Answer Counting under Guarded TGDs

Cristina Feier
Department of Computer Science, University of Bremen, Germany

Carsten Lutz
Department of Computer Science, University of Bremen, Germany

Marcin Przybyłko
Department of Computer Science, University of Bremen, Germany

Abstract

We study the complexity of answer counting for ontology-mediated queries and for querying under constraints, considering conjunctive queries and unions thereof (UCQs) as the query language and guarded TGDs as the ontology and constraint language, respectively. Our main result is a classification according to whether answer counting is fixed-parameter tractable (FPT), W[1]-equivalent, #W[1]-equivalent, #W[2]-hard, or #A[2]-equivalent, lifting a recent classification for UCQs without ontologies and constraints due to Dell et al. [19]. The classification pertains to various structural measures, namely treewidth, contract treewidth, starsize, and linked matching number. Our results rest on the assumption that the arity of relation symbols is bounded by a constant and, in the case of ontology-mediated querying, that all symbols from the ontology and query can occur in the data (so-called full data schema). We also study the meta-problems for the mentioned structural measures, that is, to decide whether a given ontology-mediated query or constraint-query specification is equivalent to one for which the structural measure is bounded.

2012 ACM Subject Classification Theory of computation → Database theory

Keywords and phrases Ontology-Mediated Querying, Querying under Constraints, Answer Counting, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.ICDT.2021.9

Related Version A full version of the paper is available at [23], at `https://arxiv.org/abs/2101.03058`.

Funding This research was funded by ERC consolidator grant 647289 CODA and by DFG project QTEC.

Acknowledgements We thank the anonymous reviewers for useful comments.

1 Introduction

Tuple-generating dependencies (TGDs) are a prominent formalism for formulating database constraints. A TGD states that if certain facts are true, then certain other facts must be true as well. This can be interpreted in different ways. In *ontology-mediated querying*, TGDs give rise to ontology languages and are used to derive new facts in addition to those that are present in the database. This makes it possible to obtain additional answers if the data is incomplete and also enriches the vocabulary that is available for querying. In a more classical setup that we refer to as *querying under constraints*, TGDs are used as integrity constraints on the database, that is, a TGD expresses the promise that if certain facts are present in the database, then certain other facts are present as well. Integrity constraints are relevant to query optimization as they might enable the reformulation of a query into a 'simpler' one. TGDs generalize a wide range of other integrity constraints, which was the original reason for introducing them [1].

When unrestricted TGDs are used as an ontology language, ontology-mediated querying is undecidable even for unary queries that consist of a single atom [12]. This has led to intense research on identifying restricted forms of TGDs that regain decidability, see [3, 12, 13, 30] and references therein. In this paper, we consider guardedness as a basic and robust such restriction: a TGD is guarded if some body atom, the guard, contains all body variables [12]. Guarded TGDs are useful also for formalizing integrity constraints. For example, the important class of referential integrity constraints (also known as inclusion dependencies) is a special case of guarded TGDs.

While being decidable, both ontology-mediated querying and querying under constraints with guarded TGDs is computationally intractable. Let us make this precise for query evaluation, i.e. the problem to decide, given a database, a query, and a candidate answer, whether the candidate is indeed an answer. We use $(\mathbb{G}, \mathbb{CQ})$ to denote the language of ontology-mediated queries $(\mathcal{O}, \mathbf{S}, q)$ that consist of an ontology $\mathcal{O}$ which is a set of guarded TGDs, a data schema $\mathbf{S}$, and a conjunctive query (CQ) q. As usual, $\mathbf{S}$ contains the relation names that can be used in the data while both the ontology and query can also use additional names. Evaluating ontology-mediated queries (OMQs) from $(\mathbb{G}, \mathbb{CQ})$ is 2ExpTime-complete in combined complexity. The same holds for $(\mathbb{G}, \mathbb{UCQ})$ where the queries are unions of CQs (UCQs) [12]. For querying under constraints, we consider constraint query specifications (CQSs) of the form $(\mathcal{T}, \mathbf{S}, q)$ where $\mathcal{T}$ is a set of integrity constraints and q is a query, both over schema $\mathbf{S}$. Overloading notation, we use $(\mathbb{G}, (\mathbb{U})\mathbb{CQ})$ also to denote the class of CQSs in which the constraints are guarded TGDs and the queries are (U)CQs; it will always be clear from the context whether $(\mathbb{G}, (\mathbb{U})\mathbb{CQ})$ denotes an OMQ language or a class of CQSs. Query evaluation for CQSs from $(\mathbb{G}, \mathbb{CQ})$ and $(\mathbb{G}, \mathbb{UCQ})$ is NP-complete.

In this paper, we are interested in counting the number of answers to OMQs and to queries posed under integrity constraints, with an emphasis on the limits of efficiency from the viewpoint of parameterized complexity theory. Counting the number of answers is important to inform the user when there are too many answers to compute all of them, and it is supported by almost every data management system. It is also a fundamental operation in data analytics and in decision support where often the count is more important than the actual answers. Despite its relevance, however, the problem has received little attention in ontology-mediated querying and querying under constraints, see [29, 28, 9, 14] for some notable exceptions.

We equate efficiency with fixed-parameter tractability (FPT), the parameter being the size of the OMQ and of the CQS, respectively. Evaluating Boolean queries is W[1]-hard both for ontology-mediated querying in $(\mathbb{G}, (\mathbb{U})\mathbb{CQ})$ and for querying under constraints in $(\mathbb{G}, (\mathbb{U})\mathbb{CQ})$ [5], and therefore answer counting (which is the same problem as query evaluation for Boolean queries) is in general not fixed-parameter tractable unless FPT = W[1]. The main question that we ask is: how can we characterize the parameterized complexity of answer counting for classes of OMQs or CQSs $\mathbb{C} \subseteq (\mathbb{G}, (\mathbb{U})\mathbb{CQ})$ and, most importantly, for which such classes $\mathbb{C}$ can we count answers in FPT? The classes $\mathbb{C}$ will primarily be defined in terms of structural restrictions of the (U)CQ, but will also take into account the interplay between the ontology/constraints and the (U)CQ. Note that PTime combined complexity, a (significant) strengthening of FPT, cannot be obtained by structural restrictions on the UCQ in ontology-mediated querying with $(\mathbb{G}, (\mathbb{U})\mathbb{CQ})$ because evaluating Boolean OMQs is 2ExpTime-complete already for unary single atom queries. For querying under constraints, in contrast, PTime combined complexity is not out of reach and in the case of query evaluation can in fact sometimes be attained in $(\mathbb{G}, (\mathbb{U})\mathbb{CQ})$ [8, 7].

A seminal result due to Grohe states that a recursively enumerable class $\mathbb{C}$ of CQs can be evaluated in FPT if and only if there is a constant that bounds the treewidths of CQs in $\mathbb{C}$, modulo equivalence [26]. Here, treewidth of a CQ q means the treewidth of the Gaifman graph of q after dropping all answer variables. The result rests on the assumptions that FPT $\neq$ W[1] and that the arity of relation symbols is bounded by a constant, which we shall also assume throughout this article. Grohe's result extends to UCQs in the expected way, that is, the characterization for UCQs is in terms of the maximum treewidth of the constituting CQs modulo equivalence, assuming w.l.o.g. that there are no containment relations among them. An adaptation of Grohe's proof was used by Dalmau and Jonsson to show that a class $\mathbb{C}$ of CQs without quantified variables admits answer counting in FPT if and only if the treewidths of CQs in $\mathbb{C}$ is bounded by a constant [18]. In a series of papers by Pichler and Skritek [32], Durand and Mengel [20, 21], Chen and Mengel [16, 17], and Dell et al. [19], this was extended to a rather detailed classification of the parameterized complexity of answer counting for classes of CQs and UCQs that may contain both answer variables and quantified variables. The characterization uses treewidth, which now refers to the entire Gaifman graph including the answer variables. It also refers to the additional structural measures of contract treewidth, starsize,[1] and linked matching number. It links boundedness of these measures by a constant, modulo equivalence, to the relevant complexities, which turn out to be FPT, W[1]-equivalence, #W[1]-equivalence, #W[2]-hardness, and #A[2]-equivalence. Here, we speak of 'equivalence' rather than of 'completeness' to emphasize that hardness is defined in terms of (parameterized counting) Turing (fpt-)reductions.

