Approximately Counting Answers to Conjunctive Queries with Disequalities and Negations

JACOB FOCKE[*], CISPA Helmholtz Center for Information Security, Germany
LESLIE ANN GOLDBERG[*], Department of Computer Science, University of Oxford, U.K.
MARC ROTH[*], School of Electronic Engineering and Computer Science, Queen Mary University of London, U.K.
STANISLAV ŽIVNÝ[*], Department of Computer Science, University of Oxford, U.K.

We study the complexity of approximating the number of answers to a small query φ in a large database $\mathcal{D}$. We establish an exhaustive classification into tractable and intractable cases if φ is a conjunctive query possibly including disequalities and negations:

- If there is a constant bound on the arity of φ, and if the randomised Exponential Time Hypothesis (rETH) holds, then the problem has a fixed-parameter tractable approximation scheme (FPTRAS) if and only if the treewidth of φ is bounded.
- If the arity is unbounded and φ does not have negations, then the problem has an FPTRAS if and only if the adaptive width of φ (a width measure strictly more general than treewidth) is bounded; the lower bound relies on the rETH as well.

Additionally we show that our results cannot be strengthened to achieve a fully polynomial randomised approximation scheme (FPRAS): We observe that, unless NP = RP, there is no FPRAS even if the treewidth (and the adaptive width) is 1.

However, if there are neither disequalities nor negations, we prove the existence of an FPRAS for queries of bounded fractional hypertreewidth, strictly generalising the recently established FPRAS for conjunctive queries with bounded hypertreewidth due to Arenas, Croquevielle, Jayaram and Riveros (STOC 2021).

CCS Concepts: • **Theory of computation → Design and analysis of algorithms**; • **Information systems → Relational database query languages**.

Additional Key Words and Phrases: approximate counting, conjunctive queries, fully polynomial randomised approximation scheme (FPRAS), fixed-parameter tractable randomised approximation scheme (FPTRAS)

ACM Reference Format:
Jacob Focke, Leslie Ann Goldberg, Marc Roth, and Stanislav Živný. 2022. Approximately Counting Answers to Conjunctive Queries with Disequalities and Negations. 1, 1 (August 2022), 28 pages. https://doi.org/XXXXXXX.XXXXXXX

[*]For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. All data is provided in full in the results section of this paper. The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The research was also supported by the European Research Council (ERC) consolidator grant No 725978 SYSTEMATICGRAPH. Stanislav Živný was supported by a Royal Society University Research Fellowship. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein. An extended abstract of this work is published in the proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems [23]

Authors' addresses: Jacob Focke, CISPA Helmholtz Center for Information Security, Saarbrücken, Germany; Leslie Ann Goldberg, Department of Computer Science, University of Oxford, Oxford, U.K.; Marc Roth, School of Electronic Engineering and Computer Science, Queen Mary University of London, Oxford, U.K.; Stanislav Živný, Department of Computer Science, University of Oxford, Oxford, U.K..

1 INTRODUCTION

The evaluation of conjunctive queries is amongst the most central and well-studied problems in database theory [1, 5, 9, 25]. These queries are also called *select-project-join queries* in relational algebra and *select-from-where* queries in SQL. In this work, we study the *counting* problem associated with conjunctive queries and with extensions to conjunctive queries allowing negations, disequalities, and unions of queries. Given a query φ and a database $\mathcal{D}$, an "answer" of φ in $\mathcal{D}$ is an assignment of values from the universe of $\mathcal{D}$ to the free variables of φ that can be extended (by also assigning values to the existential variables of φ) to an assignment satisfying φ. For example, the universe of the database $\mathcal{D}$ could be a set of people U, and $\mathcal{D}$ has an entry $F(a, b)$ whenever two people $a, b \in U$ are "friends". Then an answer to the query

$$\varphi(x) = \exists y \exists z\, F(x, y) \wedge F(x, z) \wedge (y \neq z) \tag{1}$$

is a person that has at least two friends (from the people in U).

The counting problem is to compute the number of answers of φ in $\mathcal{D}$. Our goal is to determine the parameterised complexity of this counting problem in the situation where the query φ is significantly smaller than the database $\mathcal{D}$; a formal exposition is given in Section 1.1.

Previous work [10, 16, 18] established that the problem of exactly counting answers to conjunctive queries is *extremely* difficult: Even very simple queries, such as acyclic conjunctive queries, which can be evaluated in polynomial time [24, 45], are sufficiently powerful to encode intractable problems in their counting versions, making any non-trivial improvement over the brute-force algorithm impossible under the Strong Exponential Time Hypothesis [16].

Therefore, the relaxation to approximate counting is necessary if efficient algorithms are sought. In this work, we quantify the complexity of approximating the number of answers to conjunctive queries with negations and disequalities, and unions thereof, in terms of several natural width measures of the queries, such as treewidth, fractional hypertreewidth, and adaptive width. This leads to a complete classification (and a new approximation algorithm) in the bounded-arity case, to a complete classification (and another new approximation algorithm) in the unbounded-arity case when negations are excluded, and to a new FPRAS in the unparameterised setting. The formal setup, including the definitions of the problems and the approximation schemes, are introduced in Section 1.1 and we present our results in Section 1.2.

1.1 Technical Background

A *signature* σ consists of a finite set of relation symbols with specified positive arities. A *(relational) database* $\mathcal{D}$ with signature $\mathrm{sig}(\mathcal{D})$ consists of a finite *universe* $U(\mathcal{D})$[1] together with, for each relation symbol $R \in \mathrm{sig}(\mathcal{D})$, a relation $R^{\mathcal{D}}$ over the universe $U(\mathcal{D})$ with the same arity that $\mathrm{sig}(\mathcal{D})$ specifies for R. The tuples in the relations $R^{\mathcal{D}}$ are called the *facts* of $\mathcal{D}$. A *conjunctive query* (CQ) φ with signature $\mathrm{sig}(\varphi)$ is a formula of the form

$$\varphi(x_1, \ldots, x_\ell) = \exists x_{\ell+1} \cdots \exists x_{\ell+k} \psi(x_1, \ldots, x_{\ell+k}),$$

where $\mathrm{vars}(\varphi)$ denotes the set of variables $\{x_1, \ldots, x_{k+\ell}\}$ of φ, $\mathrm{free}(\varphi)$ denotes the set of free (output) variables $\{x_1, \ldots, x_\ell\}$ of φ, and ψ is a conjunction of a finite number of atoms, which are predicates of the form $R(y_1, \ldots, y_j)$, where R is an arity-j relation symbol in $\mathrm{sig}(\varphi)$ and each y_i is a variable in $\mathrm{vars}(\varphi)$. Each symbol of $\mathrm{sig}(\varphi)$ appears in at least one predicate. Also, each variable in $\mathrm{vars}(\varphi)$ appears in at least one atom.

[1]$U(\mathcal{D})$ is also often referred to as the "domain" of $\mathcal{D}$.

In an *extended* conjunctive query (ECQ) there are three more types of allowable atoms.

- Equality: $y_i = y_j$.
- Disequality: $y_i \neq y_j$.
- Negated predicate: $\neg R(y_1, \dots, y_j)$, where R is an arity-j relation symbol in $\mathrm{sig}(\varphi)$ and each y_i is a variable in $\mathrm{vars}(\varphi)$.

Again, each element of $\mathrm{sig}(\varphi)$ appears at least once in φ (as a predicate, as a negated predicate, or both).

In fact, without loss of generality we can assume that φ has no equalities, since we can re-write φ to avoid these by replacing equal variables with a single variable. Thus, from now on, we assume that ECQs do not have equalities.

It is natural to extend conjunctive queries by adding disequalities and negations. Such extended queries were studied for instance in [4, 11, 28, 32, 33, 39]. Sometimes we will be interested in extending CQs by adding disequalities but not negations. We refer to these partially-extended queries as DCQs.

Consider an ECQ φ. The following notation of "Solution" captures the assignments (of elements in $U(\mathcal{D})$ to the variables of φ) that satisfy φ. It does not distinguish between existential and free variables, but we do that later in Definition 2.

Definition 1. (solution, $\mathsf{Sol}(\varphi, \mathcal{D})$) Let φ be an ECQ and let $\mathcal{D}$ be a database with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$. A *solution* of $(\varphi, \mathcal{D})$ is an assignment $\alpha\colon \mathrm{vars}(\varphi) \to U(\mathcal{D})$ which has the following property.

- For every predicate $R(y_1, \dots, y_j)$ of φ, the tuple $(\alpha(y_1), \dots, \alpha(y_j))$ is in $R^{\mathcal{D}}$,
- For every negated predicate $\neg R(y_1, \dots, y_j)$ of φ, the tuple $(\alpha(y_1), \dots, \alpha(y_j))$ is not in $R^{\mathcal{D}}$, and
- For every disequality $y_i \neq y_j$ of φ we have $\alpha(y_i) \neq \alpha(y_j)$.

We use $\mathsf{Sol}(\varphi, \mathcal{D})$ to denote the set of solutions of $(\varphi, \mathcal{D})$.

In this work, we will not be interested so much in the solutions of $(\varphi, \mathcal{D})$ but rather in their projections onto the free (output) variables of φ.

Definition 2. (proj, answer, $\mathsf{Ans}(\varphi, \mathcal{D})$) Let φ be an ECQ and let $\mathcal{D}$ be a database with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$. Let $\alpha\colon \mathrm{vars}(\varphi) \to U(\mathcal{D})$ be an assignment of elements in $U(\mathcal{D})$ to the variables of φ. We use $\mathrm{proj}(\alpha, \mathrm{free}(\varphi))$ to denote α's projection onto the free variables of φ. That is, $\mathrm{proj}(\alpha, \mathrm{free}(\varphi))$ is the assignment from $\mathrm{free}(\varphi)$ to $U(\mathcal{D})$ that agrees with α. An *answer* of $(\varphi, \mathcal{D})$ is an assignment $\tau\colon \mathrm{free}(\varphi) \to U(\mathcal{D})$ of elements in $U(\mathcal{D})$ to the free variables of φ which can be extended to a solution in the sense that there is a solution α of $(\varphi, \mathcal{D})$ with $\mathrm{proj}(\alpha, \mathrm{free}(\varphi)) = \tau$. We write $\mathsf{Ans}(\varphi, \mathcal{D})$ for the set of all answers of $(\varphi, \mathcal{D})$.

Our main focus is on the problem of approximately counting answers to extended conjunctive queries φ, parameterised[2] by the size of φ, which is denoted by $\|\varphi\|$, and is defined to be the sum of $|\mathrm{vars}(\varphi)|$ and the sum of the arities of the atoms in φ.

The formal problem definition is as follows. Let Φ be a class of ECQs.
Name: #ECQ (Φ)
Input: An ECQ $\varphi \in \Phi$ and a database $\mathcal{D}$ with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$.
Parameter: $\|\varphi\|$.
Output: $|\mathsf{Ans}(\varphi, \mathcal{D})|$.

[2]This choice of the parameter allows one to produce fine-grained complexity results which are appropriate for instances in which the query size is (significantly) smaller than the size of the database (see for example the section "Why Fixed-Parameter Tractability" in [37, Section 1] for a longer discussion of this point). The notion of fixed-parameter tractability is made formal below.

We define the problems #CQ (Φ) and #DCQ (Φ) analogously, by requiring the input to be a CQ (in the case of #CQ (Φ)) or a DCQ (in the case of #DCQ (Φ)).

The size of the encoding of the input pair $(\varphi, \mathcal{D})$ is taken to be the sum of $\|\varphi\|$ and the size of the encoding of $\mathcal{D}$ (written $\|\mathcal{D}\|$) which is defined to be $|\mathrm{sig}(\mathcal{D})| + |U(\mathcal{D})|$ plus the sum of the lengths of the tuples in the relations of $\mathcal{D}$.

Note that singleton unary relations in $\mathcal{D}$ can be used to implement "constants" in φ. To see this, for any $v \in U(\mathcal{D})$ let $R_v^{\mathcal{D}}$ denote the relation $\{v\}$. It is possible to refer to the constant v in φ by including R_v in $\mathrm{sig}(\varphi)$ and constraining some variable $x \in \mathrm{vars}(\varphi)$ with the predicate $R_v(x)$. Of course the size $\|\varphi\|$ increases by an additional constant amount by the introduction of the variable x and the predicate $R_v(x)$. Adding all singleton unary relations to the signature of $\mathcal{D}$ does not increase $\|\mathcal{D}\|$ significantly, since $|U(\mathcal{D})|$ is already included in $\|\mathcal{D}\|$.

While we focus in this work on counting, there is also a corresponding decision problem ECQ(Ψ) with the same input and parameter as #ECQ (Ψ). The output of ECQ(Ψ) is a single bit, indicating whether $|\mathrm{Ans}(\varphi, \mathcal{D})| > 0$. The complexity of ECQ$(\Psi)$ is not fully resolved and some special cases, such as parameterised subgraph isomorphism are thought to be difficult to resolve [17, Chapter 33.1].

We next design the notion of efficient approximation for counting problems such as #ECQ (Ψ).

Randomised Approximation Schemes and Fixed-Parameter Tractability. Given a value V and $\varepsilon, \delta \in (0,1)$, an (ε, δ)-*approximation* of V is a random variable X that satisfies $\Pr(|X - V| \le \varepsilon V) \ge 1 - \delta$.

Let $\#A$ be a counting problem that, when given input x, asks for the value $V(x)$. Slightly overloading notation, an (ε, δ)-*approximation* for $\#A$ is a randomised algorithm that, given an input x to $\#A$, outputs an (ε, δ)-approximation of $V(x)$. A *fully polynomial randomised approximation scheme* (FPRAS) for $\#A$ is a randomised algorithm that, on input x, ε, δ, computes an (ε, δ)-approximation of $V(x)$ in time polynomial in $\|x\|$, $1/\varepsilon$, and $\log(1/\delta)$.

Suppose that the counting problem $\#A$ is parameterised by a parameter k (as the problem #ECQ (Φ) is parametetrised by $\|\varphi\|$). A *fixed-parameter tractable randomised approximation scheme* (FPTRAS) for $\#A$ is a randomised algorithm that, on input x, ε, δ, computes an (ε, δ)-approximation of $V(x)$ in time $f(k) \cdot \mathrm{poly}(\|x\|, 1/\varepsilon, \log(1/\delta))$, for some function $f \colon \mathbb{R} \to \mathbb{R}$.

Applying this definition, note that an FPTRAS for #ECQ (Φ) has a running time bound of $f(\|\varphi\|) \cdot p(\|\mathcal{D}\|, 1/\varepsilon, \log(1/\delta))$. In other words, relative to the definition of FPRAS, the definition of FPTRAS relaxes the condition that the algorithm must run in polynomial time by allowing a super-polynomial factor in the size of the query. Since the query is assumed to be significantly smaller than the database, this is a very natural notion of an efficient algorithm. Indeed, we will show that all FPTRASes for #ECQ (Φ) constructed in this work cannot be upgraded to FPRASes (subject to natural complexity hypotheses). The reason that they cannot be upgraded is that, even for very restricted query classes Φ, there are reductions from NP-hard problems to the problem of producing an FPRAS for #ECQ (Φ).

Considering #ECQ (Φ) as a parameterised problem allows one to interpolate between the classical complexity of the problem, in which no assumptions are made regarding the size of the input query, and its *data complexity*, in which the input query is fixed.

From the viewpoint of data complexity, there is a brute-force polynomial-time algorithm for counting answers to a query, by iterating through all assignments of the variables, roughly in time $\|\mathcal{D}\|^{O(\|\varphi\|)}$. If the query φ is fixed, the running time of this brute-force algorithm is bounded by a polynomial in the input size. In the fixed-parameter setting the goal is instead to separate the (potentially exponential) running time in the query size from the (polynomial) running time, in the size of the database.

