition 6.11 the structures $\mathbf{A}_i$ are pairwise not homomorphically equivalent. For $i, j \in [n]$, we write $\phi_i < \phi_j$ if there is a homomorphism from $\mathbf{A}_i$ to $\mathbf{A}_j$. It is easy to check that $<$ induces a partial order on the ϕ_i. Let ϕ_i be a minimal element of this partial order, then there is no homomorphism from any $\mathbf{A}_j$ to ϕ_i with $i \neq j$. Setting $\mathbf{C} = \mathbf{A}_i$ completes the proof. $\square$

Proof. (Lemma 6.12) The algorithm recursively computes the $|\phi_i(\mathbf{B})|$ one after the other. So let the parameter and the input be given as in the statement of the lemma. By Proposition G.1, there is an $i \in [n]$ and a structure $\mathbf{C}$ such that $\mathbf{C} \models \phi_i$ but $\mathbf{C} \not\models \phi_j$ for all $j \in [s] \backslash \{i\}$. W.l.o.g. assume $i = s$. Then $|\phi_i(\mathbf{B} \times \mathbf{C})| = 0$ for $i < s$. Consequently, we have that the oracle lets us compute $c_s \cdot |\phi_n(\mathbf{B} \times \mathbf{C})| = c_s \cdot |\phi_n(\mathbf{B})| \cdot |\phi_n(\mathbf{C})|$. Computing $|\phi_n(\mathbf{C})|$ by brute force then yields $|\phi_s(\mathbf{B})|$.

Now note that for every structure $\mathbf{B}'$ we can also compute $\sum_{i=1}^{s-1} c_i \cdot |\phi_i(\mathbf{B}')|$ by this approach with one subtraction. So we can apply the algorithm again for $\phi_1, \ldots, \phi_{s-1}$, answering oracle queries for $\sum_{i=1}^{s-1} c_i \cdot |\phi_i(\mathbf{B}')|$ with the help of the oracle for $\sum_{i=1}^{s} c_i \cdot |\phi_i(\mathbf{B}')|$. $\square$

H. Proof of Theorem 4.1

Proof. (Theorem 4.1) We first describe a counting slice reduction (U, r) from $\mathsf{count}[\Phi]$ to $\mathsf{count}[\Phi^+]$. Let (U', r') denote the counting slice reduction from $\mathsf{count}[\Phi_{\mathrm{af}}]$ to $\mathsf{count}[\Phi^*_{\mathrm{af}}]$ given by Theorem 6.13. Define U to be the set $\{(\phi, \phi^+) \mid \phi$ is a normalized ep-formula $\}$. When $(\phi, \phi^+) \in U$, we define $r(\phi(V), \phi^+, \mathbf{B})$ to be the result of the following algorithm, which is FPT with respect to (π_1, π_2). For each sentence disjunct θ of $\phi(V)$, the algorithm queries $\mathsf{count}(\theta, \mathbf{B})$; if for some such disjunct θ it holds that $\mathbf{B} \models \theta$, then the algorithm outputs $|V|^{|B|}$. Otherwise, for any assignment $f : V \to B$, it holds that $\mathbf{B}, f \models \phi$ if and only if $\mathbf{B}, f \models \phi_{\mathrm{af}}$. So, the algorithm returns $r'(\phi_{\mathrm{af}}, \phi^*_{\mathrm{af}}, \mathbf{B})$ by running the corresponding algorithm for r'. In this run, the algorithm for r' only makes queries of the form $(\psi, \mathbf{B})$ (with $\psi \in \phi^*_{\mathrm{af}}$); such queries where $\psi \in \phi^-_{\mathrm{af}}$ are resolved using the oracle in the definition of counting slice reduction, and queries where $\psi \in \phi^*_{\mathrm{af}} \backslash \phi^-_{\mathrm{af}}$ are answered with 0. Correctness is straightforward to verify.