The main results of this article are classifications of the complexity of answer counting for classes of ontology-mediated queries from $(\mathbb{G}, (\mathbb{U})\mathbb{CQ})$, assuming that the data schema contains all symbols used in the ontology and query, and for classes of constraint query specifications from $(\mathbb{G}, (\mathbb{U})\mathbb{CQ})$. Our classifications parallel the one for the case without TGDs, involve the same five complexities mentioned above, and link them to the same structural measures. However, there is a twist. The ontology interacts with all of the mentioned structural measures in the sense that for each measure, there is a class of CQs $\mathbb{C}$ and an ontology $\mathcal{O}$ such that the measure is unbounded for $\mathbb{C}$ modulo equivalence while there is a constant k such that each OMQ $(\mathcal{O}, \mathbf{S}, q)$, $q \in \mathbb{C}$, is equivalent to an OMQ $(\mathcal{O}, \mathbf{S}, q')$ with the measure of q' bounded by k. A similar effect can be observed for querying under constraints. We can thus not expect to link the complexity of a class $\mathbb{C}$ of OMQs to the structural measures of the actual queries in the OMQs. Instead, we consider a certain class of CQs that we obtain from the OMQs in $\mathbb{C}$ by first rewriting away the existential quantifiers in TGD heads in the ontology, then taking the CQs that occur in the resulting OMQs, combining them conjunctively guided by the inclusion-exclusion principle, next chasing them with the ontology (which is a finite operation due to the first step), and then taking the homomorphism core. The structural measures of the resulting class of CQs turn out to determine the complexity of answer counting for the original class of OMQs $\mathbb{C}$. Interestingly, the same is also true for classes of constraint query specifications and thus the characterizations for OMQs and for CQSs coincide. We in fact establish the latter by mutual reduction between answer counting for OMQs and answer counting for CQSs.

Inspired by our complexity classifications, we also study the meta problems to decide whether a given query is equivalent to a query in which some selected structural measures are small, and to construct the latter query if it exists. We do this both for ontology-mediated queries and for queries under constraints, considering all four measures that are featured in the classifications (and sets thereof). We start with querying under constraints where

[1] The measure is called dominating starsize in [19] and strict starsize in [16]. We only speak of starsize. Note that this is not identical to the original notion of starsize from [20, 21].

we are able to obtain decidability results in all relevant cases. These results can also be applied to ontology-mediated querying when (i) the data schema contains all symbols used in the ontology and query and (ii) we require that the ontology used in the OMQ cannot be replaced with a different one. For contract treewidth and starsize, we additionally show that it is never necessary to modify the ontology to attain equivalent OMQs with small measures, and we provide decidability results without assumptions (i) and (ii). We also observe that treewidth behaves differently in that modifying the ontology might result in smaller measures. Deciding the meta problems for the measure of treewidth is left open as an interesting and non-trivial open problem.

Some proofs are deferred to the appendix of the long version of this paper [23], available at `https://arxiv.org/abs/2101.03058`.

Related Work. The complexity of ontology-mediated querying has been a subject of intense study from various angles, see for example [10, 11, 33] and references therein. The parameterized complexity of evaluating ontology-mediated queries has been studied in [6, 5]. In [5], it is shown that query evaluation in FPT coincides with bounded treewidth modulo equivalence in $(\mathbb{G}, \mathbb{UCQ})$ when the arity of relation symbols is bounded by a constant. An FPT upper bound for querying under constraints that are guarded TGDs has been established in [8, 7] for CQs that have bounded generalized hypertreewidth modulo equivalence. That paper also studies the meta problems for querying under constraints that are guarded TGDs and for the measure of generalized hypertree width. A related topic is query containment under constraints, see for example [15, 27, 24].

2 Preliminaries

For an integer $n \geq 1$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. To indicate the cardinality of a set S, we may write $\#S$ or $|S|$.

Relational Databases. A *schema* $\mathbf{S}$ is a set of relation symbols R with associated arity $\mathsf{ar}(R) \geq 0$. We write $\mathsf{ar}(\mathbf{S})$ for $\max_{R \in \mathbf{S}}\{\mathsf{ar}(R)\}$. An $\mathbf{S}$-*fact* is an expression of the form $R(\bar{c})$, where $R \in \mathbf{S}$ and $\bar{c}$ is an $\mathsf{ar}(R)$-tuple of constants. An $\mathbf{S}$-*instance* is a (possibly infinite) set of $\mathbf{S}$-facts and an $\mathbf{S}$-*database* is a finite $\mathbf{S}$-instance. We write $\mathsf{adom}(I)$ for the set of constants in an instance I. For a set $S \subseteq \mathsf{adom}(I)$, we denote by $I_{|S}$ the restriction of I to facts that mention only constants from S. A *homomorphism* from I to an instance J is a function $h : \mathsf{adom}(I) \to \mathsf{adom}(J)$ such that $R(h(\bar{c})) \in J$ for every $R(\bar{c}) \in I$. A database D' is obtained from a database D by *cloning constants* if $D' \supseteq D$ can be constructed by choosing $a_1, \ldots, a_n \in \mathsf{adom}(D)$ and positive integers $m_1, \ldots, m_n$, reserving fresh constants $a_1^{i_1}, \ldots, a_n^{i_n}$, $1 \leq i_\ell \leq m_\ell$ for $1 \leq \ell \leq n$, and adding to D each atom $R(\bar{a}')$ that can be obtained from some $R(\bar{a}) \in D$ by replacing each occurrence of a_i, $1 \leq i \leq n$, with a_i^j for some j with $1 \leq j \leq m_i$.

CQs and UCQs. A *conjunctive query* (CQ) $q(\bar{x})$ over a schema $\mathbf{S}$ is a first-order formula of the form $\exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ where φ is a conjunction of *relational atoms* $R_i(\bar{x}_i)$ with $R_i \in \mathbf{S}$ and $\bar{x}_i$ a tuple of variables of length $\mathsf{ar}(R_i)$ and *equality atoms* $x_1 = x_2$. We require that only variables from $\bar{x}$ appear in equality atoms. With $\mathsf{var}(q)$, we denote the set of variables that occur in $\bar{x}$ or in $\bar{y}$. Whenever convenient, we identify a conjunction of atoms with a set of atoms. When we are not interested in order and multiplicity, we treat $\bar{x}$ as a set of variables. We write $\mathbb{CQ}$ for the class of CQs.

Every CQ q can be naturally seen as a database D_q, known as the *canonical database* of q, obtained by dropping the existential quantifier prefix and the equality atoms, and viewing variables as constants. A *homomorphism* h from a CQ q to an instance I is a homomorphism from D_q to I such that $x = y \in q$ implies $h(x) = h(y)$. A tuple $\bar{c} \in \mathsf{adom}(I)^{|\bar{x}|}$ is an *answer* to Q on I if there is a homomorphism h from q to I with $h(\bar{x}) = \bar{c}$. The *evaluation of* $q(\bar{x})$ *on* I, denoted $q(I)$, is the set of all answers to Q on I.

A *union of conjunctive queries* (UCQ) over a schema $\mathbf{S}$ is a first-order formula of the form $q(\bar{x}) := q_1(\bar{x}) \vee \cdots \vee q_n(\bar{x})$, where $n \geq 1$, and $q_1(\bar{x}), \ldots, q_n(\bar{x})$ are CQ over $\mathbf{S}$. We refer to the variables in $\bar{x}$ as the *answer variables* of q and the *arity* of q is defined as the number of its answer variables. The *evaluation* of q on an instance I, denoted $q(I)$, is the set of tuples $\bigcup_{i \in [n]} q_i(I)$. We write $\mathbb{UCQ}$ for the class of UCQs. A (U)CQ of arity zero is called *Boolean*. The only possible answer to a Boolean query is the empty tuple. For a Boolean (U)CQ q, we may write $I \models q$ if $q(I) = \{()\}$ and $I \not\models q$ otherwise.

Let $q_1(\bar{x})$ and $q_2(\bar{x})$ be two UCQs over the same schema $\mathbf{S}$. We say that q_1 is *contained* in q_2, written $q_1 \subseteq_{\mathbf{S}} q_2$, if $q_1(D) \subseteq q_2(D)$ for every $\mathbf{S}$-database D. Moreover, q_1 and q_2 are *equivalent*, written $q_1 \equiv_{\mathbf{S}} q_2$, if $q_1 \subseteq_{\mathbf{S}} q_2$ and $q_2 \subseteq_{\mathbf{S}} q_1$.

A CQ $q(\bar{x})$ is a *core* if every homomorphism h from q to D_q with $h(\bar{x}) = \bar{x}$ is surjective. Every CQ $q(\bar{x})$ is equivalent to a CQ $p(\bar{x})$ that is a core and can be obtained from q by dropping atoms. In fact, p is unique up to isomorphism and we call it the *core of* q. For a UCQ q, we use $\mathsf{core}(q)$ to denote the disjunction whose disjuncts are the cores of the CQs in q.

For a UCQ q, but also for any other syntactic object q, we use $||q||$ to denote the number of symbols needed to write q as a word over a suitable alphabet.

Our main interest is in the complexity of counting the number of answers. Every choice of a query language $\mathbb{Q}$, such as $\mathbb{CQ}$ and $\mathbb{UCQ}$, and a class of databases $\mathbb{D}$ gives rise to the following answer counting problem:

PROBLEM :	$\mathsf{AnswerCount}(\mathbb{Q}, \mathbb{D})$
INPUT :	A query $q \in \mathbb{Q}$ over some schema $\mathbf{S}$ and an $\mathbf{S}$-database $D \in \mathbb{D}$
OUTPUT :	$\#q(D)$

Our main interest is in the parameterized version of the above problem where we generally assume that the parameter is the size of the input query, see below for more details. When $\mathbb{D}$ is the class of all databases, we simply write $\mathsf{AnswerCount}(\mathbb{Q})$.