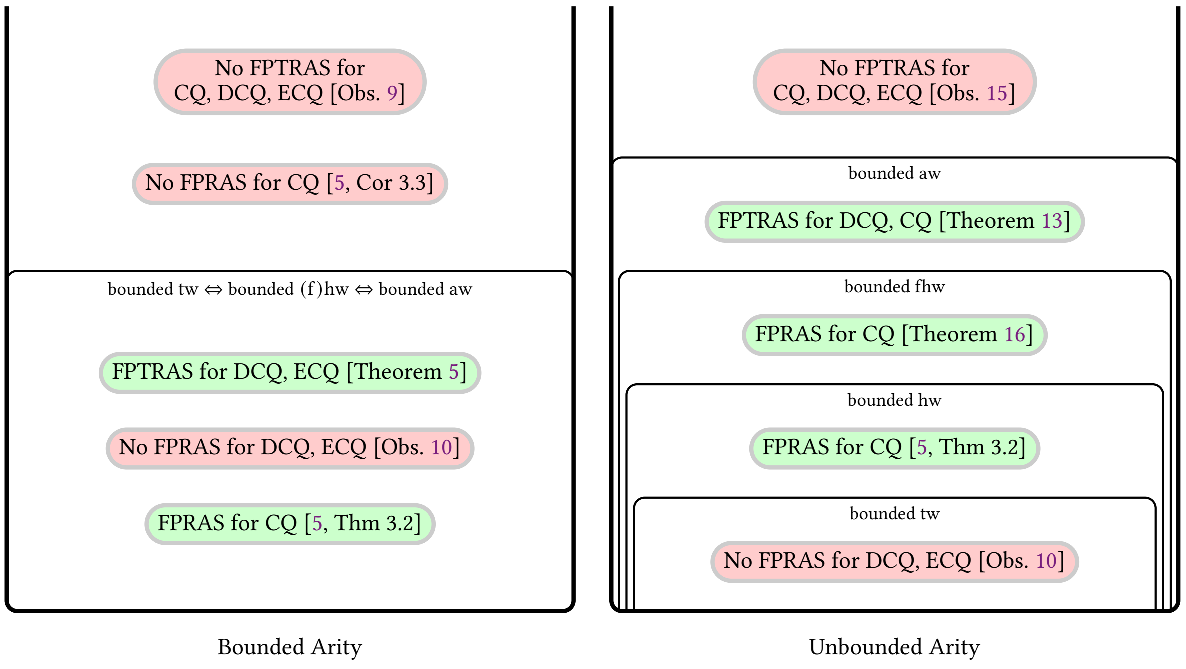
No FPTRAS for CQ, DCQ, ECQ [Obs. 9]
No FPRAS for CQ [5, Cor 3.3]
bounded tw ⇔ bounded (f)hw ⇔ bounded aw
FPTRAS for DCQ, ECQ [Theorem 5]
No FPRAS for DCQ, ECQ [Obs. 10]
FPRAS for CQ [5, Thm 3.2]
Bounded Arity
No FPTRAS for CQ, DCQ, ECQ [Obs. 15]
bounded aw
FPTRAS for DCQ, CQ [Theorem 13]
bounded fhw
FPRAS for CQ [Theorem 16]
bounded hw
FPRAS for CQ [5, Thm 3.2]
bounded tw
No FPRAS for DCQ, ECQ [Obs. 10]
Unbounded Arity

Fig. 1. Overview of our results on approximately counting answers to conjunctive queries (CQs), to conjunctive queries with disequalities (DCQs), and to conjunctive queries with disequalities and negations (ECQs). Upper and lower bounds depend on a variety of width measures of the input queries, namely, treewidth (tw), hypertreewidth (hw), fractional hypertreewidth (fhw), and adaptive width (aw). The equivalence of the width parameters in the case of bounded arity is well known; we provide an explicit argument in Observation 34. For completeness, we also compare our results to recent work of Arenas et al. [5]. The lower bounds either rely on the assumption that NP ≠ RP or on the rETH. All referenced theorems and observations are stated in Section 1.2. Note that, while our results complete the picture of the complexity of CQ, DCQ, and ECQ in the bounded arity case, two questions remain open for the unbounded arity case: Assuming the adaptive width is bounded, does ECQ have an FPTRAS, and does CQ have an FPRAS?

1.2 Our Results

In order to give an overview of our results, we provide an illustration in Figure 1.

Bounded-Treewidth ECQs. The tractability criteria in our results will depend on the hypergraph associated with a conjunctive query (Definition 3, see also [25]). A *hypergraph* H consists of a (finite) set of vertices $V(H)$ and a set $E(H) \subseteq 2^{V(H)}$ of non-empty hyperedges. The *arity* of a hypergraph is the maximum size of its hyperedges.

Definition 3. $(H(\varphi), \Phi_C)$ Given an ECQ φ, the *hypergraph of* φ, denoted $H(\varphi)$, has vertex set $V(H(\varphi)) = \text{vars}(\varphi)$. For each predicate of φ there is a hyperedge in $E(H(\varphi))$ containing the variables appearing in the predicate. For each negated predicate of φ, there is a hyperedge in $E(H(\varphi))$ containing the variables appearing in the negated predicate. For any class of hypergraphs C, Φ_C denotes the class of all ECQs φ with $H(\varphi) \in C$.

We emphasise that $H(\varphi)$ does **not** contain any hyperedges corresponding to the disequalities of φ; note that this makes positive results in terms of $H(\varphi)$ stronger, but it also makes these results harder to prove.[3]

Our first result uses the treewidth of a hypergraph (Definition 4, originally from [41]). The definitions of other width measures that are used throughout this work, such as fractional hypertreewidth and adaptive width, are deferred to the sections in which they are used.

Definition 4. (tree decomposition, treewidth) A *tree decomposition* of a hypergraph H is a pair $(T, \mathbf{B})$ where T is a (rooted) tree and $\mathbf{B}$ assigns a subset $B_t \subseteq V(H)$ (called a *bag*) to each $t \in V(T)$. The following two conditions are satisfied: (i) for each $e \in E(H)$ there is a vertex $t \in V(T)$ such that $e \subseteq B_t$, and (ii) for each $v \in V(H)$ the set $\{t \in V(T) \mid v \in B_t\}$ induces a (connected) subtree of T. The treewidth $\mathrm{tw}(T, \mathbf{B})$ of the tree decomposition $(T, \mathbf{B})$ is $\max_{t \in V(T)} |B_t| - 1$. The *treewidth* $\mathrm{tw}(H)$ of H is the minimum of $\mathrm{tw}(T, \mathbf{B})$, minimised over all tree decompositions $(T, \mathbf{B})$ of H.

Our first theorem is as follows.

Theorem 5. *Let t and a be positive integers. Let C be a class of hypergraphs such that every member of C has treewidth at most t and arity at most a. Then #ECQ (Φ_C) has an FPTRAS, running in time* $\exp(\mathrm{O}(||\varphi||^2)) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||)$.

Technical Challenges. While, in the case of bounded arity, negated predicates can be simulated by adding a negated relation (symbol) $\overline{R}$ for each relation (symbol) R, the disequalities have to be treated separately since we do not include them in the query hypergraph (see the discussion below Definition 3).

However, the main difficulty stems from the fact that our queries have both quantified and free variables; recall e.g. the query in (1). We note that the restricted case in which there are no quantified variables can be dealt with in a much easier way by using standard and well-established reductions from approximate counting to decision. For example, in the special case of arity 2, and with all disequalities present, approximate counting of answers to queries without quantified variables can be encoded as approximate counting subgraphs, which can be done efficiently for bounded treewidth graphs using colour-coding [3, 6].

If quantified variables are allowed, the situation changes drastically: While it does not matter for the decision version which variables are quantified and which are not, even quantifying a single variable can transform the counting version from easy to infeasible.[4] The core technical difficulty arising in both the recent result of Arenas et al. [5] and also in our work is the problem of handling quantified variables in the context of approximate counting. While Arenas et al. were able to establish an FPRAS for the case of conjunctive queries of bounded (hyper)treewidth, we show that an FPRAS is not possible if disequalities are allowed (see Observation 10). For this reason, we relax the condition of feasibility of approximation by aiming for an FP**T**RAS.

[3]If we included hyperedges corresponding to disequalities, we could just model these disequalities as (binary) relations and reduce to the case of conjunctive queries without disequalities. However, adding those hyperedges can increase the treewidth of the hypergraph (see Definition 4) significantly, so it would weaken the results significantly.

[4]Consider for example the following query in the signature of graphs

$$\varphi(x_1, \dots, x_k) = \exists y \bigwedge_{i=1}^{k} E(y, x_i)\,.$$

Deciding whether φ has an answer is computationally trivial, since it is equivalent to deciding whether there are k (not necessarily distinct) vertices that have a common neighbour. In other words, we can always return "Yes" if the graph has at least one edge. However, (exactly) counting answers to φ cannot be done in time $O(|V(G)|^{k-\varepsilon})$ unless the Strong Exponential Time Hypothesis fails [16], ruling out any improvement over the brute-force algorithm for the counting version. In the case of approximate counting, the result of Arenas et al. [5] yields an FPRAS for counting answers to φ. Also, our result, Theorem 5, yields an FPTRAS even if we additionally add the constraint that the x_is should be pairwise different. Note that our result relaxes the notion of feasibility from an FPRAS to an FPTRAS since it turns out that the former is not always possible if disequalities are allowed (see Observation 10). Finally, we point out that even exact counting becomes easy if we make y a free variable, that is, if we modify the query as follows: $\varphi'(x_1, \dots, x_k, y) = \bigwedge_{i=1}^{k} E(y, x_i)$. The reason for this easiness is that counting answers to φ' in G is equivalent to counting homomorphisms from $H(\varphi')$ to G, which can be done efficiently as $H(\varphi')$ has treewidth 1 [13].

In this work, we solve the problem of handling quantified variables by relying on the recent k-Hypergraph framework of Dell, Lapinskas and Meeks [15], which, in combination with colour-coding, will ultimately establish Theorem 5. We note that the presentation of our methods could be streamlined if we would only consider the bounded arity case. However, since understanding the unbounded arity case is also part of our goal, we present the framework in slightly more generality than we need for the bounded arity case (see Lemma 22).

Before we continue with the presentation of further results, we will present a useful application of Theorem 5.

Application to Locally Injective Homomorphisms. Given graphs G and G', a homomorphism from G to G' is a mapping from the vertices of G to the vertices of G' that maps the edges of G to edges of G'. A homomorphism h is *locally injective* if for each vertex v of G, the restriction of h to the neighbourhood $N_G(v)$ of v in G is injective. Locally injective homomorphisms have been studied extensively, see [22] for an overview, and they can be applied, for instance, to model interference-free assignments of frequencies (of networks such as wireless networks) [21]. Some recent works on locally injective homomorphisms include [43] and [7]. The complexity of exactly counting locally injective homomorphisms from a fixed graph G to an input graph G' has been considered and is fully classified [42].

Given G and G', it is easy to construct an ECQ $\varphi(G)$ and a database $\mathcal{D}(G')$ such that locally injective homomorphisms from G to G' are in one-to-one correspondence with $\mathsf{Ans}(\varphi(G), \mathcal{D}(G'))$. For this, the signature of $\varphi(G)$ and $\mathcal{D}(G')$ has one binary relation E (representing the edge set of a graph). $\mathcal{D}(G')$ is the structure representing G' — its universe $U(\mathcal{D})$ is $V(G')$ and its relation $E^{\mathcal{D}(G')}$ is the edge set of G'. The query $\varphi(G)$ is constructed as follows. For convenience, let $k = |V(G)|$ and assume that $V(G) = [k]$. Let $\mathrm{cn}(G)$ be the set of pairs $i \neq j$ of vertices of G such that i and j have a common neighbour. Then $\varphi(G)$ is the following query φ (which has no existential variables).

$$\varphi(x_1, \dots, x_k) = \bigwedge_{\{i,j\} \in E(G)} E(x_i, x_j) \ \wedge \bigwedge_{(i,j) \in \mathrm{cn}(G)} x_i \neq x_j.$$

Consequently, Theorem 5 also gives an FPTRAS for counting locally injective homomorphisms from graphs G with bounded treewidth. We define the problem of counting locally injective homomorphisms as follows: Let C and C' be two classes of graphs.

Name: #LIHom(C, C')

Input: Graphs $G \in C$ and $G' \in C'$.

Parameter: $|V(G)|$.

Output: The number of locally injective homomorphisms from G to G'.

Corollary 6. *Let t be a positive integer. Let C_t be the class of all graphs with treewidth at most t, and let C be the class of all graphs. Then #LIHom(C_t, C) has an FPTRAS.*

Matching Lower Bound. Theorem 5 is optimal under standard assumptions — subject to the randomised Exponential Time Hypothesis (rETH) there is a matching lower bound showing that there is no FPTRAS for #ECQ (Φ_C) if the treewidth of C is unbounded. In fact, there is no FPTRAS for #CQ (Φ_C) in this case. In order to explain the lower bound, we first state the rETH.

Conjecture 7 (rETH, [29])**.** *There is a positive constant c such that no algorithm, deterministic or randomised, can decide the satisfiability of an n-variable* 3*-SAT instance in time* $\exp(c \cdot n)$ *(with failure probability at most* $1/4$*).*

The lower bound relies on a result by Marx [35, Theorem 1.3]. We state it here using our notation — in addition we allow randomised algorithms and therefore replace ETH by rETH.[5]

Theorem 8 ([35, Theorem 1.3])**.** *Let a be positive integer. Let C be a recursively enumerable class of hypergraphs such that every member of C has arity at most a. Suppose that the treewidth of hypergraphs in C is unbounded. If there is a computable function f and a randomised algorithm that, given a CQ $\varphi \in \Phi_C$ and a database $\mathcal{D}$ with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$, decides in time $f(H(\varphi)) \cdot (\|\varphi\| + \|\mathcal{D}\|)^{o(\mathrm{tw}(H(\varphi)/\log \mathrm{tw}(H(\varphi))))}$ whether $(\varphi, \mathcal{D})$ has an answer, then rETH fails.*

In order to apply Theorem 8, note that an FPTRAS for #CQ (Φ_C) provides a $(1/2, 1/4)$-approximation for #CQ (Φ_C) that runs in time $f(\|\varphi\|) \cdot \mathrm{poly}(\|\varphi\| + \|\mathcal{D}\|)$. It follows from the definition of $\|\varphi\|$ that $\|\varphi\|$ is a function of $H(\varphi)$. Thus, the FPTRAS provides a $(1/2, 1/4)$-approximation for #CQ (Φ_C) that runs in time $f(H(\varphi)) \cdot \mathrm{poly}(\|\varphi\| + \|\mathcal{D}\|)$. This approximation algorithm can be used to solve the decision problem of determining whether the output of #CQ (Φ_C) is nonzero. Thus, by Theorem 8, we obtain the following observation.

Observation 9. *Let a be positive integer. Let C be a recursively enumerable class of hypergraphs such that every member of C has arity at most a. If the treewidth of hypergraphs in C is unbounded then #CQ (Φ_C) does not have an FPTRAS, unless rETH fails.*

Observation 9 shows that Theorem 5 provides a tight result for hypergraph classes with bounded arity in the sense that, assuming rETH, there is an FPTRAS if and only if the treewidth is bounded. Note that the stated lower bound is slightly stronger than required since it also applies to conjunctive queries without extensions.

Theorem 5 shows that if C is any class of hypergraphs such that every member of C has treewidth at most t and arity at most a then there is an (ε, δ)-approximation algorithm for #ECQ (Φ_C) that runs in time $f(\|\varphi\|) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||)$, where f is exponential in $\|\varphi\|^2$. A natural question is whether the function f can be improved. Specifically, a polynomial f would imply the existence of an FPRAS for #ECQ (Φ_C). However, the following observation, proved by reduction from the Hamilton path problem, shows that there is no such FPRAS unless NP = RP, even when $t = 1$ and $a = 2$.

Observation 10. *Let C be the class of all hypergraphs with treewidth at most 1 and arity at most 2. Then there is no FPRAS for #DCQ (Φ_C), unless NP = RP.*

Proof. Given an n-vertex graph G, we will show how to construct (in time $\mathrm{poly}(n)$) an instance $(\varphi, \mathcal{D})$ of #DCQ (Φ_C) such that the answers in $\mathsf{Ans}(\varphi, \mathcal{D})$ are in one-to-one correspondence with the Hamiltonian paths of G. This implies (e.g., [20, Theorem 1]) that there is no FPRAS for #DCQ (Φ_C) unless NP = RP.

The construction is as follows. $U(\mathcal{D}) = V(G)$. The signature $\mathrm{sig}(\varphi) = \mathrm{sig}(\mathcal{D})$ contains a single binary relation symbol E. The relation $E^{\mathcal{D}}$ is the edge set $E(G)$. The query φ is defined as follows.

$$\varphi(x_1, \dots, x_n) = \bigwedge_{i \in [n-1]} E(x_i, x_{i+1}) \ \wedge \bigwedge_{1 \le i < j \le n} x_i \neq x_j.$$

Note that φ has no existential variables so the solutions of $(\varphi, \mathcal{D})$ are in one-to-one correspondence with $\mathsf{Ans}(\varphi, \mathcal{D})$. It is clear from the definition of φ that these are also in one-to-one correspondence with the Hamilton paths of G.