We next describe a counting slice reduction (U, r) from $\mathsf{count}[\Phi^+]$ to $\mathsf{count}[\Phi]$. Let (U', r') denote the counting slice reduction from $\mathsf{count}[\Phi^*_{\mathrm{af}}]$ to $\mathsf{count}[\Phi_{\mathrm{af}}]$ given by Theorem 6.13. Define U to be the set $\{(\psi, \{\phi\}) \mid \psi \in \phi^+\}$. We need to define $r(\psi(V), \phi(V), \mathbf{B})$ when $(\psi, \phi) \in U$.

Let us describe first an algorithm for the mapping r in the case that $\psi \in \phi^-_{\mathrm{af}}$. Let $(\mathbf{C}_1, V), \ldots, (\mathbf{C}_m, V)$ denote the pp-formulas in ϕ^-_{af}, and let $\mathbf{C}$ denote the disjoint union of the structures $\mathbf{C}_i$. Observe that for any structure $\mathbf{D}$, it holds that $\mathbf{D} \times \mathbf{C}, f \models \phi$ if and only if $\mathbf{D} \times \mathbf{C}, f \models \phi_{\mathrm{af}}$, since no sentence disjunct of ϕ holds on $\mathbf{C}$ (due to the definitions of $\mathbf{C}$ and ϕ^-_{af}). Call the algorithm for r' to compute $r'(\psi, \{\phi_{\mathrm{af}}\}, \mathbf{B} \times \mathbf{C}) = |\psi(\mathbf{B} \times \mathbf{C})|$; note that the oracle queries made by this algorithm can be resolved by an oracle for $\mathsf{count}(\phi, \cdot)$, since all such oracle queries have the form $\mathsf{count}(\phi_{\mathrm{af}}, \cdot \times \mathbf{B})$. As $|\psi(\mathbf{B} \times \mathbf{C})| = |\psi(\mathbf{B})| \cdot |\psi(\mathbf{C})|$, by dividing this quantity by $|\psi(\mathbf{C})|$, one can determine

$|\psi(\mathbf{B})|$, which is the desired value. Note that by the definition of $\mathbf{C}$, it holds that $|\psi(\mathbf{C})|$ is non-zero.

In order to describe the behavior of the algorithm for r in the case that ψ is a sentence disjunct of ϕ, we establish the following claim. Let $(\mathbf{A}, V)$ be the structure view of ψ.

Claim: Let $i : V \to V$ be the identity map on V. For each disjunct θ of ϕ, it holds that $\mathbf{A}, i \models \theta(V)$ if and only if $\theta = \psi$.

The backwards direction is clear, so we prove the forwards direction. If a disjunct θ is a free pp-formula, then $\mathbf{A}, i \not\models \theta(V)$ since θ contains an atom using a variable $v \in V$, whereas no tuple of a relation of $\mathbf{A}$ contains any variable from V. If a disjunct θ is a pp-sentence $(\mathbf{A}', V)$ not equal to ψ, then by definition of *normalized* ep-formula, there is no homomorphism from $\mathbf{A}'$ to $\mathbf{A}$ and hence $\mathbf{A}, i \not\models \theta(V)$. This establishes the claim.

Now suppose that ψ is a sentence disjunct of ϕ. In this case, the algorithm for $r(\psi(V), \phi(V), \mathbf{B})$ behaves as follows. It queries $\mathsf{count}(\phi, \mathbf{A} \times \mathbf{B})$ to determine $|\phi(\mathbf{A} \times \mathbf{B})|$; it outputs $|B|^{|V|}$ if $|\phi(\mathbf{A} \times \mathbf{B})|$ is equal to $(|A| \cdot |B|)^{|V|}$ (the *maximum count* possible there), and outputs 0 otherwise. We prove that this is correct by showing that $|\phi(\mathbf{A} \times \mathbf{B})|$ is the maximum count if and only if $\mathbf{B} \models \psi$.