Treewidth. Treewidth is a widely used notion that measures the degree of tree-likeness of a graph. Let $G = (V, E)$ be an undirected graph. A *tree decomposition* of G is a pair $\delta = (T_\delta, \chi)$, where $T_\delta = (V_\delta, E_\delta)$ is a tree, and χ is a labeling function $V_\delta \to 2^V$, i.e., χ assigns a subset of V to each node of T_δ, such that:

1. $\bigcup_{t \in V_\delta} \chi(t) = V$,
2. if $\{u, v\} \in E$, then $u, v \in \chi(t)$ for some $t \in V_\delta$,
3. for each $v \in V$, the set of nodes $\{t \in V_\delta \mid v \in \chi(t)\}$ induces a connected subtree of T_δ.

The *width* of δ is the number $\max_{t \in V_\delta}\{|\chi(t)|\} - 1$. If the edge set E of G is non-empty, then the *treewidth* of G is the minimum width over all its tree decompositions; otherwise, it is defined to be one. Each instance I is associated with an undirected graph (without self loops) $G_I = (V, E)$, called the *Gaifman graph* of I, defined as follows: $V = \mathsf{adom}(I)$, and $\{a, b\} \in E$ iff there is a fact $R(\bar{c}) \in I$ that mentions both a and b. The *treewidth* of I is the treewidth of G_I.

TGDs, Guardedness. A *tuple-generating dependency* (TGD) T over $\mathbf{S}$ is a first-order sentence of the form $\forall \bar{x} \forall \bar{y} \big(\phi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})\big)$ such that $\exists \bar{y}\, \phi(\bar{x}, \bar{y})$ and $\exists \bar{z}\, \psi(\bar{x}, \bar{z})$ are CQs without equality atoms. As a special case, we also allow $\phi(\bar{x}, \bar{y})$ to be the empty conjunction, i.e. logical truth, denoted by true. For simplicity, we write T as $\phi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$. We call

ϕ and ψ the *body* and *head* of T, denoted $\mathsf{body}(T)$ and $\mathsf{head}(T)$, respectively. An instance I over $\mathbf{S}$ *satisfies* T, denoted $I \models T$, if $q_\phi(I) \subseteq q_\psi(I)$. It *satisfies* a set of TGDs S, denoted $I \models S$, if $I \models T$ for each $T \in S$. We then also say that I is a *model* of S. We write $\mathbb{TGD}$ to denote the class of all TGDs.

A TGD T is *guarded* if $\mathsf{body}(T)$ is true or there exists an atom α in its body that contains all variables that occur in $\mathsf{body}(T)$ [12]. Such an atom α is the *guard* of T, denoted $\mathsf{guard}(T)$. We write $\mathbb{G}$ for the class of guarded TGDs. A TGD T is *full* if the tuple $\bar{z}$ of variables is empty. We use $\mathbb{FULL}$ to denote the class of full TGDs and shall often refer to $\mathbb{G} \cap \mathbb{FULL}$, the class of TGDs that are both guarded and full. Note that this class is essentially the class of Datalog programs with guarded rule bodies.

We next introduce the well-known chase procedure for making explicit the consequences of a set of TGDs [31, 27, 22, 12]. We first define a single chase step. Let I be an instance over a schema $\mathbf{S}$ and $T = \phi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ a TGD over $\mathbf{S}$. We say that T is *applicable* to a tuple $(\bar{c}, \bar{c}')$ of constants in I if $\phi(\bar{c}, \bar{c}') \subseteq I$. In this case, *the result of applying T in I at* $(\bar{c}, \bar{c}')$ is the instance $J = I \cup \psi(\bar{c}, \bar{c}'')$, where $\bar{c}''$ is the tuple obtained from $\bar{z}$ by simultaneously replacing each variable z with a fresh distinct constant that does not occur in I. We describe such a single chase step by writing $I \xrightarrow{T,\, (\bar{c}, \bar{c}')} J$. Let I be an instance and S a finite set of TGDs. A *chase sequence for I with S* is a sequence of chase steps

$$I_0 \xrightarrow{T_0,\, (\bar{c}_0, \bar{c}_0')} I_1 \xrightarrow{T_1,\, (\bar{c}_1, \bar{c}_1')} I_2 \ldots$$

such that (1) $I_0 = I$, (2) $T_i \in S$ for each $i \geq 0$, and (3) $J \models S$ with $J = \bigcup_{i \geq 0} I_i$. The instance J is the (potentially infinite) *result* of this chase sequence, which always exists. The chase sequence is *fair* if whenever a TGD $T \in S$ is applicable to a tuple $(\bar{c}, \bar{c}')$ in some I_i, then $I_j \xrightarrow{T,\, (\bar{c}, \bar{c}')} I_{j+1}$ is part of the sequence for some $j \geq i$. Note that our chase is oblivious, that is, a TGD is triggered whenever its body is satisfied, even if also its head is already satisfied. As a consequence, every fair chase sequence for I with S leads to the same result, up to isomorphism. Thus, we can refer to *the* result of chasing I with S, denoted $\mathsf{ch}_S(I)$.

▶ **Lemma 1.** *Let S be a finite set of TGDs and I an instance. Then for every model J of S with $I \subseteq J$, there is a homomorphism h from $\mathsf{ch}_S(I)$ to J that is the identity on $\mathsf{adom}(I)$.*

For sets S of TGDs from $\mathbb{G} \cap \mathbb{FULL}$, we may also chase a CQ $q(\bar{x})$ with S, denoting the result with $\mathsf{ch}_S(q)$. What we mean is the (finite!) result of chasing database D_q with S and viewing the resulting as a CQ with answer variables $\bar{x}$.

Parameterized Complexity. A *counting problem* over a finite alphabet Λ is a function $P : \Lambda^* \to \mathbb{N}$ and a *parameterized counting problem* over Λ is a pair (P, κ), with P a counting problem over Λ and κ the *parameterization* of P, a function $\kappa : \Lambda^* \to \mathbb{N}$ that is computable in PTime. An example of a parameterized counting problem is #$\mathsf{pClique}$ in which P maps (a suitable encoding of) each pair (G, k) with G an undirected graph and $k \geq 0$ a clique size to the number of k-cliques in G, and where $\kappa(G, k) = k$. Another example is #$\mathsf{pDomSet}$ where P maps each pair (G, k) to the number of dominating sets of size k, and where again $\kappa(G, k) = k$.

A counting problem P is a *decision problem* if the range of P is $\{0, 1\}$, and a *parameterized decision problem* is defined accordingly. An example of a parameterized decision problem is $\mathsf{pClique}$ in which P maps each pair (G, k) to 1 if the undirected graph G contains a k-clique and to 0 otherwise, and where $\kappa(G, k) = k$.

A parameterized problem (P, κ) is *fixed-parameter tractable* (fpt) if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that $P(x)$ can be computed in time $|x|^{O(1)} \cdot f(\kappa(x))$ for all inputs x.

We use FPT to denote the class of all parameterized counting problems that are fixed-parameter tractable.

A *Turing fpt-reduction* from a parameterized counting problem (P_1, κ_1) to a parameterized counting problem (P_2, κ_2) is an algorithm that computes P_1 with oracle access to P_2, runs within the time bounds of fixed parameter tractability for (P_1, κ_1), and when started on input x only makes oracle calls with argument y such that $\kappa_2(y) \leq f(\kappa_1(x))$, for some computable function f. The reduction is called a *parsimonious fpt-reduction* if only a single oracle call is made at the end of the computation and its output is then returned as the output of the algorithm without any further modification.

A parameterized counting problem (P, κ) is $\#\text{W}[1]$*-easy* if it can be reduced to #pClique and it is $\#\text{W}[1]$*-hard* if #pClique reduces to (P, κ), both in terms of Turing fpt-reductions. W[1]-easiness and -hardness are defined analogously, but using pClique in place of #pClique, and likewise for #W[2] and #pDomSet, and for #A[2] and the parameterized problem of counting the answers to CQs, the parameter being the size of the CQ. For $C \in \{\text{W}[1], \#\text{W}[1], \#\text{W}[2], \#\text{A}[2]\}$, (P, κ) is C*-equivalent* if it is C-easy and C-hard. Note that we follow [16, 19] in defining both easiness and hardness in terms of Turing fpt-reductions; stronger notions would rely on parsimonious fpt-reductions [25].

3 The Classification Without TGDs

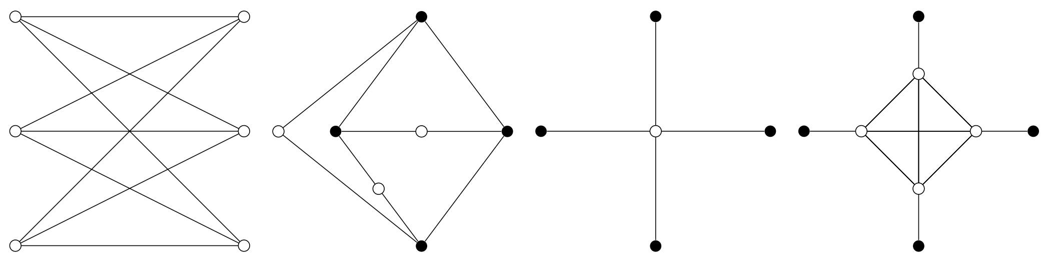
In the series of papers [20, 21, 16, 17, 19], the parameterized complexity of answer counting is studied for classes of CQs and UCQs, resulting in a rather detailed classification. We present it in this section as a reference point and as a basis for establishing our own classifications later on. We start with introducing the various structural measures that play a role in the classification.