It remains to show that $\varphi \in \Phi_C$, that is that $H(\varphi)$ has treewidth 1 and arity 2. Both of these follow from the fact that $H(\varphi)$ is the path $x_1, \dots, x_n$, which follows from the definition of $H(\varphi)$ (Definition 3). □

[5]In case it is not clear to the reader how Theorem 8 matches [35, Theorem 1.3], we refer to the introduction of [37] where this result is stated as Theorem 1.2.

Interestingly, the situation changes if we consider CQs without extensions. Arenas, Croquevielle, Jayaram, and Riveros [5] give an FPRAS for $\#\mathrm{CQ}\ (\Phi_C)$ for any bounded-treewidth class C of hypergraphs. Their result will be stated formally as Theorem 38 in Section 5.2 (where it will be improved). Observation 10 ensures that their result cannot be generalised to cover extended conjunctive queries (unless NP = RP). However, their result does apply to classes of hypergraphs with unbounded arity.

Beyond Bounded Arity. Even though Observation 9 gives a tight lower bound for classes C of hypergraphs with bounded arity, there is room for improvement if the arity in C is unbounded. In this setting, it is worth considering other notions of hypergraph width, such as hypertreewidth, fractional hypertreewidth, adaptive width and submodular width. These width measures will be defined in the sections where they are used. Here we just give the relationships between them, from [37, Figure 2].

Definition 11 (weakly dominated, strongly dominated, weakly equivalent)**.** A width measure is a function from hypergraphs to $\mathbb{R}_{\geq 0}$. Given two width measures w_1 and w_2, we say that w_1 is *weakly dominated* by w_2 if there is a function f such that every hypergraph H has $w_2(H) \leq f(w_1(H))$. We say that w_1 is *strongly dominated* by w_2 if w_1 is weakly dominated by w_2 and there is a class of hypergraphs that has unbounded w_1-width, but bounded w_2-width. We say that w_1 and w_2 are *weakly equivalent* if they weakly dominate each other.

If w_1 is strongly dominated by w_2 then the class of hypergraphs with bounded w_1-width is strictly contained in the class of hypergraphs with bounded w_2-width. Thus, algorithmic results based on bounded w_2-width have strictly greater applicability than algorithmic results based on bounded w_1-width. If w_1 and w_2 are weakly equivalent then algorithmic results for bounded w_1-width and bounded w_2-width are equivalent.

Lemma 12 ([37])**.** *Treewidth is strongly dominated by hypertreewidth. Hypertreewidth is strongly dominated by fractional hypertreewidth. Fractional hypertreewidth is strongly dominated by adaptive width, which is weakly equivalent to submodular width.*

Note that all of the width measures from Lemma 12 are weakly equivalent if we assume an overall bound on the arity of hypergraphs; we make this formal in Observation 34.

When restricting the queries to DCQs instead of ECQs, we can improve Theorem 5 by allowing unbounded arity, extending it to adaptive width. The following Theorem is proved in Section 5.1.

Theorem 13. *Let b be a positive integer. Let C be a class of hypergraphs such that every member of C has adaptive width at most b. Then $\#\mathrm{DCQ}\ (\Phi_C)$ has an FPTRAS.*

Further Technical Challenges. In addition to the challenges that arose in the bounded arity case, a further issue that arises in the unbounded arity case is finding the right notion of treewidth on hypergraphs; recall that the notions of treewidth, hypertreewidth, fractional hypertreewidth and adaptive width are all equivalent in the bounded arity setting, but not in the unbounded arity setting.

We ultimately ended up with adaptive width as the most general width measure for which we can establish the existence of an FPTRAS. Despite the fact that approximate counting is often harder than decision [8, 20], we find that in the current setting, the criterion for tractability is the same for decision and approximate counting. In fact, it turns out that Theorem 13, i.e., the choice of adaptive width, is optimal, unless the rETH fails. This matching lower bound comes from another result of Marx [37, Theorem 7.1], which we express using our notation; for what follows, we write $\mathrm{aw}(H)$ for the adaptive width of a hypergraph H:

Theorem 14 ([37, Theorem 7.1])**.** *Let C be a recursively enumerable class of hypergraphs with unbounded adaptive width. If there is a computable function f and a randomised algorithm that, given a CQ $\varphi \in \Phi_C$ and a database $\mathcal{D}$ with* $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$*, decides in time $f(H(\varphi)) \cdot (\|\varphi\| + \|\mathcal{D}\|)^{o(\mathrm{aw}(H(\varphi))^{1/4})}$ whether $(\varphi, \mathcal{D})$ has an answer, then rETH fails.*

As we noted earlier, an FPTRAS for #CQ (Φ_C) provides a $(1/2, 1/4)$-approximation for #CQ (Φ_C) that runs in time $f(\|\varphi\|) \cdot \mathrm{poly}(\|\varphi\| + \|\mathcal{D}\|)$. It follows from the definition of $\|\varphi\|$ that $\|\varphi\|$ is a function of $H(\varphi)$. Thus, the FPTRAS provides a $(1/2, 1/4)$-approximation for #CQ (Φ_C) that runs in time $f(H(\varphi)) \cdot \mathrm{poly}(\|\varphi\| + \|\mathcal{D}\|)$. This approximation algorithm can be used to solve the decision problem of determining whether the output of #CQ (Φ_C) is nonzero. Thus, by Theorem 14, we obtain the following observation.

Observation 15. *Let C be a recursively enumerable class of hypergraphs with unbounded adaptive width. Then* #CQ (Φ_C) *does not have an FPTRAS, unless rETH fails.*

Clearly, Observation 15 also rules out an FPTRAS for #DCQ (Φ_C) (matching Theorem 13) or an FPTRAS for #ECQ (Φ_C) when C has unbounded adaptive width.

As stated in Observation 10, there is little hope of improving Theorem 13 to obtain an FPRAS instead of an FPTRAS. However, as mentioned previously, if disequalities are not part of the query, the result by Arenas et al. [5] also applies to classes of hypergraphs with unbounded arity — they give an FPRAS for #CQ (Φ_C) if C is a class of hypergraphs with bounded hypertreewidth. We improve this result by showing that it suffices to require bounded fractional hypertreewidth — the following theorem is proved in Section 5.2.

Theorem 16. *Let b be a positive integer. Let C be a class of hypergraphs such that every member of C has fractional hypertreewidth at most b. Then* #CQ (Φ_C) *has an FPRAS.*

We don't know whether Theorem 16 can be extended to instances with bounded adaptive width (closing the gap between Theorem 16 and the lower bound presented in Observation 15). However, this situation now mirrors the situation in the decision world: Bounded fractional hypertreewidth is the most general property of the class of underlying hypergraphs, for which the conjunctive query decision problem is known to be polynomial-time solvable, see [34]. Fixed-parameter tractability for this problem is also known for classes with bounded adaptive width [37], complemented by the matching lower bound stated in Theorem 14. The question whether bounded fractional hypertreewidth is the right answer for the existence of a polynomial-time algorithm remains open and we refer to the conclusion of [37] for further discussion regarding this question. With our results from Theorems 13 and 16 we now arrive at precisely the same gap for the existence of an FPRAS for the corresponding counting problem #CQ.

Extensions: Sampling and Unions. Using standard methods, the algorithmic results presented in this work can be extended in two ways. First, approximate counting algorithms can be lifted to obtain algorithms for approximately uniformly sampling answers. This is based on the fact that the problems that we considered are self-reducible (self-partitionable) [19, 30, 44]. Second, instead of single (extended) conjunctive queries, one can also consider unions thereof. We refer to Section 6 for further details.

1.3 Related Work

The first systematic study of the complexity of *exactly* counting answers to conjunctive queries is due to Pichler and Skritek [40], and Durand and Mengel [18]. Their result has been extended to unions of conjunctive queries by Chen and

Mengel [10] and to extended[6] conjunctive queries by Dell, Roth and Wellnitz [16]. Unfortunately, all of those results have shown that exact counting is infeasible even for very restricted classes of queries, indicating that relaxing to approximate counting is necessary if efficient algorithms are sought for wide classes of queries. As highlighted before, a recent result by Arenas et al. [5] shows that there is an FPRAS for approximately counting answers to conjunctive queries whenever the hypergraphs of the queries have bounded hypertreewidth, yielding a significantly wider class of tractable instances than for exact counting. We state and further discuss their result in Section 5.2.

If we restrict to instances without existential quantifiers then counting answers to conjunctive queries is equivalent to the problem of counting homomorphisms from a small relational structure to a large one, the complexity of which was investigated by Dalmau and Jonsson [13] in the case of exact counting and by Bulatov and Živný [8] in the case of approximate counting.

The notion of an FPTRAS was introduced by Arvind and Raman [6] and has since been established as the standard notion for tractability of parameterised approximate counting problems (see [38] for an overview).

1.4 Algorithmic Methods and Proof Techniques

The FPTRAS presented in our main result, Theorem 5, relies on a framework that was introduced in a recent work by Dell, Lapinskas, and Meeks [15]. Their framework establishes an algorithm for approximating the number of hyperedges in a hypergraph that uses an oracle for a related decision problem. Our contribution is to figure out how to reduce the problem of approximating answers to the problem of approximately counting hyperedges in an appropriately-constructed hypergraph, thus giving an algorithmic result that completely matches the corresponding hardness result (Observation 9). Our algorithm for the unbounded-arity case (Theorem 13) has the same general structure and here an additional contribution is determining the correct criterion for the DCQ case (which turns out to be bounded adaptive width).

The FPRAS that we present in Theorem 16 first constructs a tree decomposition in a convenient format, then collects appropriate local information in the tree decomposition. Using this local information, it reduces the problem of approximately counting answers to the problem of approximately counting outputs accepted by a tree automaton (which can be accomplished by an algorithm of Arenas, Croquevielle, Jayaram, and Riveros [5]). A key observation leading to the improved result is that the tree automaton reduction still works even when the tree decomposition may have more than polynomially many hyperedges per bag, as long as the number of relevant partial solutions is bounded by a polynomial, as is the case for bounded fractional hypertreewidth [27].

2 TECHNICAL PRELIMINARIES

2.1 Using Decision Oracles to Approximately Count Hyperedges

We start by introducing the terminology that we need. A hypergraph H is called *ℓ-uniform* if each hyperedge of H has cardinality ℓ. An *ℓ-partite subset* of a (finite) set V is a tuple $(V_1, \dots, V_\ell)$ of (pairwise) disjoint subsets of V. This is called an *ℓ-partition* of V if $V = \cup_{i=1}^{\ell} V_i$.

An ℓ-uniform hypergraph $H = (V, E)$ is *ℓ-partite* with ℓ-partition $(V_1, \dots, V_\ell)$ if $(V_1, \dots, V_\ell)$ is an ℓ-partition of V and every hyperedge in E contains exactly one vertex in each V_i.

[6]"Disequalities" are sometimes referred to as "Inequalities", and "Negations" are sometimes referred to as "Non-monotone Constraints".

Given an ℓ-uniform hypergraph $H = (V, E)$ and an ℓ-partite subset $(V_1, \dots, V_\ell)$ of V, the hypergraph $H[V_1, \dots, V_\ell]$ has vertex set $\bigcup_{i=1}^{\ell} V_i$; recall that the V_i are pairwise disjoint. The hyperedge set of $H[V_1, \dots, V_\ell]$ is the set of all hyperedges in E that contain (exactly) one vertex in each V_i. Note that $H[V_1, \dots, V_\ell]$ is ℓ-partite.

We write EdgeFree(H) for the predicate that is satisfied if a hypergraph H has no edges. The main result of Dell, Lapinskas and Meeks is as follows.

Theorem 17 ([15])**.** *There is an algorithm $\mathbb{A}(\varepsilon, \delta, H)$ with the following behaviour. Suppose that $0 < \varepsilon, \delta < 1$ are positive reals, H is an ℓ-uniform hypergraph, and that (in addition to learning $V(H)$ and ℓ) the algorithm $\mathbb{A}$ has access to an oracle for evaluating the predicate* EdgeFree$(H[V_1, \dots, V_\ell])$ *for any ℓ-partite subset $(V_1, \dots, V_\ell)$ of $V(H)$.*

$\mathbb{A}$ computes an (ε, δ)-approximation of $|E(H)|$ in time $O(NT)$*, using at most T calls to the oracle, where $N = |V(H)|$ and $T = \Theta(\log(1/\delta)\varepsilon^{-2}\ell^{6\ell}(\log N)^{4\ell+7})$.*

We emphasise that, while learning $V(H)$, ℓ, ε, and δ, the algorithm in Theorem 17 does not learn $E(H)$ as part of the input. The only access that the algorithm has to $E(H)$ is via the oracle which evaluates the predicate EdgeFree$(H[V_1, \dots, V_\ell])$. In fact, in our application H will be the hypergraph whose hyperedges are the elements of $\mathsf{Ans}(\varphi, \mathcal{D})$ for an ECQ φ with ℓ free variables and a database $\mathcal{D}$ with $|U(\mathcal{D})| = N$. Theorem 17 will help us to reduce the problem of approximating $|\mathsf{Ans}(\varphi, \mathcal{D})|$ to the (decision) problem of determining whether one exists via the oracle for the predicate EdgeFree.

2.2 From Conjunctive Queries to Relational Structures and Homomorphisms

It is well-known that answers to conjunctive queries are closely related to homomorphisms between relational structures. In this work, it will be convenient to view answers in the language of homomorphisms. We start with the relevant definitions.

Recall that a signature σ consists of a finite set of relation symbols with specified positive arities. We use ar(R) to denote the arity of a relation symbol R and ar(σ) to denote the maximum arity of any relation symbol in σ. Given a signature σ, a *structure* $\mathcal{A}$ with signature sig$(\mathcal{A}) = \sigma$ consists of a finite universe $U(\mathcal{A})$ and, for each relation symbol $R \in \sigma$, a relation $R^{\mathcal{A}} \subseteq U(\mathcal{A})^{\mathrm{ar}(R)}$. Following [26], we use $\|\mathcal{A}\|$ to denote the size of structure $\mathcal{A}$, which is given by $\|\mathcal{A}\| = |\mathrm{sig}(\mathcal{A})| + |U(\mathcal{A})| + \sum_{R \in \mathrm{sig}(\mathcal{A})} |R^{\mathcal{A}}| \cdot \mathrm{ar}(R)$. Note that a (relational) database is a structure.

Given two structures $\mathcal{A}$ and $\mathcal{B}$ with sig$(\mathcal{A}) \subseteq$ sig$(\mathcal{B})$, a *homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a function $h\colon U(\mathcal{A}) \to U(\mathcal{B})$ such that for all $R \in$ sig$(\mathcal{A})$ with $t = \mathrm{ar}(R)$ and all tuples $(a_1, \dots, a_t) \in R^{\mathcal{A}}$ it holds that $(h(a_1), \dots, h(a_t)) \in R^{\mathcal{B}}$. Then Hom$(\mathcal{A} \to \mathcal{B})$ denotes the set of homomorphisms from $\mathcal{A}$ to $\mathcal{B}$.

Let φ be an ECQ and let $\mathcal{D}$ be a database with sig$(\varphi) \subseteq$ sig$(\mathcal{D})$. We define a pair of associated structures $\mathcal{A}(\varphi)$ and $\mathcal{B}(\varphi, \mathcal{D})$ with the goal of expressing query answers in $\mathsf{Ans}(\varphi, \mathcal{D})$ as homomorphisms from $\mathcal{A}$ to $\mathcal{B}$.

Definition 18 ($\mathcal{A}(\varphi)$)**.** The universe of $\mathcal{A}(\varphi)$ is $U(\mathcal{A}(\varphi)) = \mathrm{vars}(\varphi)$. The signature of $\mathcal{A}(\varphi)$ is constructed from sig(φ) as follows

- If there is a predicate in φ involving the relation symbol R, then R is in the signature of $\mathcal{A}(\varphi)$ (with the same arity as R has in sig(φ)).
- If there is a negated predicate in φ involving the relation symbol R then the relation symbol $\overline{R}$ is in the signature of $\mathcal{A}(\varphi)$ (with the same arity as R has in sig(φ)).