For the backwards direction, suppose that $\mathbf{B} \models \psi$, and denote ψ by $(\mathbf{A}, V)$. Then, there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, and hence there is a homomorphism from $\mathbf{A}$ to $\mathbf{A} \times \mathbf{B}$. It follows that for any assignment $f : V \to V$, one has $\mathbf{A} \times \mathbf{B}, f \models \psi(V)$. For the forwards direction, suppose that $|\phi(\mathbf{A} \times \mathbf{B})|$ is the maximum count. Let $i' : V \to A \times B$ be any map such that for each $v \in V$, the value $i'(v)$ has the form $(i(v), j(v))$ where $j : V \to B$ is a map. We have that $\mathbf{A} \times \mathbf{B}, i' \not\models \phi(V)$. It follows that there is a disjunct θ of ϕ such that $\mathbf{A} \times \mathbf{B}, i' \models \theta(V)$. It follows that $\mathbf{A}, i \models \theta(V)$ and $\mathbf{B}, j \models \theta(V)$. By the claim established above, we have that $\theta = \psi$. Then, it holds that $\mathbf{B}, j \models \psi$, and we are done. $\square$

I. Remark I.1

Remark I.1 *The quantifier aware width can be arbitrarily higher than their treewidth. To see this consider the formula $\phi = \exists z \bigwedge_{i \in [n]} E(x_i, z)$. The primal graph of ϕ is a star, so it has treewidth 1. We claim that the quantifier aware width of ϕ is $n + 1$. To see this, observe first that the free variables $x_1, \ldots, x_n$ must appear above $\mathsf{top}(z)$ in a bag of any quantifier aware tree decomposition. But since $x_i z$ is an edge in the primal graph for every i, the variable x_i must also appear in a common bag with z and consequently also in $\mathsf{top}(z)$. Thus $\mathsf{top}(z)$ must contain $n + 1$ variables.*

J. Proof of Lemma 8.1

Proof. For the first inequality, observe first that any quantifier aware tree decomposition of ϕ is a tree decomposition of ϕ, so $\mathsf{tw}(\phi) + 1 \leqslant \mathsf{qaw}(\phi)$ is obvious. Now let $(T, (B_t)_{t \in T})$ be a quantifier aware tree decomposition of ϕ. Introduce for every free-component of ϕ a new vertex v_C. Then substitute in every B_t every non-free variable x of ϕ by v_C where C is the free-component that contains x. Call the result B'_t. We claim that

$(T, (B'_t)_{t \in T})$ is a tree decomposition of $\mathsf{contract}(\phi)$. To see this, note that by the same argument as in Remark I.1 we have for every free-component C a bag that contains $V(C) \cap \text{free}(\phi) \cup \{x_C\}$. This proves $\mathsf{tw}(\mathsf{contract}(\phi)) + 1 \leqslant \mathsf{qaw}(\phi)$.

For the second inequality, first compute a tree decomposition $(T, (B_t)_{t \in T})$ of the contraction $\mathsf{contract}(\phi)$. Note that for every free-component C of ϕ there is a bag B_{t^*} that contains $V(C) \cap \text{free}(\phi)$, because this variable set forms a clique in $\mathsf{contract}(\phi)$. Now compute a tree decomposition $(T', (B'_t)_{t \in T'})$ for $G[V(C) \backslash \text{free}(\phi)]$, where G is the primal graph of ϕ. Then construct for every $t \in T'$ a new bag $B''_t := B'_t \cup (V(C) \cap \text{free}(\phi))$. Finally, connect T to T' by connecting an arbitrary vertex of T' to t^*. Doing this for every free-component yields a quantifier aware tree decomposition of ϕ. Moreover, the width of the decomposition is at most $\mathsf{tw}(\mathsf{contract}(\phi)) + \mathsf{tw}(\phi) + 1$ which completes the proof. □

K. Proof of Lemma 8.2

Proof. Let ϕ be a pp-formula with primal graph G and $S := \text{free}(\phi)$. For each free-component C of ϕ choose a vertex $x_C \in V(C) \backslash \text{free}(\phi)$ and connect it to all vertices $y \in V(C) \cap \text{free}(\phi)$. Moreover, connect the vertices in $V(C) \cap \text{free}(\phi)$ by a clique. Call the resulting graph G'. We will show that the minimum of $\mathsf{tw}(G') + 1$ over the choices of the x_C is $\mathsf{qaw}(\phi)$.