Let $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ be a CQ. The *Gaifman graph* of q, denoted G_q, is G_{D_p} where CQ p is obtained from q by replacing answer variable x_2 with answer variable x_1 whenever $x_1 = x_2$ is an atom in q. The *treewidth (TW)* of $q(\bar{x})$ is the treewidth of G_q.

An $\bar{x}$*-component* of G_q is the undirected graph obtained as follows: (1) take the subgraph of G_q induced by vertex set $\mathsf{var}(q) \setminus \bar{x}$, (2) choose a maximal connected component (V_c, E_c), and (3) re-add all edges from G_q that contain at least one vertex from V_c. The *contract* of G_q, denoted $\mathsf{contract}(G_q)$, is the restriction of G_q to the variables in $\bar{x}$, extended with every edge $\{x_1, x_2\} \subseteq \bar{x}$ such that x_1, x_2 co-occur in some $\bar{x}$-component of G_q. We shall often be interested in the treewidth of the contract of a CQ q, which we refer to as the *contract treewidth (CTW)* of q.

The *starsize (SS)* of q is the maximum number of answer variables in any $\bar{x}$-component of G_q. Note that the same notion is called strict starsize in [16] and dominating starsize in [19]. It is different from the original notion of starsize from [20, 21].

A set of quantified variables S in q is *node-well-linked* if for every two disjoint sets $S_1, S_2 \subseteq S$ of the same cardinality, there are $|S_1|$ vertex disjoint paths in G_q that connect the vertices in S_1 with the vertices in S_2. For example, S is node-well-linked if $D_q|_S$ takes the form of a grid or of a clique. A matching M from the answer variables $\bar{x}$ to the quantified variables $\mathsf{var}(q) \setminus \bar{x}$ in the graph G_q (in the standard sense of graph theory) is *linked* if the set S of quantified variables that occur in M is node-well-linked. The *linked matching number (LMN)* of q is the size of the largest linked matching from $\bar{x}$ to $\mathsf{var}(q) \setminus \bar{x}$ in G_q. One should think of the linked matching number as a strengthening of starsize. We do not only demand that many answer variables are interlinked by the same $\bar{x}$-component, but additionally require that this component is sufficiently large and highly connected ('linked').



(a) TW = 3, CTW = 1 **(b)** CTW = 3, SS = 2 **(c)** SS = 4, LMN = 1 **(d)** LMN = 4

Figure 1 Examples for structural measures: Example (a) is the (3,3)-complete bipartite graph, the contract of Example (b) is the 4-clique, Example (c) is a 4-star, and Example (d) is a 4-star with the 4-clique in the centre. Filled nodes are answer variables, hollow nodes are quantified variables.

Figure 1 contains some example CQs with associated measures. For a class of CQs $\mathbb{C}$, the contract treewidths of CQs in $\mathbb{C}$ being bounded by a constant implies that the same is true for starsizes, and bounded starsizes in turn imply bounded linked matching numbers. There are no implications between bounded treewidths and bounded contract treewidths; in Figure 1, Example (a) generalizes to any treewidth while always having contract treewidth 1 and Example (c), which has contract treewidth 3, generalizes to any contract treewidth (and starsize) while always having treewidth 1. See also [16, 19] for additional examples.

It is a fundamental observation that cores of CQs are guaranteed to have minimum measures among all equivalent CQs, as stated by the following lemma [16, 19].

▶ **Lemma 2.** *If a CQ q is equivalent to a CQ of treewidth k, then* $\mathsf{core}(q)$ *has treewidth at most k. The same is true for contract treewidth, starsize, and linked matching number.*

An additional ingredient needed to formulate the classification for UCQs emerges from [17]. There, Chen and Mengel associate with every UCQ q a set of CQs $\mathsf{cl}_{\mathsf{CM}}(q)$ such that, informally speaking, counting the number of answers to q is equivalent to counting the number of answers to the CQs in $\mathsf{cl}_{\mathsf{CM}}(q)$. We now introduce this set in detail.

Two CQs $q_1(\bar{x}_1)$ and $q_2(\bar{x}_2)$ over the same schema $\mathbf{S}$ are *counting equivalent* if $\#q_1(D) = \#q_2(D)$ for all $\mathbf{S}$-databases D. Let $q(\bar{x}) = p_1 \vee \cdots \vee p_n$. The starting point for defining $\mathsf{cl}_{\mathsf{CM}}(q)$ is the observation that, by the inclusion-exclusion principle, every database D satisfies

$$\#q(D) \quad = \quad \sum_{I \subseteq [n]} (-1)^{|I|+1} \cdot \#\big(\bigwedge_{i \in I} p_i(D)\big)$$

We can manipulate this sum as follows: if there are two summands $c_1 \cdot \#\big(\bigwedge_{i \in I_1} p_i(D)\big)$ and $c_2 \cdot \#\big(\bigwedge_{i \in I_2} p_i(D)\big)$ such that $\bigwedge_{i \in I_1} p_i$ and $\bigwedge_{i \in I_2} p_i$ are counting equivalent, then delete both summands and add $(c_1 + c_2) \cdot \#(\bigwedge_{i \in I_1} p_i(D))$ to the sum. After doing this exhaustively, delete all summands with coefficient zero. The elements of $\mathsf{cl}_{\mathsf{CM}}(q)$ are all CQs $\bigwedge_{i \in I} p_i$ in the original sum that are counting equivalent to some CQ CQ $\bigwedge_{i \in J} p_i$ which remains in the sum.[2] Note that the number of CQs in $\mathsf{cl}_{\mathsf{CM}}(q)$ might be exponentially larger than the number of CQs in q and that $\mathsf{cl}_{\mathsf{CM}}(q)$ does not need to contain all CQs from the original UCQ q. The main property of $\mathsf{cl}_{\mathsf{CM}}(q)$ is as follows.

[2] This definition slightly deviates from that of Chen and Mengel, who include no two CQs that are counting equivalent. For all relevant purposes, however, the two definitions are interchangable.

▶ **Lemma 3** ([17]). *Let q be a UCQ over schema $\mathbf{S}$ and D an $\mathbf{S}$-database. Then $\#q(D)$ can be computed in polynomial time from the counts $\#q'(D)$, $q' \in \mathsf{cl}_{\mathsf{CM}}(q)$. Conversely, for every $q' \in \mathsf{cl}_{\mathsf{CM}}(q)$, there is a set of databases $D_1, \dots, D_n$ such that $\#q'(D)$ can be computed in polynomial time from the counts $\#q(D_i)$, $1 \leq i \leq n$, and $D_1, \dots, D_n$ can be computed in time $f(||q||) \cdot p(||D||)$ where f is a computable function and p is a polynomial.*

For the first part of Lemma 3, note that $\#q(D)$ can be computed from $\#q'(D)$, $q' \in \mathsf{cl}_{\mathsf{CM}}(q)$, simply by evaluating the sum derived above from the inclusion-exclusion principle, after the manipulation.

We are now ready to state the characterization.

▶ **Theorem 4** ([16, 17, 19]). *Let $\mathbb{Q} \subseteq \mathbb{UCQ}$ be recursively enumerable and have relation symbols of bounded arity, and let $\mathbb{Q}^\star = \{\mathsf{core}(q) \mid q \in \mathsf{cl}_{\mathsf{CM}}(\mathbb{Q})\}$. Then the following holds:*

1. *If the treewidths and the contract treewidths of CQs in $\mathbb{Q}^\star$ are bounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is in FPT; it is even in* PTIME *when $\mathbb{Q} \subseteq \mathbb{CQ}$.*
2. *If the treewidths of CQs in $\mathbb{Q}^\star$ are unbounded and the contract treewidths of CQs in $\mathbb{Q}^\star$ are bounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is* $\mathrm{W}[1]$*-equivalent.*
3. *If the contract treewidths of CQs in $\mathbb{Q}^\star$ are unbounded and the starsizes of CQs in $\mathbb{Q}^\star$ are bounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is* $\#\mathrm{W}[1]$*-equivalent.*
4. *If the starsizes of CQs in $\mathbb{Q}^\star$ are unbounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is* $\#\mathrm{W}[2]$*-hard.*
5. *If the linked matching numbers of CQs in $\mathbb{Q}^\star$ are unbounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is* $\#\mathrm{A}[2]$*-equivalent.*

We remark that $\mathsf{cl}_{\mathsf{CM}}(q) = \{q\}$ when q is a CQ, and thus it suffices to define $\mathbb{Q}^\star$ as $\{\mathsf{core}(q) \mid q \in \mathbb{Q}\}$ when $\mathbb{Q} \subseteq \mathbb{CQ}$ in Theorem 4.