Finally, for each $R \in$ sig(φ) and $j = \mathrm{ar}(R)$, $R^{\mathcal{A}(\varphi)}$ is the set of tuples $(y_1, \dots, y_j)$ for which $R(y_1, \dots, y_j)$ is a predicate in φ; and $\overline{R}^{\mathcal{A}(\varphi)}$ is the set of tuples $(y_1, \dots, y_j)$ for which $\neg R(y_1, \dots, y_j)$ is a negated predicate in φ.

For $R \in \mathrm{sig}(\varphi)$, let $P^+_\varphi(R)$ be the set of predicates of φ that use R, and let $P^-_\varphi(R)$ be the set of negated predicates of φ that use R. Then

$$|U(\mathcal{A}(\varphi))| + \sum_{R \in \mathrm{sig}(\mathcal{A}(\varphi))} |R^{\mathcal{A}(\varphi)}| \cdot \mathrm{ar}(R) = |\mathrm{var}(\varphi)| + \sum_{R \in \mathrm{sig}(\varphi)} \left(|P^+_\varphi(R)| + |P^-_\varphi(R)|\right) \cdot \mathrm{ar}(R) \leq \|\varphi\|.$$

Recall that $\|\varphi\|$ is the sum of $|\mathrm{vars}\,(\varphi)\,|$ and the sum of the arities of the atoms in φ. Also, the size of a structure $\mathcal{A}$ is $\|\mathcal{A}\| = |\mathrm{sig}(\mathcal{A})| + |U(\mathcal{A})| + \sum_{R \in \mathrm{sig}(\mathcal{A})} |R^{\mathcal{A}}| \cdot \mathrm{ar}(R)$. Thus, we obtain the following observation regarding the size of $\mathcal{A}(\varphi)$.

Observation 19. *Let φ be an ECQ with v negated predicates. Then $\|\mathcal{A}(\varphi)\| \leq |\mathrm{sig}(\varphi)| + v + \|\varphi\| \leq 3\|\varphi\|$.*

Definition 20 ($\mathcal{B}(\varphi, \mathcal{D})$)**.** The universe of $\mathcal{B}(\varphi, \mathcal{D})$ is the universe $U(\mathcal{D})$. The signature is $\mathrm{sig}(\mathcal{B}(\varphi, \mathcal{D})) = \mathrm{sig}(\mathcal{A}(\varphi))$ and the relations are defined as follows.

- For each $R \in \mathrm{sig}(\mathcal{A}(\varphi)) \cap \mathrm{sig}(\mathcal{D})$, $R^{\mathcal{B}(\varphi,\mathcal{D})} = R^{\mathcal{D}}$.
- For each $\overline{R} \in \mathrm{sig}(\mathcal{A}(\varphi)) \setminus \mathrm{sig}(\mathcal{D})$, $\overline{R}^{\mathcal{B}(\varphi,\mathcal{D})} = U(\mathcal{D})^{\mathrm{ar}(R)} \setminus R^{\mathcal{D}}$.

Note that $\|\mathcal{B}(\varphi, \mathcal{D})\|$ is bounded from above by

$$|\mathrm{sig}(\mathcal{A}(\varphi))| + |U(\mathcal{D})| + \sum_{R \in \mathrm{sig}(\mathcal{A}(\varphi)) \cap \mathrm{sig}(\mathcal{D})} |R^{\mathcal{D}}| \cdot \mathrm{ar}(R) + \sum_{\overline{R} \in \mathrm{sig}(\mathcal{A}(\varphi)) \setminus \mathrm{sig}(\mathcal{D})} |U(\mathcal{D})|^{\mathrm{ar}(R)} \cdot \mathrm{ar}(R),$$

where, for an ECQ φ with v negated predicates $|\mathrm{sig}(\mathcal{A}(\varphi))| \leq |\mathrm{sig}(\varphi)| + v \leq |\mathrm{sig}(\mathcal{D})| + v$. Thus, we obtain the following observation regarding the size of $\mathcal{B}(\varphi, \mathcal{D})$.

Observation 21. *Let φ be an ECQ with $a = \mathrm{ar}(\mathrm{sig}(\varphi))$. If φ has v negated predicates then $\|\mathcal{B}(\varphi, \mathcal{D})\| \leq \|\mathcal{D}\| + v + v\,a\,|U(\mathcal{D})|^a \leq 2\|\varphi\|(\|\mathcal{D}\| + v|U(\mathcal{D})|^a)$.*

Let φ be an ECQ and let $\mathcal{D}$ be a database with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$. Let $\Delta(\varphi) = \{\{x_i, x_j\} \mid x_i \neq x_j \text{ is an atom of } \varphi\}$. Note that $\mathsf{Sol}(\varphi, \mathcal{D}) = \{h \in \mathrm{Hom}(\mathcal{A}(\varphi) \to \mathcal{B}(\varphi, \mathcal{D})) : \forall \{x_i, x_j\} \in \Delta(\varphi) : h(x_i) \neq h(x_j)\}$. Therefore,

$$\begin{aligned} \mathsf{Ans}(\varphi, \mathcal{D}) = \{\tau\colon \mathrm{free}\,(\varphi) \to U(\mathcal{D}) \mid \exists h \in \mathrm{Hom}(\mathcal{A}(\varphi) \to \mathcal{B}(\varphi, \mathcal{D})) : \mathrm{proj}(h, \mathrm{free}\,(\varphi)) = \tau \\ \wedge\ \forall \{x_i, x_j\} \in \Delta(\varphi) : h(x_i) \neq h(x_j)\}. \end{aligned} \tag{2}$$

3 USING DECISION ORACLES TO COUNT ANSWERS

We start by defining the homomorphism decision problem.
Name: Hom
Input: Structures $\mathcal{A}$ and $\mathcal{B}$ with $\mathrm{sig}(\mathcal{A}) \subseteq \mathrm{sig}(\mathcal{B})$.
Output: Is there a homomorphism from $\mathcal{A}$ to $\mathcal{B}$?

The goal of this section is to establish Lemma 22, the proof of which will be obtained by a combination of the k-hypergraph framework (Theorem 17) and colour-coding.

Lemma 22. *There is a randomised algorithm that is equipped with oracle access to Hom and takes the following inputs*

- *an ECQ φ,*
- *a database $\mathcal{D}$ with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$,*
- *rational numbers ε and δ in $(0, 1)$.*

Let $a = \mathrm{ar}(\mathrm{sig}(\varphi))$ and let ν be the number of negated predicates in φ. The algorithm computes an (ε, δ)-approximation of $|\mathsf{Ans}(\varphi, \mathcal{D})|$ in time

$$\exp(\mathrm{O}(\|\varphi\|^2)) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, \|\mathcal{D}\|, \nu|U(D)|^a)\,.$$

Each oracle query $\textsc{Hom}(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$ *that is made by the algorithm has the property that* $\widehat{\mathcal{A}}$ *can be obtained from* $\mathcal{A}(\varphi)$ *by adding unary relations and satisfies* $\|\widehat{\mathcal{A}}\| \le 5\|\varphi\|^2$.

In order to prove Lemma 22 we require some prerequisites and definitions.

Definition 23. ($U_i(\mathcal{D})$) Given a database $\mathcal{D}$ and an integer i we define $U_i(\mathcal{D}) = U(\mathcal{D}) \times \{i\}$.

Intuitively, $U_i(\mathcal{D})$ will be used to specify the image of the ith variable of some query φ.

Definition 24 ($H(\varphi, \mathcal{D})$)**.** Let φ be an ECQ and let $\mathcal{D}$ be a database with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$. Let $\ell = |\mathrm{free}\,(\varphi)\,|$ and let $x_1, \dots, x_\ell$ be an enumeration of the variables in $\mathrm{free}\,(\varphi)$. We define an $(\ell|U(\mathcal{D})|)$-vertex ℓ-uniform hypergraph $H(\varphi, \mathcal{D})$ as follows.

- $V(H(\varphi, \mathcal{D})) = \bigcup_{i=1}^{\ell} U_i(\mathcal{D})$
- $E(H(\varphi, \mathcal{D})) = \{\{(v_1, 1), \dots, (v_\ell, \ell)\} \mid \exists \tau \in \mathsf{Ans}(\varphi, \mathcal{D})\ \forall i \in [\ell] : \tau(x_i) = v_i\}$

Observation 25. *Given an ECQ φ and a database $\mathcal{D}$ with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$, the hyperedges of $H(\varphi, \mathcal{D})$ are in bijection with the elements of* $\mathsf{Ans}(\varphi, \mathcal{D})$.

By Observation 25, the problem of approximating $|\mathsf{Ans}(\varphi, \mathcal{D})|$ reduces immediately to approximating the number of hyperedges of $H(\varphi, \mathcal{D})$. The latter can be achieved by Theorem 17 as long as we can (efficiently) simulate the oracle to evaluate $\mathrm{EdgeFree}(H(\varphi, \mathcal{D})[V_1, \dots, V_\ell])$ for any ℓ-partite subset $(V_1, \dots, V_\ell)$ of $V(H(\varphi, \mathcal{D}))$.

We will show later that the most important case is where, for each $i \in [\ell]$, $V_i \subseteq U_i(\mathcal{D})$. It turns out that, in this case, the evaluation of the predicate $\mathrm{EdgeFree}(H(\varphi, \mathcal{D})[V_1, \dots, V_\ell])$ can be reduced to deciding the existence of a homomorphism between two structures which, intuitively, can be viewed as coloured versions of $\mathcal{A}(\varphi)$ and $\mathcal{B}(\varphi, \mathcal{D})$. We define these coloured versions in Definitions 26 and 28 respectively. The colouring arises in Definition 28 in the following manner. Let r and b be two colours. To handle disequalities, we introduce a collection of colouring functions $\mathbf{f} = \{f_\eta\}$, where for each $\eta = \{x_i, x_j\} \in \Delta(\varphi)$, f_η is a function $f_\eta : U(\mathcal{D}) \mapsto \{r, b\}$.

Definition 26 ($\widehat{\mathcal{A}}(\varphi)$)**.** Let φ be an ECQ. Let $\ell = |\mathrm{free}\,(\varphi)\,|$ and $k = |\mathrm{vars}\,(\varphi)\,| - \ell$. Let $\{x_1, \dots, x_{\ell+k}\}$ be an enumeration of the variables in $\mathrm{vars}\,(\varphi)$. Recall the definition of $\mathcal{A}(\varphi)$ from Definition 18. The structure $\widehat{\mathcal{A}}(\varphi)$ is a modification of $\mathcal{A}(\varphi)$ defined as follows.

- $U(\widehat{\mathcal{A}}(\varphi)) = \mathrm{vars}\,(\varphi) = U(\mathcal{A}(\varphi))$.
- For each $R \in \mathrm{sig}(\mathcal{A}(\varphi))$, $R^{\widehat{\mathcal{A}}(\varphi)} = R^{\mathcal{A}(\varphi)}$.
- For each variable x_i of φ, $\widehat{\mathcal{A}}(\varphi)$ has a unary relation $P_i^{\widehat{\mathcal{A}}(\varphi)} := \{x_i\}$.
- For each $\eta = \{x_i, x_j\}$ in $\Delta(\varphi)$ with $i < j$, $\widehat{\mathcal{A}}(\varphi)$ has unary relations $R_\eta^{\widehat{\mathcal{A}}(\varphi)} := \{x_i\}$ and $B_\eta^{\widehat{\mathcal{A}}(\varphi)} := \{x_j\}$.

Observation 27. *$\widehat{\mathcal{A}}(\varphi)$ is obtained from $\mathcal{A}(\varphi)$ by adding $|\mathrm{vars}\,(\varphi)\,| + 2|\Delta(\varphi)|$ unary relations. Since there are at most $\binom{|\mathrm{vars}(\varphi)|}{2}$ disequalities in $\Delta(\varphi)$, at most $|\mathrm{vars}\,(\varphi)\,|^2$ unary relations are added in all. Thus, $\|\widehat{\mathcal{A}}(\varphi)\| \le \|\mathcal{A}(\varphi)\| + 2|\mathrm{vars}\,(\varphi)\,|^2$. By Observation 19, $\|\widehat{\mathcal{A}}(\varphi)\| \le 3\|\varphi\| + 2|\mathrm{vars}\,(\varphi)\,|^2 \le 5\|\varphi\|^2$.*

Definition 28 ($\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$)**.** Let φ be an ECQ and let $\mathcal{D}$ be a database with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$. Let $\ell = |\mathrm{free}\,(\varphi)\,|$ and $k = |\mathrm{vars}\,(\varphi)\,| - \ell$. Let $(V_1, \dots, V_\ell)$ be an ℓ-partite subset of $V(H(\varphi, \mathcal{D})) = \bigcup_{i=1}^{\ell} U_i(\mathcal{D})$ (from Definition 24, recall that $U_i(\mathcal{D}) = U(\mathcal{D}) \times \{i\}$ from Definition 23). Let $\{x_1, \dots, x_\ell\}$ be an enumeration of the variables in free (φ) and let $\{x_1, \dots, x_{\ell+k}\}$ be an enumeration of the variables in vars (φ). Recall the definition of $\mathcal{B}(\varphi, \mathcal{D})$ from Definition 20. The structure $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$ is a modification of $\mathcal{B}(\varphi, \mathcal{D})$ defined as follows. To avoid notational clutter, we will just write $\widehat{\mathcal{B}}$, rather than $\widehat{\mathcal{B}}(\varphi, \mathcal{B}, V_1, \dots, V_\ell, \mathbf{f})$.

- For each $i \in \{1, \dots, \ell\}$ let $S_i = V_i$ and for each $i \in \{\ell+1, \dots, \ell+k\}$ let $S_i = U_i(\mathcal{D})$. Define the universe of $\widehat{\mathcal{B}}$ as $U(\widehat{\mathcal{B}}) = \bigcup_{i=1}^{\ell+k} S_i$.
- For each arity-a relation symbol $R \in \mathrm{sig}(\mathcal{B}(\varphi, \mathcal{D}))$, $\widehat{\mathcal{B}}$ has the arity-a relation
$$R^{\widehat{\mathcal{B}}} := \{((w_1, i_1), \dots, (w_a, i_a)) \in U(\widehat{\mathcal{B}})^a \mid (w_1, \dots, w_a) \in R^{\mathcal{B}(\varphi, \mathcal{D})}\}.$$
Here $i_1, \dots, i_a$ are any values such that $((w_1, i_1), \dots, (w_a, i_a)) \in U(\widehat{\mathcal{B}})^a$.
- For each variable x_i of φ, $\widehat{\mathcal{B}}$ has a unary relation $P_i^{\widehat{\mathcal{B}}} := S_i$.
- For each $\eta \in \Delta(\varphi)$, we add to $\widehat{\mathcal{B}}$ the unary relation $R_\eta^{\widehat{\mathcal{B}}} := \{(x_i, j) \in U(\widehat{\mathcal{B}}) \mid f_\eta(x_i) = r\}$ and the unary relation $B_\eta^{\widehat{\mathcal{B}}} := \{(x_i, j) \in U(\widehat{\mathcal{B}}) \mid f_\eta(x_i) = b\}$. Here, j is any value such that $(x_i, j) \in U(\widehat{\mathcal{B}}$.