We first show that for every choice of the x_C we have $\mathsf{tw}(G') + 1 \geqslant \mathsf{qaw}(\phi)$. To see this, fix a tree decomposition $(T, (B_t)_{t \in T})$ of G'. Since $V(C) \backslash \text{free}(\phi)$ is connected in G', the bags containing $V(C) \backslash \text{free}(\phi)$ are contained in a subtree T' of T. Moreover, because $\{x_c\} \cup (V(C) \cap \text{free}(\phi))$ is a clique in G', we know that $\{x_c\} \cup (V(C) \cap \text{free}(\phi)) \subseteq B_{t^*}$ for some t^* in T'. Since none of the vertices in $V(C) \backslash \text{free}(\phi)$ have any neighbors outside of $V(C)$, we may assume that t^* is the root of T'. Then it is easy to see that $(T, (B_t)_{t \in T})$ can be turned into a quantifier aware tree decomposition: We only have to potentially add a new bag $B_{t^{**}} := V(C) \cap \text{free}(\phi)$ and a vertex t^{**} in the decomposition. Then connect t^{**} to t^* and its parent and delete the edge between t^* and its parent.

For the other direction, let $(T, (B_t)_{t \in T})$ be a quantifier aware tree decomposition of ϕ. We will show that it is also a tree decomposition of G' for a choice of the x_C. First note that by the same argument as before, the vertices of $V(C) \cap \text{free}(\phi)$ are contained in a subtree T' of T. Let x'_C be the only variable of $V(C) \backslash \text{free}(\phi)$ that is contained in B_r where B_r is the root of T'. Note that by the same argument as in Remark I.1, we know that B_r contains $V(C) \cap \text{free}(\phi)$. Thus B_r covers all edges introduced in the construction of G' when choosing $x_C = x'_C$. Thus $(T, (B_t)_{t \in T})$ is indeed a tree decomposition of G' for the right choice of the x_C.

Since computing tree decompositions is fixed parameter tractable parameterized by the treewidth (see e.g. [FG06]), the only problem left to solve is the right choice of the x_C. But since the quantified variables of the different free-components are independent, we can do this choice independently for every free-component C as follows: Construct G'' by choosing a vertex $x_C \in V(C) \cap \text{free}(\phi)$ and proceed as in the construction of G'. Now for all other free-components C' connect $V(C') \cap \text{free}(\phi)$ to a clique and delete all variables in $V(C')\backslash\text{free}(\phi)$. Clearly, trying all potential choices of x_C lets us optimize the choice for C. Doing this for all free-components gives the desired choice and thus the optimal quantifier aware tree decomposition. □

L. Proof of Lemma 8.3

Proof. a) Let ϕ' be a basic $\sharp$PP-sentence. We construct ϕ by deleting all C-, E- and P-quantifiers and substituting all $\times$ by $\wedge$. Obviously, the result is a pp-formula. By potentially renaming quantified variables, make sure that every variable in ϕ is either free or quantified exactly once. For every subformula ψ of ϕ we define $\text{lib}(\psi)$ to be the variables of ϕ that are not quantified in ψ. Note that for every subformula ψ' of ϕ', we have an associated subformula ψ of ϕ. We claim that for all $\sharp$-subformulas ψ' and every assignment h to $\text{lib}(\psi')$ that

$$[\mathbf{B}, \psi'](h) = |\{h' : \text{lib}(\phi') \to B \mid h' \text{ extends } h, (\mathbf{B}, h') \models \psi\}|. \tag{5}$$

We show (5) by induction on the structure of basic $\sharp$PP-formulas. If $\psi' = C\psi''$ for a pp-formula ψ'', then we actually have $\psi = \psi''$. Moreover, h assigns to values to all free variables of ψ, so both sides of (5) are 1 if and only if h satisfies ψ. If $\psi' = EV\psi''$ or $\psi' = PV\psi''$, we get (5) directly from the semantics of $\sharp$-formulas and induction. Finally, if $\psi' = \psi'_1 \times \psi'_2$, we have that $\text{free}(\phi_1) \cap \text{free}(\phi_2) \subseteq \text{free}(\phi)$ and thus (5) follows easily.