Note that the classification given by Theorem 4 is not complete. It leaves open the possibility that there is a class of (U)CQs $\mathbb{Q}$ such that $\mathsf{AnswerCount}(\mathbb{Q})$ is $\#\mathrm{W}[2]$-hard, but neither $\#\mathrm{W}[2]$-equivalent nor $\#\mathrm{A}[2]$-equivalent. It is conjectured in [19] that such a class $\mathbb{Q}$ indeed exists and in particular that there might be classes $\mathbb{Q}$ such that $\mathsf{AnswerCount}(\mathbb{Q})$ is $\#\mathrm{W}_{\mathsf{func}}[2]$-equivalent. The classification also leaves open whether unbounded linked matching numbers is a necessary condition for $\#\mathrm{A}[2]$-hardness. While a complete classification is certainly desirable we note that, from our perspective, the most relevant aspect is the delineation of the FPT cases from the hard cases, achieved by Points 1-3 of the theorem.

4 Problems Studied and Main Results

We introduce the problems studied and state the main results of this paper. We start with ontology-mediated querying and then proceed to querying under constraints.

An *ontology* $\mathcal{O}$ is a finite set of TGDs. An *ontology mediated query (OMQ)* takes the form $Q = (\mathcal{O}, \mathbf{S}, q)$ where $\mathcal{O}$ is an ontology, $\mathbf{S}$ is a finite schema called the *data schema*, and q is a UCQ. Both $\mathcal{O}$ and q can use symbols from $\mathbf{S}$, but also additional symbols, and in particular $\mathcal{O}$ can 'introduce' additional symbols to enrich the vocabulary available for querying. We assume w.l.o.g. that all relation symbols in q that are not from $\mathbf{S}$ occur also in $\mathcal{O}$. This can always be achieved by introducing dummy TGDs $R(\bar{x}) \rightarrow R(\bar{x})$. When $\mathcal{O}$ and q only use symbols from $\mathbf{S}$, then we say that the data schema of Q is *full*. The *arity* of Q is defined as the arity of q. We write $Q(\bar{x})$ to emphasize that the answer variables of q are $\bar{x}$ and for brevity often refer to the data schema simply as the schema.

A tuple $\bar{c} \in \mathsf{adom}(D)^{|\bar{x}|}$ is an *answer* to Q over D if $\bar{c} \in q(I)$ for each model I of $\mathcal{O}$ with $I \supseteq D$. The *evaluation of* $Q(\bar{x})$ over D, denoted $Q(D)$, is the set of all answers to Q over D.

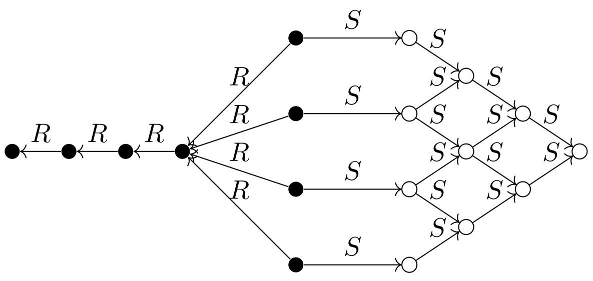


Figure 2 CQ q_4 from Example 5. Filled circles indicate answer variables.

Note that, as a consequence of Lemma 1, $Q(D) = q(\mathsf{ch}_{\mathcal{O}}(D))$ for every OMQ $Q = (\mathcal{O}, \mathbf{S}, q)$ and $\mathbf{S}$-database D.

An *OMQ language* is a class of OMQs. For a class of TGDs $\mathbb{C}$ and a class of UCQs $\mathbb{Q}$, we write $(\mathbb{C}, \mathbb{Q})$ to denote the OMQ language that consists of all OMQs $(\mathcal{O}, \mathbf{S}, q)$ where $\mathcal{O}$ is a set of TGDs from $\mathbb{C}$ and $q \in \mathbb{Q}$. For example, we may write $(\mathbb{G} \cap \mathbb{FULL}, \mathbb{UCQ})$. We say that an OMQ language $(\mathbb{C}, \mathbb{Q})$ has *full data schema* if every OMQ in it has. If $\mathbb{Q}$ is an OMQ language, the problem $\mathsf{AnswerCount}(\mathbb{Q})$ is defined exactly as in Section 2 with query language $\mathbb{Q}$.

Our first main result is a counterpart of Theorem 4 for OMQs from $(\mathbb{G}, \mathbb{UCQ})$ based on the full schema. To illustrate the effect of adding an ontology, we first observe that the ontology interacts with all of the measures in Theorem 4.

▶ **Example 5.** Let $\mathcal{O} = \{R(x, y) \to S(x, y)\}$ and $\mathbf{S} = \{R, S\}$. For all $n \geq 0$, let

$$\begin{aligned} q_n(x_1, \ldots, x_n, z_1, \ldots, z_n) \quad = \quad & \exists_{1<i+j<n+2}\, y_{i,j} \bigwedge_{1\leq i\leq n} R(x_i, z_1) \wedge \bigwedge_{1\leq i<n} R(z_i, z_{i+1}) \wedge \\ & \bigwedge_{i+j=n+1} S(x_i, y_{i,j}) \wedge \\ & \bigwedge_{2<i+j<n+2} S(y_{i+1,j}, y_{i,j}) \wedge S(y_{i,j+1}, y_{i,j}). \end{aligned}$$

Then q_n is a core of treewidth $\lfloor \frac{n}{2} \rfloor$, contract treewidth n, starsize n, and linked matching number n. But the OMQ $(\mathcal{O}, \mathbf{S}, q_n)$ is equivalent to $(\mathcal{O}, \mathbf{S}, p_n)$ with p_n obtained from q_n by dropping all S-atoms. Since p_n is tree-shaped and has no quantified variables, all measures are at most 1. Figure 2 depicts query q_4.

Before we state our characterization, we observe as a preliminary that OMQs from $(\mathbb{G}, \mathbb{UCQ})$ can be rewritten into equivalent ones from $(\mathbb{G} \cap \mathbb{FULL}, \mathbb{UCQ})$, that is, existential quantifiers can be removed from rule heads when the actual query is adjusted in a suitable way. This has already been observed, for example, in [5].

▶ **Theorem 6.** *For every OMQ $Q \in (\mathbb{G}, \mathbb{UCQ})$, there is an equivalent OMQ from $(\mathbb{G} \cap \mathbb{FULL}, \mathbb{UCQ})$ that can be effectively computed.*

The proof of Theorem 6 is constructive, that is, it provides an explicit way of computing, given an OMQ $Q = (\mathcal{O}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$, an equivalent OMQ from $(\mathbb{G} \cap \mathbb{FULL}, \mathbb{UCQ})$. We denote this OMQ with $Q^\exists = (\mathcal{O}^\exists, \mathbf{S}, q^\exists)$ and call it the $\exists$-*rewriting* of Q. It is worth noting that even if q contains no equality atoms, such atoms might be introduced during the construction of $q^\exists$. This is the main reason for admitting equality atoms in (U)CQs in this paper in the first place.

For OMQs $Q \in (\mathbb{G}, \mathbb{UCQ})$, we define a set $\mathsf{cl}_{\mathsf{CM}}(Q)$ in exact analogy with the definition of $\mathsf{cl}_{\mathsf{CM}}(q)$ for UCQs q, that is, for $Q = (\mathcal{O}, \mathbf{S}, p_1 \vee \cdots \vee p_n)$, we use the OMQs $(\mathcal{O}, \mathbf{S}, p_i)$ in place of the CQs p_i from the UCQ q in the definition of $\mathsf{cl}_{\mathsf{CM}}(Q)$. This requires the use of counting equivalence for OMQs, which is as defined in the expected way. For a class $\mathbb{Q} \subseteq (\mathbb{G}, \mathbb{UCQ})$, we now identify a class $\mathbb{Q}^\star$ of CQs by setting

$$\mathbb{Q}^\star = \{\mathsf{core}(\mathsf{ch}_{\mathcal{O}^\exists}(p)) \mid \exists Q \in \mathbb{C} : (\mathcal{O}^\exists, \mathbf{S}, p) \in \mathsf{cl}_{\mathsf{CM}}(Q^\exists)\}.$$

Our main result is now as follows.