Observation 29. *To avoid notational clutter, let $\widehat{\mathcal{B}} = \widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$. Note that $|U(\widehat{\mathcal{B}})| \le \mathrm{vars}(\varphi) \cdot |U(\mathcal{D})| = |\mathrm{vars}(\varphi)| \cdot |U(\mathcal{B}(\varphi, \mathcal{D}))|$. Let $a = \mathrm{ar}(\mathrm{sig}(\varphi)) = \mathrm{ar}(\mathrm{sig}(\mathcal{B}(\varphi, \mathcal{D})))$. Assume that φ is not trivial, so $a \ge 1$. For each relation $R^{\mathcal{B}(\varphi,\mathcal{D})}$ of $\mathcal{B}(\varphi, \mathcal{D})$, $|R^{\widehat{\mathcal{B}}}| \le |\mathrm{vars}(\varphi)|^a \cdot |R^{\mathcal{B}(\varphi,\mathcal{D})}|$. Additionally, we add at most $|\mathrm{vars}(\varphi)|^2$ unary relations to $\widehat{\mathcal{B}}$, each of size at most $|U(\widehat{\mathcal{B}})| \le |\mathrm{vars}(\varphi)| \cdot |U(\mathcal{D})|$. Therefore,*

$$\begin{aligned}\|\widehat{\mathcal{B}}\| &\le |\mathrm{sig}(\widehat{\mathcal{B}})| + |U(\widehat{\mathcal{B}})| + |\mathrm{vars}(\varphi)|^a \sum_{R \in \mathrm{sig}(\mathcal{B}(\varphi,\mathcal{D}))} |R^{\mathcal{B}(\varphi,\mathcal{D})}| \mathrm{ar}(R) + |\mathrm{vars}(\varphi)|^2 |\mathrm{vars}(\varphi)| \cdot |U(\mathcal{D})| \\ &\le |\mathrm{vars}(\varphi)|^a \cdot \|\mathcal{B}(\varphi, \mathcal{D})\| + |\mathrm{vars}(\varphi)|^2 + |\mathrm{vars}(\varphi)|^3 |U(\mathcal{D})|.\end{aligned}$$

If φ has v negated predicates then, Observation 21 guarantees that

$$\|\mathcal{B}(\varphi, \mathcal{D})\| \le 2\|\varphi\|(\|\mathcal{D}\| + v|U(\mathcal{D})|^a)$$

so, plugging this in, $\|\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})\| \le \exp(\mathrm{O}(\|\varphi\|^2)) \cdot (\|\mathcal{D}\| + v|U(D)|^a)$.

Note that $\widehat{\mathcal{A}}(\varphi)$ and $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$ have the same signature by construction. Recall that our main goal is to simulate the oracle for EdgeFree$(H(\varphi, \mathcal{D})[V_1, \dots, V_\ell])$, and that we have stated (but not yet proved) that the most important case is when each $V_i \subseteq U_i(\mathcal{D})$. The following lemma establishes certain properties of the structures $\widehat{\mathcal{A}}(\varphi)$ and $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$ that apply in this case.

Lemma 30. *Let φ be an ECQ and let $\mathcal{D}$ be a database with $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$. Let $\ell = |\mathrm{free}(\varphi)|$. Let $(V_1, \dots, V_\ell)$ be an ℓ-partite subset of $V(H(\varphi, \mathcal{D}))$. Suppose that for each $i \in [\ell]$, $V_i \subseteq U_i(\mathcal{D})$. Then the hypergraph $H(\varphi, \mathcal{D})[V_1, \dots, V_\ell]$ has a hyperedge if and only if there is a collection $\mathbf{f}$ of colouring functions such that there is a homomorphism from $\widehat{\mathcal{A}}(\varphi)$ to $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$.*

Proof. Let $k = |\mathrm{vars}(\varphi)| - \ell$. Let $\{x_1, \dots, x_\ell\}$ be an enumeration of the variables in free (φ) and let $\{x_1, \dots, x_{\ell+k}\}$ be an enumeration of the variables in vars (φ). We consider both directions.

- If $H(\varphi, \mathcal{D})[V_1, \dots, V_\ell]$ has any hyperedges then, by Definition 24, it must contain a hyperedge of the form $\{(v_1, 1), \dots, (v_\ell, \ell)\}$, where $(v_i, i) \in V_i$ for all $i \in [\ell]$, and there is an assignment $\tau \in \mathsf{Ans}(\varphi, \mathcal{D})$ such that $\forall i \in [\ell]$ we have $\tau(x_i) = v_i$. Consequently by (2) there is a homomorphism h from $\mathcal{A}(\varphi)$ to $\mathcal{B}(\varphi, \mathcal{D})$ that extends τ and satisfies all disequalities in $\Delta(\varphi)$, that is, $h(x_i) \ne h(x_j)$ for all $\{x_i, x_j\} \in \Delta(\varphi)$.

Now fix such a homomorphism h and choose the collection $\mathbf{f}$ of colouring functions so that for each $\eta = \{x_i, x_j\} \in \Delta(\varphi)$ with $i < j$, f_η maps $h(x_i)$ to r and $h(x_j)$ to b, which is possible since $h(x_i) \neq h(x_j)$. The values of f_η on variables other than x_i and x_j are irrelevant and can be chosen arbitrarily.

Using h, we construct a homomorphism $\hat{h}$ from $\widehat{\mathcal{A}}(\varphi)$ to $\widehat{\mathcal{B}} = \widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$, proceeding as follows. The elements of $U(\widehat{\mathcal{A}}(\varphi))$ are the variables x_i of φ. For each x_i, take $\hat{h}(x_i) = (h(x_i), i)$. The fact that the non-unary relations are preserved by $\hat{h}$ follows from the fact that h is a homomorphism and from the definition of the predicates $R^{\widehat{B}}$. To preserve $P_i^{\widehat{\mathcal{A}}(\varphi)}$, $\hat{h}$ must map x_i to an element in S_i. In particular, for each free variable x_i, we have $\hat{h}(x_i) = (h(x_i), i) = (v_i, i) \in V_i$, which was noted above.

Now consider $\eta = \{x_i, x_j\} \in \Delta(\varphi)$ with $i < j$. To preserve $R_\eta^{\widehat{\mathcal{A}}(\varphi)}$ and $B_\eta^{\widehat{\mathcal{A}}(\varphi)}$, it must be the case that $f_\eta(h(x_i)) = r$ and $f_\eta(h(x_j)) = b$, which, however, is satisfied for our choice of f. Thus, $\hat{h}$ is a homomorphism as desired.

- In the other direction, suppose for some collection $\mathbf{f}$ of colouring functions that there is a homomorphism $\hat{h}$ from $\widehat{\mathcal{A}}(\varphi)$ to $\widehat{\mathcal{B}} = \widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$. The relation P_i ensures that $\hat{h}$ maps each x_i to S_i. So for each $i \in [\ell]$, $\hat{h}(x_i) = (w_i, i) \in V_i$ for some $w_i \in U(\mathcal{D})$.

  We construct a homomorphism h from $\mathcal{A}(\varphi)$ to $\mathcal{B}(\varphi, \mathcal{D})$ by setting $h(x_i) = w_i$ for all variables $x_i \in \text{vars}(\varphi)$. The relations from the signature of $\mathcal{A}(\varphi)$ ensure that h is a homomorphism from $\mathcal{A}(\varphi)$ to $\mathcal{B}(\varphi, \mathcal{D})$. Let $\tau = \text{proj}(h, \text{free}(\varphi))$. We will show that $\tau \in \text{Ans}(\varphi, \mathcal{D})$. By Definition 24, that implies that $\{(\tau(x_1), 1), \dots, (\tau(x_\ell), \ell)\}$ is a hyperedge of $H(\varphi, \mathcal{D})[V_1, \dots, V_\ell]$, completing the proof.

  So it remains to prove that $\tau \in \text{Ans}(\varphi, \mathcal{D})$. By (2) it suffices to show that for all $\{x_i, x_j\} \in \Delta(\varphi)$, we have $h(x_i) \neq h(x_j)$. To see this, consider $\{x_i, x_j\} \in \Delta(\varphi)$ and suppose that $i < j$. Then the relation R_η ensures that $h(x_i)$ is in $R_\eta^{\widehat{\mathcal{B}}}$ and that $h(x_j)$ is in $B_\eta^{\widehat{\mathcal{B}}}$. Since these two relations are disjoint, we find that $h(x_i) \neq h(x_j)$, as required.

□

Given Lemma 30, we will be able to use colour-coding to simulate the oracle for $\text{EdgeFree}(H(\varphi, \mathcal{D})[V_1, \dots, V_\ell])$ using an oracle for the decision homomorphism problem. Colour-coding is common in parameterised algorithms and our application is similar to the approach that has been used in the decision setting by Papadimitriou and Yannakakis [39] and Koutris et al. [32]. Using this, we are now able to prove Lemma 22, which we restate here for convenience.

Lemma 22. *There is a randomised algorithm that is equipped with oracle access to* Hom *and takes the following inputs*

- *an ECQ φ,*
- *a database $\mathcal{D}$ with $\text{sig}(\varphi) \subseteq \text{sig}(\mathcal{D})$,*
- *rational numbers ε and δ in $(0, 1)$.*

Let $a = \text{ar}(\text{sig}(\varphi))$ and let ν be the number of negated predicates in φ. The algorithm computes an (ε, δ)-approximation of $|\text{Ans}(\varphi, \mathcal{D})|$ in time

$$\exp(\mathrm{O}(\|\varphi\|^2)) \cdot \text{poly}(\log(1/\delta), \varepsilon^{-1}, \|\mathcal{D}\|, \nu |U(D)|^a) .$$

Each oracle query Hom$(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$ *that is made by the algorithm has the property that $\widehat{\mathcal{A}}$ can be obtained from $\mathcal{A}(\varphi)$ by adding unary relations and satisfies $\|\widehat{\mathcal{A}}\| \leq 5\|\varphi\|^2$.*

Proof. Let $(\varphi, \mathcal{D}, \varepsilon, \delta)$ be an input to the algorithm. Let $\ell = |\text{free}(\varphi)|$ and $k = |\text{vars}(\varphi)| - \ell$. Let $\{x_1, \dots, x_\ell\}$ be an enumeration of the variables in $\text{free}(\varphi)$ and let $\{x_1, \dots, x_{\ell+k}\}$ be an enumeration of the variables in $\text{vars}(\varphi)$.

To avoid notational clutter, we set $\mathcal{A} := \mathcal{A}(\varphi)$, $\mathcal{B} := \mathcal{B}(\varphi, \mathcal{D})$, $\Delta := \Delta(\varphi)$, $H := H(\varphi, \mathcal{D})$ and $V := V(H)$. Let $N = |V|$. Note that, by construction (Definition 24), $N = \ell \cdot |U(\mathcal{D})|$.

The goal of the algorithm is to provide an (ε, δ)-approximation of $|\mathsf{Ans}(\varphi, \mathcal{D})|$. By Observation 25, this is the same as providing an (ε, δ)-approximation of $|E(H)|$.

Simulating the oracle calls. Our goal is to apply Theorem 17 with $\varepsilon, \delta/2$, and H. To do this, we must provide a simulation strategy for an oracle query $\mathsf{EdgeFree}(H[W_1, \dots, W_\ell])$, where $(W_1, \dots, W_\ell)$ is any ℓ-partite subset of V. We must ensure that the probability that any simulation of the oracle fails (during the whole run of the algorithm from Theorem 17) is at most $\delta/2$. To do this, we provide a simulation strategy for an individual oracle call with failure probability at most $\delta/(2T)$, where $T = \Theta(\log(1/\delta)\varepsilon^{-2}\ell^{6\ell}(\log N)^{4\ell+7})$ is the upper bound on the number of calls to EdgeFree in Theorem 17. Note (by a union bound) that this implies that the overall probability that any oracle call fails is at most $\delta/2$.

We simulate an arbitrary oracle call $\mathsf{EdgeFree}(H[W_1, \dots, W_\ell])$ by evaluating $\ell!$ more restricted oracle calls, each with failure probability at most $\delta/(2T\ell!)$ (which gives the desired failure probability of at most $\delta/(2T)$ by a union bound). Each of the restricted oracle calls is of the form $\mathsf{EdgeFree}(H[V_1, \dots, V_\ell])$ where each $V_i \subseteq U_i(\mathcal{D})$. The restricted oracle calls are chosen by considering each permutation π of $[\ell]$ and setting $V'_i = W_i \cap U_{\pi(i)}(\mathcal{D})$. Since (from Definition 24) each hyperedge of H contains exactly one element from each $U_i(\mathcal{D})$ with $i \in [\ell]$, it is the case that $H[W_1, \dots, W_\ell]$ has a hyperedge if and only if there is a permutation π so that $H[V'_1, \dots, V'_\ell]$ has a hyperedge. Equivalently, setting $V_i = V'_{\pi(i)}$, $H[W_1, \dots, W_\ell]$ has a hyperedge if and only if there is a permutation π so that $H[V_1, \dots, V_\ell]$ has a hyperedge.

By Lemma 30, simulating the restricted oracle call $\mathsf{EdgeFree}(H[V_1, \dots, V_\ell])$ is equivalent to checking whether there is a collection $\mathbf{f}$ of colouring functions such that there is a homomorphism from $\widehat{\mathcal{A}} := \widehat{\mathcal{A}}(\varphi)$ to $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$. For a given collection $\mathbf{f}$ of colouring functions, we use access to the oracle for Hom to determine whether such a homomorphism exists.

We now consider how to simulate the restricted oracle call $\mathsf{EdgeFree}(H[V_1, \dots, V_\ell])$. Let $Q = \lceil \log(2T\ell!/\delta) \rceil 4^{|\Delta|}$. We make Q repetitions, each time choosing the collection $\mathbf{f}$ of colouring functions uniformly at random. During a given repetition, we choose the collection $\mathbf{f}$ as follows. For each $\eta \in \Delta$ and each $u \in U(\mathcal{D})$, with probability $1/2$, we set $f_\eta(u) = r$. Otherwise, we set $f_\eta(u) = b$. Having chosen $\mathbf{f}$, we query the oracle for Hom to determine whether there is a homomorphism from $\widehat{\mathcal{A}}$ to $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$. If, for any of the Q choices of $\mathbf{f}$, the oracle for Hom finds a homomorphism, the simulation announces that $H[V_1, \dots, V_\ell]$ has a hyperedge. Otherwise, it announces that $H[V_1, \dots, V_\ell]$ has no hyperedge.

Failure probability of the simulation. We now bound the failure probability of the simulation. If there is no $\mathbf{f}$ such that there is a homomorphism from $\widehat{\mathcal{A}}$ to $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$ then the simulation is correct. Suppose instead that there is a collection $\mathbf{f}'$ of colouring functions such that there is a homomorphism h from $\widehat{\mathcal{A}}$ to $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f}')$. When $\mathbf{f}$ is chosen uniformly at random during one of the repetitions, the probability that h is a homomorphism from $\widehat{\mathcal{A}}$ to $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$ is at least the probability that, for each $\eta = \{x_i, x_j\} \in \Delta$, we have $f_\eta(x_i) = f'_\eta(x_i)$ and $f_\eta(x_j) = f'_\eta(x_j)$. This probability is at least $4^{-|\Delta|}$. Thus, the probability that all Q guesses fail is at most $(1 - 4^{-|\Delta|})^Q \leq \exp(-Q4^{-|\Delta|}) \leq \delta/(2T\ell!)$, as required.

Size of the constructed structures. For a fixed ℓ-partite subset $(V_1, \dots, V_\ell)$ of V and collection $\mathbf{f}$ of colouring functions, consider the structures $\widehat{\mathcal{A}}$ and $\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \dots, V_\ell, \mathbf{f})$. It follows from Observation 27 that $\widehat{\mathcal{A}}$ is obtained from $\mathcal{A}$ by adding unary relations and that $\|\widehat{\mathcal{A}}\| \leq 5\|\varphi\|^2$ (as required). It is clear from Definitions 18 and 26 that the time needed to construct $\widehat{\mathcal{A}}$ is linear in its size. Moreover, Observation 29 guarantees that $\|\widehat{\mathcal{B}}(\varphi, \mathcal{D}, V_1, \ldots, V_\ell, \mathbf{f})\| \leq \exp(\mathrm{O}(\|\varphi\|^2)) \cdot (\|\mathcal{D}\| + \nu|U(D)|^a)$. It is clear from Definition 20 and 28 that the time needed to construct $\widehat{\mathcal{B}}$ is linear in its size.