It remains to show the inequalities of the width measures. To this end, consider the syntax tree T of ϕ'. For every node t of T, define $B_t := \text{free}(\phi'_t)$ where ϕ'_t is the subformula of ϕ' that has t as its root. Note that $(T, (B_t)_{t\in T})$ satisfies the connectivity condition and is thus a tree decomposition of ϕ of width $\mathsf{width}(\phi') - 1$. Also, $(T, (B_t)_{t\in T})$ is quantifier aware because in ϕ' existential quantification is only allowed in the pp-part in which all free variables of ϕ are still free. This shows $\mathsf{width}(\phi') \geqslant \mathsf{qaw}(\phi)$. Now observe that by deleting all bags that contain quantified variables we end up with a tree decomposition for $\mathsf{contract}(\phi)$. This shows $\sharp\text{-}\mathsf{width}(\phi) \geqslant \mathsf{tw}(\mathsf{contract}(\phi)) + 1$.

b) Let now ϕ be a pp-formula and let $(T, (B_t)_{t\in T})$ be a nice quantifier aware tree decomposition of ϕ of width $k - 1$. For every free-component C of ϕ, the vertices $V(C)\backslash\text{free}(\phi)$ all lie in the bags of a subtree T_C of T. Moreover, we may w.l.o.g. assume that the bags in T_C do not contain any vertices not in $V(C)$. Finally, we have that the bag B_{r_C} where r_c is the root of T_C contains $V(C) \cap \text{free}(\phi)$, because $(T, (B_t)_{t\in T})$ is quantifier aware. The results of [DKV02] assure that there is a pp-formula ϕ_C of width k that is logically equivalent to the pp-formula that we get by restricting ϕ to the atoms that have all of their variables in $V(C)$.

We now construct for every $t \in T$ such that B_t does not contain any quantified variables of ϕ a basic $\sharp$PP-sentence ϕ'_t. So let t be a node of t with the desired properties. Let $\mathsf{atom}(t)$ be the atoms of ϕ containing only variables in B_t and set $\bar{\phi}_t := \prod_{\psi\in\mathsf{atom}(t)} C\psi$. If t has no children, set $\phi_t := \bar{\phi}_t$.

If t has a child t' such that $B_{t'}\backslash\text{free}(\phi) \neq \emptyset$, then t has only that one child because $(T, (B_t)_{t\in T})$ is nice. Let C be the unique free-component of the variable in $B_{t'}\backslash\text{free}(\phi)$. We set $\phi_t := \bar{\phi}_t \times C\phi_C$.

If t has a child t' such that $B_{t'} \backslash \text{free}(\phi) = \varnothing$ and a variable x is forgotten when going from t' to t, then set $\phi_t := Px\phi_{t'}$.

If t has a child t' and a variable x is introduced when going from t' to t, then set $\phi_t := \bar{\phi}_t \times Ex\phi_{t'}$.

If t has two children t_1 and t_2, then note that $B_{t_1} \backslash \text{free}(\phi) = B_{t_2} \backslash \text{free}(\phi) = \varnothing$. Moreover, $\text{free}(\phi_{t_1}) = \text{free}(\phi_{t_2})$. We define $\phi_t := \phi_{t_1} \times \phi_{t_2}$.

Set $\phi' := P\text{free}(\phi_r)\phi_r$ where r is the root of ϕ.

An easy induction along the construction of ϕ similar to that in a) shows that ϕ' does indeed compute the correct value for every structure $\mathbf{B}$. Moreover, the width of ϕ' is at most k which completes the proof.

If we do not have a bound on $\mathsf{qaw}(\phi)$ but only on $\mathsf{tw}(\mathsf{contract}(\phi))$, the same construction as above yields the the bound $\mathsf{tw}(\mathsf{contract}(\phi)) + 1 \geqslant \sharp\text{-}\mathsf{width}(\phi')$. The only difference is that we do not have to bound the width of the pp-formulas with [DKV02]. □