▶ **Theorem 7.** *Let $\mathbb{Q} \subseteq (\mathbb{G}, \mathbb{UCQ})$ be a recursively enumerable class of OMQs with full data schema and relation symbols of bounded arity. Then the following hold:*

1. *If the treewidths and contract treewidths of CQs in $\mathbb{Q}^\star$ are bounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is in FPT.*
2. *If the treewidths of CQs in $\mathbb{Q}^\star$ are unbounded and the contract treewidths of CQs in $\mathbb{Q}^\star$ are bounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is* W[1]*-equivalent.*
3. *If the contract treewidths of CQs in $\mathbb{Q}^\star$ are unbounded and the starsizes of CQs in $\mathbb{Q}^\star$ are bounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is* #W[1]*-equivalent.*
4. *If the starsizes of CQs in $\mathbb{Q}^\star$ are unbounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is* #W[2]*-hard.*
5. *If the linked matching numbers of CQs in $\mathbb{Q}^\star$ are unbounded, then* $\mathsf{AnswerCount}(\mathbb{Q})$ *is* #A[2]*-equivalent.*

Points 1 to 5 of Theorem 7 parallel exactly those of Theorem 4, but of course the definition of $\mathbb{Q}^\star$ is a different one. It is through this definition that we capture the potential interaction between the ontology and the structural measures. Note, for example, that the class of OMQs $(\mathcal{O}, \mathbf{S}, q_n)$, $n \geq 1$, from Example 5 would be classified as #A[2]-equivalent if $\mathsf{core}(\mathsf{ch}_{\mathcal{O}^\exists}(p))$ was replaced with p in the definition of $\mathbb{Q}^\star$ while it is in fact in FPT. Also note that the PTime statement in Point 1 of Theorem 4 is absent in Theorem 7. In fact, evaluating Boolean OMQs from $(\mathbb{G}, \mathbb{UCQ})$ is 2ExpTime-complete [12] and since for Boolean OMQs evaluation coincides with answer counting, PTime cannot be attained.

Our second main result concerns querying under integrity constraints that take the form of guarded TGDs. In contrast to OMQs, the constraints are thus not used for deductive reasoning, but instead give rise to a promise regarding the shape of the input database. Following [5], we define a *constraint-query specification* (CQS) as a triple $S = (\mathcal{T}, \mathbf{S}, q)$ where $\mathcal{T}$ is a set of TGDs over finite schema $\mathbf{S}$ and q a UCQ over $\mathbf{S}$. We call $\mathcal{T}$ the set of *integrity constraints*. Overloading notation, we write $(\mathbb{C}, \mathbb{Q})$ for the class of CQSs in which the set of integrity constraints is formulated in the class of TGDs $\mathbb{C}$, and the query is coming from the class of queries $\mathbb{Q}$. It will be clear from the context whether $(\mathbb{C}, \mathbb{Q})$ is an OMQ language or a class of CQSs. Every class $\mathbb{C}$ of CQSs gives rise to the following answer counting problem.

PROBLEM :	$\mathsf{AnswerCount}(\mathbb{C})$
INPUT :	A set of TGDs $\mathcal{T}$, a query q, and an $\mathbf{S}$-database D that satisfies $\mathcal{T}$ such that $(\mathcal{T}, \mathbf{S}, q) \in \mathbb{C}$.
OUTPUT :	$\#q(D)$

Our second main result parallels Theorems 4 and 7. We refrain from explicitly listing all cases again.

▶ **Theorem 8.** *Let $\mathbb{Q} \subseteq (\mathbb{G}, \mathbb{UCQ})$ be a recursively enumerable class of CQSs with relation symbols of bounded arity. Then Statements 1-5 of Theorem 7 hold.*

Note that the delineation of the considered complexities is identical for ontology-mediated querying and for querying under constraints. In particular, Theorem 8 (implicitly) uses exactly the same class of CQs $\mathbb{Q}^\star$ and the same associated measures.

It would be interesting to know whether $\mathsf{AnswerCount}(\mathbb{Q})$ being in FPT coincides with $\mathsf{AnswerCount}(\mathbb{Q})$ being in PTime for classes of CQSs $\mathbb{Q} \subseteq (\mathbb{G}, \mathbb{CQ})$. Note that this is the case for evaluation in the presence of constraints that are guarded TGDs [8, 7] and also for answer counting without constraints [16]. The proof of these results, however, break in our setting.

We derive Theorem 8 from Theorem 7 by means of reduction. In fact, Theorem 8 is a consequence of Theorem 7 and the following result.

▶ **Theorem 9.** *Let $\mathbb{C} \subseteq (\mathbb{G}, \mathbb{UCQ})$ be a recursively enumerable class of CQSs and let $\mathbb{C}'$ be $\mathbb{C}$ viewed as a class of OMQs based on the full schema.*[3] *Then there is a Turing fpt-reduction from $\mathsf{AnswerCount}(\mathbb{C}')$ to $\mathsf{AnswerCount}(\mathbb{C})$ and there is a parsimonious polynomial time reduction from $\mathsf{AnswerCount}(\mathbb{C})$ to $\mathsf{AnswerCount}(\mathbb{C}')$.*

The reduction from $\mathsf{AnswerCount}(\mathbb{C})$ to $\mathsf{AnswerCount}(\mathbb{C}')$ is immediate: given a set of guarded TGDs $\mathcal{T}$, a CQ q, and an $\mathbf{S}$-database D that satisfies $\mathcal{T}$, we can view $(\mathcal{T}, \mathbf{S}, q)$ as an OMQ Q based on the full schema and return $\#Q(D)$ as $\#q(D)$. It is easy to see that this is correct.

For the converse reduction, we are given a $Q = (\mathcal{O}, \mathbf{S}, q)$ that is a CQS from $\mathbb{C}$ viewed as an OMQ and an $\mathbf{S}$-database D. It seems a natural idea to simply view Q as a CQS, which it originally was, and replace D with $\mathsf{ch}_{\mathcal{O}}(D)$ so that the promise is satisfied, and to then return $\#q(\mathsf{ch}_{\mathcal{O}}(D))$ as $\#Q(D)$. However, there are two obstacles. First, $\mathsf{ch}_{\mathcal{O}}(D)$ need not be finite; and second, chasing adds fresh constants which changes the answer count. We solve the first problem by replacing the infinite chase with a (finite!) database $D^\star$ that extends D and satisfies $\mathcal{O}$. The following result from [5] is essentially a consequence of $\mathbb{G}$ being finitely controllable.

▶ **Theorem 10** ([5]). *Given an ontology $\mathcal{O} \subseteq \mathbb{G}$, an $\mathbf{S}$-database D, and an $n \geq 1$, one can effectively construct a finite database D^* that satisfies the following conditions:*

1. *$D^* \models \mathcal{O}$ and $D \subseteq D^*$;*
2. *$\bar{a} \in q(D^*)$ iff $\bar{a} \in Q(D)$ for all OMQs $(\mathcal{O}, \mathbf{S}, q)$ where q has at most n variables and for all tuples $\bar{a}$ that use only constants in $\mathsf{adom}(D)$.*

The construction of D^* takes time $||D||^{O(1)} \cdot f(||\mathcal{O}|| + n)$.

To address the second problem, we correct the count. Note that this cannot be done by introducing fresh unary relation symbols as markers to distinguish the original constants from those introduced by the chase as this would require us to change the query. We instead use an approach inspired by [16]. The idea is to compute $\#q(D')$ on a set of databases D' obtained from $D^\star$ by cloning constants in $\mathsf{adom}(D) \subseteq \mathsf{adom}(D^\star)$. The results can be arranged in a system of equations whose coefficients form a Vandermonde matrix. Finally, the system can be solved to obtain $\#q(D)$. This is formalized by the following lemma where we use $\mathsf{clones}(D)$ to denote the class of all $\mathbf{S}$-databases that can be obtained from $\mathbf{S}$-database D by cloning constants. A proof is in the appendix of the long version.

▶ **Lemma 11.** *Fix a constant r. There is a polynomial time algorithm that, given a UCQ $q(\bar{x})$ over schema $\mathbf{S}$ with $\mathsf{ar}(\mathbf{S}) \leq r$, an $\mathbf{S}$-database D, and a set $F \subseteq \mathsf{adom}(D)$, computes $\#(q(D) \cap F^{|\bar{x}|})$ using an oracle for $\mathsf{AnswerCount}(\{q\}, \mathsf{clones}(D))$.*

[3] Syntactically, a CQS $(\mathcal{T}, \mathbf{S}, q)$ and an OMQ $(\mathcal{T}, \mathbf{S}, q)$ are actually the same thing except that the definition of CQSs is more strict regarding the schema $\mathbf{S}$; as a consequence when viewing a CQS as an OMQ, the latter is based on the full schema.

Now for the reduction from $\mathsf{AnswerCount}(\mathbb{C}')$ to $\mathsf{AnswerCount}(\mathbb{C})$ claimed in Theorem 9. Let $Q(\bar{x}) = (\mathcal{O}, S, q)$ be a CQS from $\mathbb{C}$ viewed as an OMQ, and let D be an $\mathbf{S}$-database. We first construct the database D^* as per Theorem 10 with n being the number of variables in q. We then apply the algorithm asserted by Lemma 11 with D^* in place of D and with $F := \mathsf{adom}(D)$. Cloning preserves guarded TGDs and thus we can use the oracle (which can compute $\#q(D')$ for any $\mathbf{S}$-database D' that satisfies $\mathcal{O}$) for computing $\mathsf{AnswerCount}(\{q\}, \mathsf{clones}(D))$ as required by Lemma 11.