Runtime analysis. First, we bound the number $X \leq T\ell!Q$ of oracle calls. Plugging in the definition of Q and applying crude upper bounds, we have $X = \mathrm{O}(T^2(\ell!)^2 \log(1/\delta) 4^{|\Delta|})$. Plugging in the definition of T and pulling out $\log(1/\delta)$ and ε^{-1} factors, we have

$$X = \mathrm{poly}(\log(1/\delta), \varepsilon^{-1})\mathrm{O}(\ell^{12\ell}(\log N)^{8\ell+14}(\ell!)^2 4^{|\Delta|}) .$$

Since ℓ^ℓ, $\ell!$ and $4^{|\Delta|}$ are all at most $\exp(\mathrm{O}(\|\varphi\|^2))$ we obtain

$$X = \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}) \cdot \exp(\mathrm{O}(\|\varphi\|^2)) \cdot \mathrm{O}((\log N)^{8\ell+14}).$$

If $\log N < e^\ell$ then final term can be subsumed by the $\exp(\mathrm{O}(\|\varphi\|^2))$ term. Otherwise, the final term is $\mathrm{O}(N) = \mathrm{O}(\ell|U(\mathcal{D})|)$. To see this, note that

$$(\log N)^{8\ell+14} = (e^\ell)^{8\log\log N} \cdot (\log N)^{14} \leq (\log N)^{8\log\log N+14} \in O(N)$$

Therefore,

$$X = \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}) \cdot \exp(\mathrm{O}(\|\varphi\|^2)) \cdot \mathrm{O}(|U(\mathcal{D})|).$$

Second, consider the time needed to construct an oracle query $\textsc{Hom}(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$. It is dominated by the time needed to construct $\widehat{\mathcal{B}}$, and the total running time is bounded from above by

$$\mathrm{poly}(\log(1/\delta), \varepsilon^{-1}) \cdot \exp(\mathrm{O}(\|\varphi\|^2)) \cdot \mathrm{O}(|U(\mathcal{D})|) \cdot \exp(\mathrm{O}(\|\varphi\|^2)) \cdot (\|\mathcal{D}\| + \nu|U(D)|^a),$$

which can easily be simplified to prove the lemma. □

4 FPTRAS FOR #ECQ WITH BOUNDED TREEWIDTH AND ARITY

The goal of this section is to establish Theorem 5, i.e., to construct an FPTRAS for #ECQ on classes of queries whose hypergraphs have bounded treewidth and arity. Thanks to Lemma 22, it will be sufficient to rely on an efficient algorithm for the decision version of the homomorphism problem. For the case of bounded arity, we will use the algorithm due to Dalmau et al. [14] (see [26, Theorem 3.1] for an explicit statement of the extension from graphs to structures).

For each structure $\mathcal{A}$, there is an associated *hypergraph of* $\mathcal{A}$, which we denote $H(\mathcal{A})$. Its vertices are $V(H(\mathcal{A})) = U(\mathcal{A})$, and $H(\mathcal{A})$ has a hyperedge e whenever there is a relation $R \in \mathrm{sig}(\mathcal{A})$ with $e \in R^{\mathcal{A}}$. The treewidth of $\mathcal{A}$ is the treewidth of its associated hypergraph.[7]

Given a class of structures $\mathcal{S}$, we write $\textsc{Hom}(\mathcal{S})$ for the restriction of $\textsc{Hom}$ in which the input $(\mathcal{A}, \mathcal{B})$ must satisfy $\mathcal{A} \in \mathcal{S}$.

Theorem 31 ([14, 26]). *Let t and a be positive integers. Let $\mathcal{S}$ be a class of structures such that every structure in $\mathcal{S}$ has treewidth at most t and arity at most a. Then $\textsc{Hom}(\mathcal{S})$ is polynomial-time solvable.*

We remark that Theorem 31 is in fact a weaker version of the corresponding theorem in [14, 26], which also applies to classes $\mathcal{S}$ in which only the *homomorphic cores* of members of $\mathcal{S}$ have bounded treewidth. We do not need the more general version here. The proof of Theorem 31 relies on the fact that there is a polynomial-time algorithm that takes as

[7]Note that the hypergraph $H(\varphi)$ of the query φ and the hypergraph of its associated structure $H(\mathcal{A}(\varphi))$ are the same. Consequently, the treewidth of φ and $\mathcal{A}(\varphi)$ are equal.

input a structure $\mathcal{A} \in \mathcal{S}$ and produces a tree decomposition of $H(\mathcal{A})$ with treewidth at most $4t + 4$ (the exact treewidth of the tree decomposition that is produced is not important, but it is important that it is $O(t)$).

We are now able to prove Theorem 5, which we restate here for convenience.

Theorem 5. *Let t and a be positive integers. Let C be a class of hypergraphs such that every member of C has treewidth at most t and arity at most a. Then #ECQ (Φ_C) has an FPTRAS, running in time* $\exp(O(||\varphi||^2)) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||)$.

Proof. Let t, a and C be as in the statement. Let $\mathcal{S}$ be the class of all structures with treewidth at most t and arity at most a.

By Theorem 31, there is a polynomial-time algorithm HomAlg for Hom($\mathcal{S}$).

The desired FPTRAS is now obtained by using Lemma 22 and simulating the oracle using the algorithm HomAlg. Given an input $(\varphi, \mathcal{D})$ with $\varphi \in \Phi_C$ and $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$, we wish to approximate $|\mathrm{Ans}(\varphi, \mathcal{D})|$. By the statement of Lemma 22, every oracle query $(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$ produced in the course of the approximation has the property that $\widehat{\mathcal{A}}$ is obtained from $\mathcal{A}(\varphi)$ by adding unary relations. Our goal is to show how to simulate the oracle call Hom($\widehat{\mathcal{A}}, \widehat{\mathcal{B}}$).

In order to use Algorithm HomAlg for the simulation, we need only show that $\widehat{\mathcal{A}}$ has treewidth at most t and arity at most a. Equivalently, we need to show that $H(\widehat{\mathcal{A}})$ has treewidth at most t and arity at most a.

Since $H(\varphi) \in C$, we know that $H(\mathcal{A}(\varphi))$ has treewidth at most t and arity at most a. Since $H(\widehat{\mathcal{A}})$ is obtained from $H(\mathcal{A}(\varphi))$ by adding size-1 hyperedges and $a \geq 1$, it is clear that the arity of $H(\widehat{\mathcal{A}})$ is at most a. To see that the treewidth of $H(\widehat{\mathcal{A}})$ is at most t, consider any tree decomposition $(T, \mathbf{B})$ of $H(\mathcal{A}(\varphi))$ with $\mathrm{tw}(T, \mathbf{B}) = t$. Construct a tree decomposition $(T', \mathbf{B}')$ of $H(\widehat{\mathcal{A}})$ as follows. Start by setting $(T', \mathbf{B}') = (T, \mathbf{B})$. Then, for every $v \in \mathrm{vars}\,(\varphi)$, check whether

(1) $\{v\}$ is a hyperedge in $E(H(\widehat{\mathcal{A}}))$, and
(2) v is not in any bag B_t of T

If this occurs, add a new leaf t' to T' with $B_{t'} = \{v\}$. It is clear that $(T', \mathbf{B}')$ is a tree decomposition of $H(\widehat{\mathcal{A}})$ with treewidth t.

Now by Lemma 22 and Theorem 31, the total running time is bounded by

$$\exp(O(||\varphi||^2)) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||) \cdot \mathrm{poly}(||\widehat{\mathcal{A}}|| + ||\widehat{\mathcal{B}}||)\,,$$

where $(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$ is the oracle query that maximises $||\widehat{\mathcal{A}}|| + ||\widehat{\mathcal{B}}||$.

Since the size $||\widehat{\mathcal{A}}|| + ||\widehat{\mathcal{B}}||$ must be bounded by $\exp(O(||\varphi||^2)) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||)$ (by the bound on the running time in Lemma 22), and $\mathrm{poly}(\exp(O(||\varphi||^2))) = \exp(O(||\varphi||^2))$ we can bound the overall running time by

$$\exp(O(||\varphi||^2)) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||)\,,$$

which concludes the proof. □

5 BEYOND BOUNDED ARITY

Theorem 5 gives an FPTRAS for #ECQ (Φ_C) where Φ_C is a set of ECQs φ for which $H(\varphi)$ has bounded treewidth and bounded arity. There are many notions of width that refine treewidth, but in the bounded arity case, it is also known that bounding these widths leads to the same class of queries as bounding treewidth.

When the arity restriction is relaxed, the different notions of width play a bigger role. In this section we examine the difficulty of #DCQ (Φ_C) and #CQ (Φ_C) when Φ_C does not impose a bound on arity. In Section 5.1 we extend Theorem 5 to this setting (in the case of #DCQ (Φ_C)), obtaining a stronger result in terms of adaptive width. In Section 5.2 we improve the result of [5] which gives an FPRAS for #CQ (Φ_C) by bounding fractional hypertreewidth rather than hypertreewidth.

5.1 FPTRAS for #DCQ with bounded Adaptive Width

The goal of this section is to prove Theorem 13. We start by defining the hypergraph width measures that we need, following the framework of [2].

Definition 32 (f-width)**.** Let H be a hypergraph. For any function $f : 2^{V(H)} \to \mathbb{R}_{\geq 0}$, the *$f$-width* of a tree decomposition $(T, \mathbf{B})$ of H is the maximum of $f(B_t)$ taken over all $t \in V(T)$. The *f-width of H*, denoted by $f(H)$, is the minimum f-width over all tree decompositions of H.

It is clear from Definition 4 that for $f(X) = |X| - 1$, the f-width of H is identical to the treewidth of H.

Definition 33 (fractional independent set, adaptive width)**.** A *fractional independent set* of a hypergraph H is a function $\mu : V(H) \to [0, 1]$ such that for all $e \in E(H)$, we have $\sum_{v \in e} \mu(v) \leq 1$. For $X \subseteq V(H)$, we define $\mu(X) = \sum_{v \in X} \mu(v)$. The *adaptive width* [36] of H, denoted by $\mathrm{aw}(H)$, is the supremum of μ-width(H), where the supremum is taken over all fractional independent sets μ of H.

Recall from Lemma 12 that treewidth is strongly dominated by adaptive width. For the sake of completeness, we observe that this strict domination requires unbounded arity.

Observation 34. *Let H be a hypergraph with arity a. Then $\mathrm{tw}(H) \leq a \cdot \mathrm{aw}(H) - 1$.*

Proof. If $a = 0$ then H has no hyperedges and thus $\mathrm{tw}(H) = -1 = a \cdot \mathrm{aw}(H) - 1$. Otherwise, set $\mu(v) := 1/a$ and observe that μ is a fractional independent set. Hence there exists a tree decomposition $(T, \mathbf{B})$ of μ-width(H) at most $\mathrm{aw}(H)$, that is, for each $t \in V(T)$ we have $\mu(B_t) \leq \mathrm{aw}(H)$. Since $\mu(B_t) = |B_t|/a$, we conclude that $|B_t| \leq a \cdot \mathrm{aw}(H)$ and thus the treewidth is bounded by $a \cdot \mathrm{aw}(H) - 1$. □

Theorem 13 improves Theorem 5 for the special case where the queries are DCQs. In particular, it gives an FPTRAS for #DCQ (Φ_C) for every class C of hypergraphs with bounded adaptive width. The FPTRAS uses Lemma 22. In order to use this lemma, we first show that adding unary relations to a structure cannot increase its adaptive width in any harmful way.

Lemma 35. *Let $\mathcal{A}$ be a structure. Let $\widehat{\mathcal{A}}$ be a structure with universe $U(\widehat{\mathcal{A}}) = U(\mathcal{A})$ and signature $\mathrm{sig}(\widehat{\mathcal{A}}) = \mathrm{sig}(\mathcal{A}) \cup \rho$ where ρ is a set of arity-1 relation symbols. Suppose that, for each $R \in \mathrm{sig}(\mathcal{A})$, we have that $R^{\widehat{\mathcal{A}}} = R^{\mathcal{A}}$. Then $\mathrm{aw}(\widehat{\mathcal{A}}) \leq \max\{\mathrm{aw}(\mathcal{A}), 1\}$.*

Proof. Let $H = H(\mathcal{A})$, and let $\widehat{H} = H(\widehat{\mathcal{A}})$. Observe that $\widehat{H}$ is obtained from H by adding edges of arity 1. Also, $V(\widehat{H}) = V(H)$. From now on, we will refer to this vertex set as V.

We start by observing that every fractional independent set of H is a fractional independent set of $\widehat{H}$, and vice-versa.

Let $\hat{b} = \max\{\mathrm{aw}(H), 1\}$. We claim that $\mathrm{aw}(\widehat{H}) \leq \hat{b}$. To see this, consider any fractional independent set μ of $\widehat{H}$. We wish to show that μ-width$(\widehat{H}) \leq \hat{b}$, which means that there is a tree decomposition $(T', \mathbf{B}')$ of $\widehat{H}$ such that every $t \in V(T')$ satisfies $\mu(B_t) \leq \hat{b}$.

We start with a tree decomposition $(T, \mathbf{B})$ such that the μ-width of $(T, \mathbf{B})$ is at most $\mathrm{aw}(H)$. Consider the tree decomposition $(T', \mathbf{B}')$ of $\widehat{H}$ constructed as in the proof of Theorem 5. Start by setting $(T', \mathbf{B}') = (T, \mathbf{B})$. Then, for every $v \in \mathrm{vars}(\varphi)$, check whether

(1) $\{v\}$ is a hyperedge in $E(\widehat{H})$, and
(2) v is not in any bag B_t of T

If this occurs, add a new leaf t' to T' with $B_{t'} = \{v\}$. The claim follows since, for any $t \in V(T)$, $\mu(B_t) \leq \mathrm{aw}(H) \leq \hat{b}$. Also, for any $t \in V(T') \setminus V(T)$, $\mu(B_t) \leq 1 \leq \hat{b}$. We conclude that $\mathrm{aw}(\widehat{H}) \leq \hat{b}$, as desired. □

Recall that, given a class of structures $\mathcal{S}$, Hom($\mathcal{S}$) is the problem of deciding, given a pair $(\mathcal{A}, \mathcal{B})$ of structures with $\mathcal{A} \in \mathcal{S}$, whether there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$. The following algorithmic result is due to Marx, though we rephrase it in terms of homomorphisms between structures.

Theorem 36 ([37, Theorem 4.1]). *Let $\hat{b}$ be a positive integer. Let $\mathcal{S}$ be a class of structures such that every structure in $\mathcal{S}$ has adaptive width at most $\hat{b}$. Then there is a (computable) function f such that* Hom($\mathcal{S}$) *can be solved in time $f(\|\mathcal{A}\|) \cdot \mathrm{poly}(\|\mathcal{A}\| + \|\mathcal{B}\|)$.*

We are now able to prove Theorem 13, which we restate for convenience.

Theorem 13. *Let b be a positive integer. Let C be a class of hypergraphs such that every member of C has adaptive width at most b. Then* #DCQ (Φ_C) *has an FPTRAS.*

Proof. Let b and C be as in the statement. Let $\mathcal{S}$ be the class of all structures with adaptive width at most $\hat{b} = \max\{b, 1\}$. By Theorem 36, there is a (computable) function f, a polynomial p and an algorithm HomAlg for Hom($\mathcal{S}$) which, given input $(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$, runs in time $f(\|\mathcal{A}\|) \cdot \mathrm{poly}(\|\mathcal{A}\| + \|\mathcal{B}\|)$.

The desired FPTRAS is now obtained by using Lemma 22 and simulating the oracle using the algorithm HomAlg. Given an input $(\varphi, \mathcal{D})$ with $\varphi \in \Phi_C$ and $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$, we wish to approximate $|\mathrm{Ans}(\varphi, \mathcal{D})|$. By the statement of Lemma 22, every oracle query $(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$ produced in the course of the approximation has the property that $\widehat{\mathcal{A}}$ can be obtained from $\mathcal{A}(\varphi)$ by adding unary relations and satisfies $\|\widehat{\mathcal{A}}\| \leq 5\|\varphi\|^2$. Our goal is to show how to simulate the oracle call Hom$(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$.

In order to use algorithm HomAlg for the simulation, we need only show that $\widehat{\mathcal{A}}$ has adaptive width at most $\hat{b}$. This follows from Lemma 35.

Now by Lemma 22 and Theorem 36, since the input formula φ has no negated predicates, that is, the quantity ν in the statement of Lemma 22 is 0, the total running time is at most

$$\exp(\mathrm{O}(||\varphi||^2)) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||) \cdot f(||\widehat{\mathcal{A}}||) \cdot p(||\widehat{\mathcal{A}}|| + ||\widehat{\mathcal{B}}||),$$

where $(\widehat{\mathcal{A}}, \widehat{\mathcal{B}})$ is the oracle query that maximises $f(||\widehat{\mathcal{A}}||) \cdot p(||\widehat{\mathcal{A}}|| + ||\widehat{\mathcal{B}}||)$.