5 Proof of Theorem 7

We first establish the upper bounds in Theorem 7, starting with the FPT upper bound from Point 1. Let $\mathbb{C} \subseteq (\mathbb{G}, \mathbb{UCQ})$ be a class of OMQs such that the treewidths and the contract treewidths of CQs in $\mathbb{Q}^\star$ are bounded by a constant k. Given an OMQ $Q \in \mathbb{C}$ and an $\mathbf{S}$-database D, we first replace Q by its $\exists$-rewriting $Q^\exists = (\mathcal{O}^\exists, \mathbf{S}, q^\exists)$. Since Q is equivalent to $Q^\exists$ we have that $\#Q(D) = \#Q^\exists(D)$, thus it suffices to compute the latter count. The first part of Lemma 3 clearly lifts from UCQs to OMQs, see also the remark after that lemma. We can thus compute $\#Q^\exists(D)$ (essentially by applying the inclusion-exclusion principle) within the time requirements of FPT once we have computed $\#(\mathcal{O}^\exists, \mathbf{S}, p)(D)$ for all p such that $(\mathcal{O}^\exists, \mathbf{S}, p) \in \mathsf{cl}_{\mathsf{CM}}(Q^\exists)$. By the universality of the chase, $\#(\mathcal{O}^\exists, \mathbf{S}, p)(D) = \#p(\mathsf{ch}_{\mathcal{O}^\exists}(D))$. Moreover, since $\mathcal{O}^\exists$ is from $\mathbb{G} \cap \mathbb{FULL}$, $\mathsf{ch}_{\mathcal{O}^\exists}(D)$ is finite and can be computed within the time requirements of FPT.

Thus, to compute $\#Q(D)$ it is enough to compute $\#p(\mathsf{ch}_{\mathcal{O}^\exists}(D))$ for all CQs p such that $(\mathcal{O}^\exists, \mathbf{S}, p) \in \mathsf{cl}_{\mathsf{CM}}(Q^\exists)$ or, equivalently, to compute $\#\mathsf{core}(\mathsf{ch}_{\mathcal{O}^\exists}(p))(\mathsf{ch}_{\mathcal{O}^\exists}(D))$ for all p with $(\mathcal{O}^\exists, \mathbf{S}, p) \in \mathsf{cl}_{\mathsf{CM}}(Q^\exists)$. But the CQs $\mathsf{core}(\mathsf{ch}_{\mathcal{O}^\exists}(p))$ for these p are exactly the CQs from $\mathbb{Q}^\star$ and thus their treewidths and contract treewidths are bounded by k. Consequently, we can apply the fpt algorithm from Point 1 of Theorem 4 as a black box and overall obtain an FPT procedure. The remaining upper bounds from Theorem 7 can be proved analogously, exploiting that easiness for W[1] and #W[1] is defined in terms of Turing fpt-reductions.

We next turn towards lower bounds, which are proved by a sequence of Turing fpt-reductions. The first such reduction consists in transitioning to the $\exists$-rewritings of the OMQs in the original class. The second reduction enables us to consider OMQs that use CQs rather than UCQs.[4] And in the third reduction, we remove ontologies altogether, that is, we reduce classes of CQs to classes of OMQs. We start with the first reduction.

▶ **Theorem 12.** *Let $\mathbb{C} \subseteq (\mathbb{G}, \mathbb{UCQ})$ be recursively enumerable and let $\mathbb{C}' \subseteq (\mathbb{G} \cap \mathbb{FULL}, \mathbb{UCQ})$ be the class of $\exists$-rewritings of OMQs from $\mathbb{C}$. There is a parsimonious fpt-reduction from* $\mathsf{AnswerCount}(\mathbb{C}')$ *to* $\mathsf{AnswerCount}(\mathbb{C})$.

Proof. Given a $Q = (\mathcal{O}, \mathbf{S}, q) \in \mathbb{C}'$ and an $\mathbf{S}$-database D, find some $Q' = (\mathcal{O}', \mathbf{S}, q') \in \mathbb{C}$ such that Q is an $\exists$-rewriting of Q' by recursively enumerating $\mathbb{C}'$ and exploiting that OMQ equivalence is decidable in $(\mathbb{G}, \mathbb{UCQ})$ [4], then compute and return $\#Q'(D)$. ◀

The second reduction is given by the following theorem.

▶ **Theorem 13.** *Let $\mathbb{C} \subseteq (\mathbb{G} \cap \mathbb{FULL}, \mathbb{UCQ})$ be a recursively enumerable class of OMQs with full schema and relation symbols of bounded arity, and let $\mathbb{C}' \subseteq (\mathbb{G} \cap \mathbb{FULL}, \mathbb{CQ})$ be the class of OMQs $\{Q' \mid \exists Q \in \mathbb{C} : Q' \in \mathsf{cl}_{\mathsf{CM}}(Q)\}$. Then there is a Turing fpt-reduction from* $\mathsf{AnswerCount}(\mathbb{C}')$ *to* $\mathsf{AnswerCount}(\mathbb{C})$.

[4] The construction of $Q^\exists$ may produce a UCQ even if the original OMQ contained a CQ.

In [17], Chen and Mengel establish Theorem 13 in the special case where ontologies are empty. However, a careful analysis reveals that their proof extends to recursively enumerable classes $\mathbb{D}$ of databases over some schema $\mathbf{S}$ that satisfy the following conditions:

1. $\mathbb{D}$ is closed under disjoint unions, direct products, and contains the *well of positivity* $D_{\mathbf{S}}^{\top}$, that is, the $\mathbf{S}$-database with a single constant c defined as $D_{\mathbf{S}}^{\top} = \{R(c, \dots, c) \mid R \in \mathbf{S}\}$;
2. counting equivalence and semi-counting equivalence between CQs over $\mathbb{D}$ is decidable, where CQs $q_1(\bar{x}_1)$ and $q_2(\bar{x}_2)$ over the same schema $\mathbf{S}$ are *semi-counting equivalent* if they are counting equivalent over all $\mathbf{S}$-databases D with $\#q_1(D) > 0$ and $\#q_2(D) > 0$.

This allows us to establish Theorem 13 by observing that models of TGDs are closed under the operations mentioned in Point 1and that the two equivalence problems in Point 2 are both decidable. Details are in the appendix of the long version.

We next give the reduction that removes ontologies.

▶ **Theorem 14.** *Let $\mathbb{C} \subseteq (\mathbb{G} \cap \mathbb{FULL}, \mathbb{CQ})$ be a recursively enumerable class of OMQs with full schema and relation symbols of bounded arity. There is a class $\mathbb{C}' \subseteq \mathbb{CQ}$ that only contains cores and such that:*

1. *there is a Turing fpt-reduction from* $\mathsf{AnswerCount}(\mathbb{C}')$ *to* $\mathsf{AnswerCount}(\mathbb{C})$*;*
2. *for every OMQ $Q = (\mathcal{O}, \mathbf{S}, q) \in \mathbb{C}$, we find a CQ $p \in \mathbb{C}'$ such that the treewidth of p is equal to that of* $\mathsf{core}(\mathsf{ch}_{\mathcal{O}}(q))$*, and likewise for contract treewidth, starsize, and linked matching number.*

Using Theorems 12, 13, and 14, we can make use of the lower bounds for classes of (U)CQs stated in Theorem 4 to prove the lower bounds in Theorem 7. Let us consider, for example, the W[1] lower bound from Point 2. Take a class $\mathbb{C}_0 \subseteq (\mathbb{G}, \mathbb{UCQ})$ of OMQs such that the treewidths of CQs in

$$\mathbb{Q}^{\star} = \{\mathsf{core}(\mathsf{ch}_{\mathcal{O}^{\exists}}(p)) \mid \exists Q \in \mathbb{C}_0 : (\mathcal{O}^{\exists}, \mathbf{S}, p) \in \mathsf{cl}_{\mathsf{CM}}(Q^{\exists})\}$$

are unbounded. Theorems 12 and 13 give a Turing fpt-reduction from $\mathsf{AnswerCount}(\mathbb{C})$ to $\mathsf{AnswerCount}(\mathbb{C}_0)$ where

$$\mathbb{C} = \{Q' \mid \exists Q \in \mathbb{C}_0 : Q' \in \mathsf{cl}_{\mathsf{CM}}(Q^{\exists})\}.$$

By assumption, the treewidths of the CQs $\mathsf{core}(\mathsf{ch}_{\mathcal{O}}(q))$, $(\mathcal{O}, \mathbf{S}, q) \in \mathbb{C}$, are unbounded. Theorem 14 thus yields a Turing fpt-reduction from $\mathsf{AnswerCount}(\mathbb{C}')$ to $\mathsf{AnswerCount}(\mathbb{C})$ for some class of CQs $\mathbb{C}'$ that are all cores and such that the treewidths of CQs in $\mathbb{C}'$ are unbounded. By Point 2 of Theorem 4, $\mathsf{AnswerCount}(\mathbb{C}')$ is W[1]-hard. Composing the reductions, we obtain a Turing fpt-reduction from $\mathsf{AnswerCount}(\mathbb{C}')$ to $\mathsf{AnswerCount}(\mathbb{C}_0)$. The other lower bounds can be proved analogously.