Since $\|\widehat{\mathcal{B}}\|$ must be bounded by $\exp(\mathrm{O}(||\varphi||^2)) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||)$ (by the running time in Lemma 22), and $\|\widehat{\mathcal{A}}\|$ is bounded by $5\|\varphi\|^2$, as we have already mentioned, there is a (computable) function $\hat{f}$ such that the running time is bounded by

$$\hat{f}(||\varphi||) \cdot \mathrm{poly}(\log(1/\delta), \varepsilon^{-1}, ||\mathcal{D}||),$$

which yields the desired FPTRAS and thus concludes the proof. □

5.2 FPRAS for #CQ with bounded Fractional Hypertreewidth

Recall the definition of tree decomposition (Definition 4) — a *tree decomposition* of a hypergraph H is a pair $(T, \mathbf{B})$ where T is a (rooted) tree and $\mathbf{B}$ assigns a subset $B_t \subseteq V(H)$ (called a *bag*) to each $t \in V(T)$. The following two conditions are satisfied: (i) for each $e \in E(H)$ there exists $t \in V(T)$ such that $e \subseteq B_t$, and (ii) for each $v \in V(H)$ the set $\{t \in V(T) \mid v \in B_t\}$ induces a (connected) subtree of T.

We will use the following notation associated with a tree decomposition. Let t^* be the root of T. Given a vertex $t \in V(T)$, let T_t denote the subtree of T rooted at t.

Definition 37 (hypertree decomposition, guard, hypertreewidth)**.** A *hypertree decomposition* [25, Definition A.1] of a hypergraph H is a triple (T, B, Γ) where $(T, \mathbf{B})$ is a tree decomposition of H and Γ assigns a subset $\Gamma_t \subseteq E(H)$ (called a *guard*) to each $t \in V(T)$. In addition to the two conditions that $(T, \mathbf{B})$ satisfies, the following conditions are satisfied: (iii) for each $t \in V(T)$, $B_t \subseteq \cup_{e \in \Gamma_t} e$. (iv) for each $t \in V(T)$, $(\cup_{e \in \Gamma_t} e) \cap (\cup_{t' \in V(T_t)} B_{t'}) \subseteq B_t$. The hypertreewidth of the decomposition (T, B, Γ) is the maximum cardinality of a guard. The hypertreewidth of H, denoted by $\mathrm{hw}(H)$, is the minimum hypertreewidth of any hypergraph decomposition of H.

Arenas et al. [5, Theorem 3.2] prove the following result.

Theorem 38 (Arenas, Croquevielle, Jayaram, Riveros)**.** *Let b be a positive integer. Let C be a class of hypergraphs such that every member of C has hypertreewidth at most b. Then #CQ (Φ_C) has an FPRAS.*

Theorem 38 is incomparable to Theorem 13. Theorem 13 is stronger in the sense that it applies to DCQs rather than just to CQs (indeed, we have already seen in Observation 10 that Theorem 38 cannot be extended to DCQs unless NP = RP). Theorem 13 is also stronger in the sense that it only requires bounded adaptive width instead of bounded hypertreewidth — this gives a more general result since adaptive width strongly dominates hypertreewidth (Lemma 12 here, from [37]). However, Theorem 38 is stronger than Theorem 13 in the sense that it provides an FPRAS instead of just an FPTRAS.

In this section, we prove Theorem 16 which strengthens Theorem 38 by bounding *fractional hypertreewidth* instead of hypertreewidth. This is a stronger result since fractional hypertreewidth strongly dominates hypertreewidth (Lemma 12). Theorem 16 is still incomparable to Theorem 13 but the remaining gap now coincides with the gap between polynomial-time solvability and fixed-parameter tractability for the corresponding decision problems, see [34] versus [37].

5.2.1 Fractional hypertreewidth.

Definition 39. ($H[X]$, fractional edge cover, $\mathrm{fcn}(H)$) Let H be a hypergraph and let X be a subset of $V(H)$. The hypergraph *induced* by X, denoted by $H[X]$, is the hypergraph with $V(H[X]) = X$ and $E(H[X]) = \{e \cap X \mid e \in E(H), e \cap X \neq \emptyset\}$. A *fractional edge cover* of a hypergraph H is a function $\gamma : E(H) \longrightarrow [0, 1]$ such that for all $v \in V(H)$, we have $\sum_{e \in E(H) \mid v \in e} \gamma(e) \geq 1$. The fractional edge cover number of H, denoted by $\mathrm{fcn}(H)$, is the minimum of $\sum_{e \in E(H)} \gamma(e)$, over all fractional edge covers γ of H.

We will use the following fact about fractional edge covers.

Observation 40. *Given a hypergraph H and subsets $B \subseteq B' \subseteq V(H)$, we have that* $\mathrm{fcn}(H[B]) \leq \mathrm{fcn}(H[B'])$.

Proof. Consider a fractional edge cover γ' of $H[B']$ such that $\mathrm{fcn}(H[B']) = \sum_{e' \in E(H[B'])} \gamma'(e')$. Define a fractional edge cover γ of $H[B]$ as follows. Note that $E(H[B]) = \{e' \cap B \mid e' \in E(H[B']), e' \cap B \neq \emptyset\}$. For each $e \in E(H[B])$ let $\gamma(e) = \min\{1, \sum_{e' \in E(H[B']), e' \cap B = e} \gamma'(e')\}$. We next show that γ is a fractional edge cover. First, if $v \in V(H[B])$ is contained in a hyperedge $e'' \in E(H[B])$ where $\sum_{e' \in E(H[B']), e' \cap B = e''} \gamma'(e')\} > 1$ then $\gamma(e'') = 1$ so

$\sum_{e\in E(H[B]),v\in e}\gamma(e) \geq \gamma(e'') = 1$. Otherwise, note that for every $v \in V(H[B])$ we have

$$\sum_{e\in E(H[B]),v\in e}\gamma(e) = \sum_{e\in E(H[B]),v\in e}\;\sum_{e'\in E(H[B']),e'\cap B=e}\gamma'(e') = \sum_{e'\in E(H[B']),v\in e'}\gamma'(e') \geq 1.$$

Since γ is a fractional edge cover of $H[B]$,

$$\mathrm{fcn}(H[B]) \leq \sum_{e\in E(H[B])}\gamma(e) \leq \sum_{e'\in E(H[B'])}\gamma'(e') = \mathrm{fcn}(H[B']).$$

□

The following definition of fractional hypertreewidth, from [27, 37], builds on the definition of f-width (Definition 32).

Definition 41 (fractional hypertreewidth, fhw(H))**.** The *fractional hypertreewidth* of H, denoted by fhw(H), is its f-width where $f(X) = \mathrm{fcn}(H[X])$.

The proof of Theorem 16 follows the same approach that Arenas et al. [5] used to prove Theorem 38 — namely, given an input $(\varphi, \mathcal{D})$ the following are constructed in polynomial time. First, a tree decomposition $(T, \mathbf{B})$ of $H(\varphi)$ with bounded fractional hypertree width. Then, using $(T, \mathbf{B})$, a tree automaton $\mathcal{T}$ with the property that $|\mathsf{Ans}(\varphi, \mathcal{D})|$ is exactly the same as the number of "size N" inputs that are accepted by $\mathcal{T}$, where $N = |V(T)|$. The theorem then follows from Corollary 4.9 of [5] which gives an FPRAS for counting these accepted inputs.

5.2.2 *Finding a nice tree decomposition and enumerating solutions.*

We will be interested in certain tree decompositions called *nice* tree decompositions [12, Section 7.2].

Definition 42. (nice tree decomposition) A tree decomposition $(T, \mathbf{B})$ of a hypergraph is said to be *nice* if the following conditions are satisfied:

- the bags assigned to the root and leaf nodes of T are empty,
- every internal node of T has at most two children,
- every internal node t of T with exactly two children t_1 and t_2 has $B_t = B_{t_1} = B_{t_2}$,
- and every internal node t of T with exactly one child t_1 has the property that the symmetric difference of B_t and B_{t_1} has exactly one element.

The following lemma builds on a tree decomposition of [34].

Lemma 43. *Let b be a positive integer and let C be the set of hypergraphs with fractional treewidth at most b. There is a polynomial-time algorithm that takes as input a CQ $\varphi \in \Phi_C$ and a database $\mathcal{D}$ with* $\mathrm{sig}(\varphi) \subseteq \mathrm{sig}(\mathcal{D})$ *and returns a nice tree decomposition of $H(\varphi)$ with fractional hypertreewidth at most* $7b^3 + 31b + 7$

Proof. Let $n = \|\varphi\|$ and $m = \|\mathcal{D}\|$. It is immediate from its definition that the hypergraph $H := H(\varphi)$ can be constructed in poly(n) time. Since H has fractional hypertreewidth at most b, from [34, Theorem 4.1] there is a polynomial-time algorithm (in n) for finding a tree decomposition $(T, \mathbf{B})$ of H with fractional hypertreewidth at most $c = 7b^3 + 31b + 7$.

In poly(n) time, the tree decomposition $(T, \mathbf{B})$ can be turned into a nice one. The construction is standard, so we just give a sketch. First, add a new root and new leaves with empty bags. (The new root has one child, which is the original root. Each original leaf has exactly one child, which is a new leaf.) Then process the nodes from the root working down to the leaves as follows. If t has $k \geq 2$ children $t_1, \ldots, t_k$ then this is replaced with a (nearly) complete binary tree below t with k leaves $t'_1, \ldots, t'_k$ — all nodes in this nearly complete binary tree are given bag B_t. The new node t'_i is given the child t_i.

Finally, if a node t has exactly one child t_1 but it is not true that the symmetric difference of B_t and B_{t_1} has exactly one element, then we replace the edge from t to t_1 with a path from t to t_1, Along this path, vertices in $B_t \setminus B_{t_1}$ are dropped one-by-one then vertices in $B_{t_1} \setminus B_t$ are added.

Every bag in the nice tree decomposition is a subset of a bag in the original tree decomposition. Therefore Observation 40 ensures that the nice tree composition also has fractional hypertreewidth at most c. □

We now extend the definitions concerning assignments from Section 1. To use these in the context of tree decompositions, it helps to remember (taking $H(\varphi)$ to be the hypergraph associated with a CQ φ from Definition 3) that the vertex set of $H(\varphi)$ is defined by $V(H(\varphi)) = \text{vars}(\varphi)$.

Definition 44. (consistent assignments, proj) Let φ be a CQ and let $\mathcal{D}$ be a database. Suppose that B and B' are subsets of vars (φ). We say that assignments $\tau\colon B \to U(\mathcal{D})$ and $\tau'\colon B' \to U(\mathcal{D})$ are *consistent* if, for every $v \in B \cap B'$, $\tau(v) = \tau'(v)$. We use $\text{proj}(\tau, B')$ to denote τ's projection onto B', which is the assignment from $B \cap B'$ to $U(\mathcal{D})$ that is consistent with τ.

Note that the definition of $\text{proj}(\tau, B')$ is consistent with the one given in Definition 2 for the special case where $B = \text{vars}(\varphi)$ and $B' = \text{free}(\varphi)$.

Definition 45. (comp) Let φ be a CQ and let $\mathcal{D}$ be a database. Suppose that B and B' are subsets of vars (φ) and that $\tau\colon B \to U(\mathcal{D})$ and $\tau'\colon B' \to U(\mathcal{D})$ are consistent. We define their composition $\text{comp}(\tau, \tau')$ to be the unique assignment from $B \cup B'$ to $U(\mathcal{D})$ that is consistent with both τ and τ'.

Definition 46. (proj, as applied to sets of assignments) Let φ be a CQ and let $\mathcal{D}$ be a database. If $\mathcal{L}$ is a set of assignments, each from a subset of vars (φ) to $U(\mathcal{D})$, we use $\text{proj}(\mathcal{L}, B)$ to denote $\{\text{proj}(\tau, B) \mid \tau \in \mathcal{L}\}$.

Using this notation, the set $\text{Ans}(\varphi, \mathcal{D})$ is $\text{proj}(\text{Sol}(\varphi, \mathcal{D}), \text{free}(\varphi))$.

Definition 47. ($\text{Sol}(\varphi, \mathcal{D}, B)$) Let φ be a CQ and let $\mathcal{D}$ be a database with $\text{sig}(\varphi) \subseteq \text{sig}(\mathcal{D})$. Let B be a subset of vars (φ) A *solution* of $(\varphi, \mathcal{D}, B)$ is an assignment $\alpha\colon B \to U(\mathcal{D})$ such that for every atom $R_i(x_{i,1}, \ldots, x_{i,j})$ of φ there is an assignment $\alpha_i\colon \text{vars}(\varphi) \to U(\mathcal{D})$ which is consistent with α and has the property that $R_i^{\mathcal{D}}(\alpha_i(x_{i,1}), \ldots, \alpha_i(x_{i,j}))$ is a fact in $\mathcal{D}$. Let $\text{Sol}(\varphi, \mathcal{D}, B)$ be the set of solutions of $(\varphi, \mathcal{D}, B)$.

Note that a solution of $(\varphi, \mathcal{D})$, as defined in Definition 1, is the same as a solution of $(\varphi, \mathcal{D}, \text{vars}(\varphi))$.

The following Lemma is immediate from Theorem 3.5 of [27] (though their version is stated in the language of CSPs).

Lemma 48. *(Grohe, Marx) Let c be a positive integer. There is a polynomial-time algorithm that takes as input*

- *a CQ φ and a database $\mathcal{D}$ with $\text{sig}(\varphi) \subseteq \text{sig}(\mathcal{D})$, and*
- *a subset $B \subseteq \text{vars}(\varphi)$ such that $\text{fcn}(H[B]) \leq c$, where $H = H(\varphi)$.*

The algorithm returns $\text{Sol}(\varphi, \mathcal{D}, B)$.

5.2.3 Tree automata. We now give the tree automaton definitions from [5]. The tree automata that we define are a restricted form of tree automata, corresponding to those whose trees have degree at most 2, but this is all that we will need.

Definition 49. ($\mathrm{Trees}_2[\Sigma]$) Let Σ be a finite alphabet. $\mathrm{Trees}_2[\Sigma]$ is the set of pairs (T, ψ) where T is a rooted tree in which each vertex has at most 2 children and $\psi \colon V(T) \to \Sigma$ assigns a label to each node of T.

Definition 50. (tree automaton, run, accepts, N-slice) A *tree automaton* $\mathcal{A}$ is a tuple (S, Σ, Δ, s_0) where S is a finite set of states and $s_0 \in S$ is the initial state. Σ is a finite alphabet. The transition function Δ is a function from a subset of $S \times \Sigma$ to $\{\emptyset\} \cup S \cup (S \times S)$. A *run* ρ over a pair $(T, \psi) \in \mathrm{Trees}_2[\Sigma]$ is a function $\rho \colon V(T) \to S$ that assigns a state to each node of T in such a way that

- for each leaf $t \in V(T)$ the transition $(\rho(t), \psi(t)) \to \emptyset$ is in Δ,
- for each node $t \in V(T)$ with exactly one child t_1, $(\rho(t), \psi(t)) \to \rho(t_1)$ is in Δ, and
- for each node $t \in V(T)$ with exactly two children t_1 and t_2 (ordered left-to-right), $(\rho(t), \psi(t)) \to (\rho(t_1), \rho(t_2))$ is in Δ.

The automaton $\mathcal{A}$ *accepts* (T, ψ) if there is a run over (T, ψ) with $\rho(t^*) = s_0$, where t^* denotes the root of T. The N-slice $\mathcal{L}_N(\mathcal{A})$ is the set of pairs $(T, \psi) \in \mathrm{Trees}_2[\Sigma]$ with $|V(T)| = N$ that are accepted by $\mathcal{A}$.

Given a finite alphabet Σ, Arenas et al. [5] consider the following computational problem.
Name: #TA
Input: A tree automaton $\mathcal{A}$ and an integer N in unary.
Output: $|\mathcal{L}_N(\mathcal{A})|$.

Their Corollary 4.9 gives us the following lemma.

Lemma 51. *(Arenas, Croquevielle, Jayaram, Riveros) There is an FPRAS for* #TA.

5.2.4 The proof of Theorem 16. Theorem 16 now follows from Lemma 51 and from Lemma 52, whose proof is inspired by the reduction of Arenas et al. [5] for the case with bounded hypertreewidth.

Lemma 52. *Let b be a positive integer. Let C be a class of hypergraphs such that every member of C has fractional hypertreewidth at most b. There is a parsimonious reduction from* #CQ(Φ_C) *to* #TA.

Proof. Let $(\varphi, \mathcal{D})$ be an input to #CQ(Φ_C). Let $n = \|\varphi\|$ and $m = \|\mathcal{D}\|$.

By Lemma 43 it takes $\mathrm{poly}(n, m)$ time to construct a nice tree decomposition $(T, \mathbf{B})$ of $H(\varphi)$ with fractional hypertreewidth at most $c := 7b^3 + 31b + 7$

For each $t \in V(T)$, let $\mathrm{Sol}_t = \mathrm{Sol}(\varphi, \mathcal{D}, B_t)$. By Lemma 48 this can be computed in $\mathrm{poly}(n, m)$ time. Let $\mathrm{Sol}'_t = \mathrm{proj}(\mathrm{Sol}_t, \mathrm{free}(\varphi))$. Clearly, this can also be constructed in $\mathrm{poly}(n, m)$ time.

Recall that t^* is the root of T. Note that $\mathrm{Sol}_{t^*} = \mathrm{Sol}(\varphi, \mathcal{D}, \emptyset)$. If this is empty, then there are no answers of (φ, D), so the algorithm returns 0. We assume from now on that Sol_{t^*} is non-empty. In this case, it contains exactly one assignment, which is the empty assignment ε.

We now use $(T, \mathbf{B})$ and $\{\mathrm{Sol}_t\}$ and $\{\mathrm{Sol}'_t\}$ to construct a tree automaton $\mathcal{A} = (S, \Sigma, \Delta, s_0)$.

- The state space S is defined by $S = \{(t, \alpha) \mid t \in V(T), \alpha \in \mathrm{Sol}_t\}$.
- The initial state is $s_0 = (t^*, \varepsilon)$.
- The label set Σ is defined by $\Sigma = \{(t, \beta) \mid t \in V(T), \beta \in \mathrm{Sol}'_t\}$.
- Suppose that $t \in V(T)$ has two children t_1 and t_2. Recall that $B_t = B_{t_1} = B_{t_2}$. Thus, $\mathrm{Sol}_t = \mathrm{Sol}_{t_1} = \mathrm{Sol}_{t_2}$. For each $\alpha \in \mathrm{Sol}_t$ there is a transition $((t, \alpha), (t, \mathrm{proj}(\alpha, \mathrm{free}(\varphi)))) \to ((t_1, \alpha), (t_2, \alpha))$.
- Suppose that t has one child t_1 and $B_{t_1} \subseteq B_t$ and $B_t \setminus B_{t_1} = \{v\}$. For each $\alpha \in \mathrm{Sol}_t$ we have $\mathrm{proj}(\alpha, B_{t_1}) \in \mathrm{Sol}_{t_1}$ and there is a transition $((t, \alpha), (t, \mathrm{proj}(\alpha, \mathrm{free}(\varphi)))) \to (t_1, \mathrm{proj}(\alpha, B_{t_1}))$.

- Suppose that t has one child t_1 and $B_t \subseteq B_{t_1}$ and $B_{t_1} \setminus B_t = \{v\}$. For each $\alpha \in \mathsf{Sol}_t$ let $A_\alpha = \{\alpha_1 \in \mathsf{Sol}_{t_1} \mid \alpha_1$ is consistent with $\alpha\}$. For each $\alpha_1 \in A_\alpha$, there is a transition $((t,\alpha),(t,\mathrm{proj}(\alpha,\mathrm{free}\,(\varphi)))) \longrightarrow (t_1,\alpha_1)$.
- Suppose that t is a leaf so that $B_t = \emptyset$. For the empty assignment ε, there is a transition $((t,\varepsilon),(t,\varepsilon)) \longrightarrow \emptyset$.

Let $N = |V(T)|$. The bijection from $\mathcal{L}_N(\mathcal{A})$ to $\mathsf{Ans}(\varphi,\mathcal{D})$ is described as follows.

- Consider any element (T',ψ) of $\mathcal{L}_N(\mathcal{A})$ and any run ρ that accepts (T',ψ). It is immediate from the construction of $\mathcal{A}$ (and the definition of accept and run) that $T' = T$.
  It also follows from the construction of $\mathcal{A}$ that the assignments α in the states (t,α) of ρ are all consistent. This is from the construction of the transitions, together with item (ii) in the definition of tree decomposition, which ensures that, for each variable $v \in \mathrm{var}(\varphi)$, the set $\{t \in V(T) \mid v \in B_t\}$ is connected in T.
  Every $v \in \mathrm{var}(\varphi)$ appears in some set B_t (this follows from our assumption in the definition of CQs that every v is in at least one atom of φ, together with item (i) in the definition of tree decomposition). Thus, composing all of the α's in a run gives an assignment $\tau \colon \mathrm{var}(\varphi) \to U(\mathcal{D})$. Consider any atom R of φ. By item (i), the variables of R are all contained in some bag B_t. Let $\alpha = \mathrm{proj}(\tau, B_t)$. Since (t,α) is a state, $\alpha \in \mathsf{Sol}_t$, so α satisfies R and so does τ. We conclude that τ is a solution of $(\varphi,\mathcal{D})$.
  Therefore, $\mathrm{proj}(\tau,\mathrm{free}\,(\varphi))$, which is equal to ψ (the composition of the labels in the run ρ) is an answer of $(\varphi,\mathcal{D})$.
- On the other hand, for any answer ψ of $(\varphi,\mathcal{D})$ there is a consistent assignment $\tau \colon \mathrm{vars}\,(\varphi) \to U(\mathcal{D})$ that satisfies every atom. τ is a solution of $(\varphi,\mathcal{D})$. The run which assigns each $t \in V(T)$ state $(t,\mathrm{proj}(\tau,B_t))$ is a run accepting (T,ψ).

□

6 EXTENSIONS

All of our algorithmic results can be extended from approximately counting answers to approximately uniformly sampling them. For the classes of hypergraphs C for which we present algorithms for #ECQ (Φ_C) it is not hard to see that #ECQ (Φ_C) is self-partitionable (see [19]), and thus the technique from Jerrum, Valiant and Vazirani [30] shows that approximate sampling and approximate counting are equivalent.

Alternatively, all of our algorithms can be adjusted specifically to solve the corresponding sampling problem.

- A key ingredient of the algorithms presented in the proofs of Theorems 5 and 13 is the framework of Dell, Lapinskas, and Meeks [15, Theorem 1] (stated in Theorem 17). In their Theorem 2, Dell, Lapinskas, and Meeks also provide a framework for approximate sampling and this can be used in our algorithms instead of their Theorem 1.
- A key ingredient of the algorithm presented in the proof of Theorem 16 is the tree automaton algorithm of Arenas, Croquevielle, Jayaram and Riveros [5] (stated as Lemma 51 in this work). In their Corollary 4.9, Arenas et al. also provide a fully-polynomial almost uniform sampler for the set of pairs that are accepted by a tree automaton and this can be used in our algorithms instead of Lemma 51.

Using a standard technique for approximate counting from Karp and Luby [31], our results can also be extended to counting answers of unions of (extended) conjunctive queries. This technique has been used before in other works about fixed-parameter tractability [6].

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases.* Addison-Wesley. http://webdam.inria.fr/Alice/

[2] Isolde Adler. 2006. *Width functions for hypertree decompositions.* Ph. D. Dissertation. Albert-Ludwigs-Universität Freiburg. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.2257&rep=rep1&type=pdf

[3] Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and Süleyman Cenk Sahinalp. 2008. Biomolecular network motif counting and discovery by color coding. In *Proceedings 16th International Conference on Intelligent Systems for Molecular Biology (ISMB), Toronto, Canada, July 19-23, 2008.* 241–249. https://doi.org/10.1093/bioinformatics/btn163

[4] Marcelo Arenas, Pablo Barceló, and Juan L. Reutter. 2011. Query Languages for Data Exchange: Beyond Unions of Conjunctive Queries. *Theory Comput. Syst.* 49, 2 (2011), 489–564. https://doi.org/10.1007/s00224-010-9259-6

[5] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. 2021. When is approximate counting for conjunctive queries tractable?. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, Samir Khuller and Virginia Vassilevska Williams (Eds.). ACM, 1015–1027. https://doi.org/10.1145/3406325.3451014

[6] Vikraman Arvind and Venkatesh Raman. 2002. Approximation Algorithms for Some Parameterized Counting Problems. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2518)*, Prosenjit Bose and Pat Morin (Eds.). Springer, 453–464. https://doi.org/10.1007/3-540-36136-7_40

[7] Stefan Bard, Thomas Bellitto, Christopher Duffy, Gary MacGillivray, and Feiran Yang. 2018. Complexity of locally-injective homomorphisms to tournaments. *Discret. Math. Theor. Comput. Sci.* 20, 2 (2018). http://dmtcs.episciences.org/4999

[8] Andrei A. Bulatov and Stanislav Živný. 2020. Approximate Counting CSP Seen from the Other Side. *ACM Trans. Comput. Theory* 12, 2 (2020), 11:1–11:19. https://doi.org/10.1145/3389390

[9] Ashok K. Chandra and Philip M. Merlin. 1977. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison (Eds.). ACM, 77–90. https://doi.org/10.1145/800105.803397

[10] Hubie Chen and Stefan Mengel. 2016. Counting Answers to Existential Positive Queries: A Complexity Classification. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, Tova Milo and Wang-Chiew Tan (Eds.). ACM, 315–326. https://doi.org/10.1145/2902251.2902279

[11] Gianluca Cima, Maurizio Lenzerini, and Antonella Poggi. 2020. Answering Conjunctive Queries with Inequalities in *DL-Lite*$_{\mathcal{R}}$. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020.* AAAI Press, 2782–2789. https://aaai.org/ojs/index.php/AAAI/article/view/5666

[12] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms* (1st ed.). Springer Publishing Company, Incorporated.

[13] Víctor Dalmau and Peter Jonsson. 2004. The complexity of counting homomorphisms seen from the other side. *Theor. Comput. Sci.* 329, 1-3 (2004), 315–323. https://doi.org/10.1016/j.tcs.2004.08.008

[14] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. 2002. Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2470)*, Pascal Van Hentenryck (Ed.). Springer, 310–326. https://doi.org/10.1007/3-540-46135-3_21

[15] Holger Dell, John Lapinskas, and Kitty Meeks. 2020. Approximately counting and sampling small witnesses using a colourful decision oracle. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, Shuchi Chawla (Ed.). SIAM, 2201–2211. https://doi.org/10.1137/1.9781611975994.135

[16] Holger Dell, Marc Roth, and Philip Wellnitz. 2019. Counting Answers to Existential Questions. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 113:1–113:15. https://doi.org/10.4230/LIPIcs.ICALP.2019.113

[17] Rodney G. Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity.* Springer. https://doi.org/10.1007/978-1-4471-5559-1

[18] Arnaud Durand and Stefan Mengel. 2015. Structural Tractability of Counting of Solutions to Conjunctive Queries. *Theory Comput. Syst.* 57, 4 (2015), 1202–1249. https://doi.org/10.1007/s00224-014-9543-y

[19] Martin Dyer and Catherine Greenhill. 1999. Random walks on combinatorial objects. *London Mathematical Society Lecture Note Series* (1999), 101–136.

[20] Martin E. Dyer, Leslie Ann Goldberg, Catherine S. Greenhill, and Mark Jerrum. 2004. The Relative Complexity of Approximate Counting Problems. *Algorithmica* 38, 3 (2004), 471–500. https://doi.org/10.1007/s00453-003-1073-y

[21] Jiří Fiala, Ton Kloks, and Jan Kratochvíl. 2001. Fixed-parameter complexity of lambda-labelings. *Discret. Appl. Math.* 113, 1 (2001), 59–72. https://doi.org/10.1016/S0166-218X(00)00387-5

[22] Jiří Fiala and Jan Kratochvíl. 2008. Locally constrained graph homomorphisms - structure, complexity, and applications. *Comput. Sci. Rev.* 2, 2 (2008), 97–111. https://doi.org/10.1016/j.cosrev.2008.06.001

[23] Jacob Focke, Leslie Ann Goldberg, Marc Roth, and Stanislav Zivný. 2022. Approximately Counting Answers to Conjunctive Queries with Disequalities and Negations. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Leonid Libkin and Pablo Barceló (Eds.). ACM, 315–324. https://doi.org/10.1145/3517804.3526231

[24] Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2001. The complexity of acyclic conjunctive queries. *J. ACM* 48, 3 (2001), 431–498. https://doi.org/10.1145/382780.382783

[25] Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2002. Hypertree decomposition and tractable queries. *J. Comput. System Sci.* 64, 3 (2002), 579–627. https://doi.org/10.1006/jcss.2001.1809

[26] Martin Grohe. 2007. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM* 54, 1 (2007), 1:1–1:24. https://doi.org/10.1145/1206035.1206036

[27] Martin Grohe and Dániel Marx. 2014. Constraint Solving via Fractional Edge Covers. *ACM Transactions on Algorithms* 11, 1 (2014), 4:1–4:20. https://doi.org/10.1145/2636918

[28] Víctor Gutiérrez-Basulto, Yazmín Angélica Ibáñez-García, Roman Kontchakov, and Egor V. Kostylev. 2015. Queries with negation and inequalities over lightweight ontologies. *J. Web Semant.* 35 (2015), 184–202. https://doi.org/10.1016/j.websem.2015.06.002

[29] Russell Impagliazzo and Ramamohan Paturi. 2001. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62, 2 (2001), 367–375. https://doi.org/10.1006/jcss.2000.1727

[30] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. 1986. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theor. Comput. Sci.* 43 (1986), 169–188. https://doi.org/10.1016/0304-3975(86)90174-X

[31] Richard M. Karp and Michael Luby. 1983. Monte-Carlo Algorithms for Enumeration and Reliability Problems. In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*. IEEE Computer Society, 56–64. https://doi.org/10.1109/SFCS.1983.35

[32] Paraschos Koutris, Tova Milo, Sudeepa Roy, and Dan Suciu. 2017. Answering Conjunctive Queries with Inequalities. *Theory Comput. Syst.* 61, 1 (2017), 2–30. https://doi.org/10.1007/s00224-016-9684-2

[33] Paraschos Koutris and Jef Wijsen. 2018. Consistent Query Answering for Primary Keys and Conjunctive Queries with Negated Atoms. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, Jan Van den Bussche and Marcelo Arenas (Eds.). ACM, 209–224. https://doi.org/10.1145/3196959.3196982

[34] Dániel Marx. 2010. Approximating fractional hypertree width. *ACM Trans. Algorithms* 6, 2 (2010), 29:1–29:17. https://doi.org/10.1145/1721837.1721845

[35] Dániel Marx. 2010. Can You Beat Treewidth? *Theory Comput.* 6, 1 (2010), 85–112. https://doi.org/10.4086/toc.2010.v006a005

[36] Dániel Marx. 2011. Tractable Structures for Constraint Satisfaction with Truth Tables. *Theory of Computing Systems* 48, 3 (2011), 444–464. https://doi.org/10.1007/s00224-009-9248-9

[37] Dániel Marx. 2013. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM* 60, 6 (2013). https://doi.org/10.1145/2535926 Article No. 42.

[38] Kitty Meeks. 2016. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discret. Appl. Math.* 198 (2016), 170–194. https://doi.org/10.1016/j.dam.2015.06.019

[39] Christos H. Papadimitriou and Mihalis Yannakakis. 1999. On the Complexity of Database Queries. *J. Comput. Syst. Sci.* 58, 3 (1999), 407–427. https://doi.org/10.1006/jcss.1999.1626

[40] Reinhard Pichler and Sebastian Skritek. 2013. Tractable counting of the answers to conjunctive queries. *J. Comput. Syst. Sci.* 79, 6 (2013), 984–1001. https://doi.org/10.1016/j.jcss.2013.01.012

[41] Neil Robertson and Paul D. Seymour. 1984. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B* 36, 1 (1984), 49–64. https://doi.org/10.1016/0095-8956(84)90013-3

[42] Marc Roth. 2021. Paramaterized Counting of Partially Injective Homomorphisms. *Algorithmica* (2021). https://doi.org/10.1007/s00453-021-00805-y

[43] Paweł Rzazewski. 2014. Exact algorithm for graph homomorphism and locally injective graph homomorphism. *Inf. Process. Lett.* 114, 7 (2014), 387–391. https://doi.org/10.1016/j.ipl.2014.02.012

[44] Claus-Peter Schnorr. 1976. Optimal Algorithms for Self-Reducible Problems. In *Third International Colloquium on Automata, Languages and Programming, University of Edinburgh, UK, July 20-23, 1976*, S. Michaelson and Robin Milner (Eds.). Edinburgh University Press, 322–337.

[45] Mihalis Yannakakis. 1981. Algorithms for Acyclic Database Schemes. In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*. IEEE Computer Society, 82–94.