Now for the proof of Theorem 14. It in turn uses two consecutive fpt-reductions. In the first step, we show that we can assume that every variable in a CQ (inside an OMQ) is marked by a unary relation symbol that identifies the variable. The *marking* of a CQ q over schema $\mathbf{S}$ is the CQ q^m obtained from q by adding the atom $R_x(x)$, for each $x \in \mathsf{var}(q)$ and with R_x a fresh unary relation symbol. Note that q^m is over schema $\mathbf{S}^m$ obtained from $\mathbf{S}$ by adding all the fresh symbols. The *core-chased marking* of an OMQ $Q = (\mathcal{O}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{CQ})$ is the OMQ $Q^m = (\mathcal{O}, \mathbf{S}^m, \mathsf{core}(\mathsf{ch}_{\mathcal{O}}(q))^m) \in (\mathbb{G}, \mathbb{CQ})$. We lift this to classes of OMQs $\mathbb{C}$ as expected, that is, $\mathbb{C}^m = \{Q^m \mid Q \in \mathbb{C}\}$.

▶ **Lemma 15.** *Let $\mathbb{C} \subseteq (\mathbb{G} \cap \mathbb{FULL}, \mathbb{CQ})$ be a recursively enumerable class of OMQs with full schema and relation symbols of bounded arity. Then there is a Turing fpt-reduction from* $\mathsf{AnswerCount}(\mathbb{C}^m)$ *to* $\mathsf{AnswerCount}(\mathbb{C})$*.*

The proof of Lemma 15 in the appendix of the long version lifts a corresponding proof from [16] that applies to CQs without ontologies, essentially by verifying that the proof also applies to classes of databases chased with an ontology from $\mathbb{G} \cap \mathbb{FULL}$. We next observe that in the presence of markings it is possible to get rid of ontologies, in the following sense.

▶ **Lemma 16.** *Let $\mathbb{C}^m \subseteq (\mathbb{G} \cap \mathbb{FULL}, \mathbb{CQ})$ be a recursively enumerable class of OMQs with full schema and relation symbols of bounded arity that are core-chased markings. There exists a class $\mathbb{C} \subseteq \mathbb{CQ}$ of cores with relation symbols of bounded arity such that:*

1. *there is a Turing fpt-reduction from* $\mathsf{AnswerCount}(\mathbb{C})$ *to* $\mathsf{AnswerCount}(\mathbb{C}^m)$*;*
2. *$\mathbb{C}$ is based on the same Gaifman graphs as $\mathbb{C}^m$: $\{G_q \mid q \in \mathbb{C}^m\} = \{G_q \mid (\mathcal{O}, \mathbf{S}, q) \in \mathbb{C}\}$.*

We provide a proof of Lemma 16 below. Before, however, we show how Theorem 14 follows from Lemmas 15 and 16.

Proof of Theorem 14. Let $\mathbb{C} \subseteq (\mathbb{G} \cap \mathbb{FULL}, \mathbb{CQ})$ be a recursively enumerable class of OMQs with full schema and relation symbols of bounded arity. From Lemma 16, we obtain a class $\mathbb{C}'$ of CQs that are cores and are based on the same Gaifman graphs as $\mathbb{C}^m$. This is the class whose existence is postulated by Theorem 14. We briefly argue that Points 1 and 2 of that theorem are satisfied. The Turing fpt-reduction required by Point 1 is the composition of the reductions asserted by Lemmas 15 and 16. Point 2 is a consequence of the facts that (1) the structural measures of a CQ are defined through its Gaifman graph, (2) $\mathbb{C}'$ is based on the same Gaifman graphs as $\mathbb{C}^m$, and (3) marking a CQ does not affect its Gaifman graph. ◀

Now for the announced proof of Lemma 16, a key ingredient to the proof of Theorem 7.

Proof of Lemma 16. To prove the lemma, we define the required class of CQs $\mathbb{C}$ and describe an fpt algorithm that

- takes as an input a query $q \in \mathbb{C}$ over schema $\mathbf{S}$ and an $\mathbf{S}$-database D,
- has access to an oracle for $\mathsf{AnswerCount}(\mathbb{C}^m)$, and
- outputs $\#q(D)$.

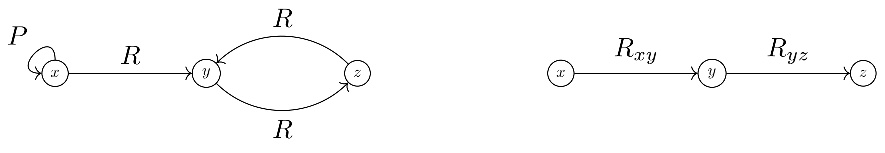
A guarded set in a database D is a set $S \subseteq \mathsf{adom}(D)$ such that all constants in S jointly occur in a fact in D, possibly together with additional constants. With a maximal guarded set, we mean a guarded set that is maximal regarding set inclusion.

The class $\mathbb{C}$ contains one CQ q^s for every OMQ $Q = (\mathcal{O}, \mathbf{S}^m, q^m) \in \mathbb{C}$ that is formulated in a different schema introduced below (whence the superscript 's'). Fix a total order on $\mathsf{var}(q^m)$. For every guarded set S in D_q, let $\overline{S}$ be the tuple that contains the variables in S in the fixed order. Now q^s contains, for every maximal guarded set S in D_q, the atom $R_S(\overline{S})$ where R_S is a fresh relation symbol of arity $|S|$. Note that q^s is self-join free, that is, it contains no two different atoms that use the same relation symbol. It is thus a core. Moreover, the Gaifman graph of q^s is identical to that of q^m since the maximal guarded sets of D_{q^m} are exactly those of D_{q^s}. An example of transformation from q to q^s can be found in Figure 3.

We now describe the algorithm. Let a CQ $q^s \in \mathbb{C}$ over schema $\mathbf{S}^s$ and an $\mathbf{S}^s$-database D^s be given as input. To compute $\#q^s(D^s)$, we first enumerate $\mathbb{C}^m$ to find an OMQ $Q = (\mathcal{O}, \mathbf{S}^m, q^m)$ that makes q^s belong to $\mathbb{C}$, as described above. Since Q is a core chased marking, q^m contains $R_x(x)$ for every $x \in \mathsf{var}(q^m)$. Let $\mathbf{S}$ be $\mathbf{S}^m$ without the unary symbols R_x.

Construct the $\mathbf{S}^s$-database $P = D_{q^s} \times D^s$ and then the $\mathbf{S}^m$-database

$$\begin{aligned} D^m = {} & \{R(\bar{a}) \mid R \in \mathbf{S} \text{ of arity } |\bar{a}| \text{ and } \bar{a} \text{ tuple over some guarded set } S \text{ in } P\} \cup \\ & \{R_x((x,a)) \mid x \in \mathsf{var}(q^m) \text{ and } (x,a) \in \mathsf{dom}(P)\} \end{aligned}$$



(a) CQ q **(b)** CQ q^s

Figure 3 A CQ $q() = \exists x \exists y \exists z\ P(x,x,x) \wedge R(x,y) \wedge R(y,z) \wedge R(z,y)$ and it's self-join free counterpart $q^s() = \exists x \exists y \exists z\ R_{xy}(x,y) \wedge R_{yz}(y,z)$ from the proof of Lemma 16.

$$\begin{tikzcd}
 & D^m & P^m \arrow[l, "\mathbf{S}^s \text{ to } \mathbf{S}", "\text{flood database}"'] & D^s \arrow[l, "\text{take product with } D_{q^s}", "\text{add unary } R_x \text{ symbols}"'] \\
Q^m \arrow[ur] \arrow[r, "\mathcal{D}^m\ \mathcal{O}\text{-saturated}"'] & q^m \arrow[u, "h : x \mapsto (x, h(x))"'] & q^{m,s} \arrow[l, "\mathbf{S}^s \text{ to } \mathbf{S}"'] \arrow[u, "h : x \mapsto (x, h(x))"'] & q^s \arrow[l, "\text{drop unary } R_x \text{ symbols}"'] \arrow[u, "h' : x \mapsto h(x)"]
\end{tikzcd}$$

Figure 4 The overall proof strategy of Lemma 16. The diagram depicts the intermediary queries, databases, and underlying relations.

where a tuple is over set S if it contains only constants from S, in any order and possibly with repetitions. Note that the relations R_x used in the second line are the marking relations from $\mathbf{S}^m$. It is easy to see that the maximal guarded sets of D^m are exactly those of P and that D^m is "flooded" in the sense that we cannot add any facts without introducing a new maximal guarded set. As a consequence and since $\mathcal{O}$ is a set of guarded TGDs, D^m is *$\mathcal{O}$-saturated*, meaning that $p(D^m) = p(\mathsf{ch}_{\mathcal{O}}(D^m))$ for all conjunctive queries p.

Clearly, the database D^m can be constructed within the time requirements of FPT and we can use the oracle to compute $\#Q(D^m)$. Let $q^{s,m}$ be obtained from q^s by adding $R_x(x)$ for every $x \in \mathsf{var}(q^s)$ and let P^m be obtained from P by adding $R_x(x,a)$ for every $a \in \mathsf{adom}(D^s)$. To end the proof, it suffices to show that

$$\#Q(D^m) = \#q^m(D^m) = \#q^{s,m}(P^m) = \#q^s(D^s).$$

The various databases and queries involved as well as the relationships between them are illustrated in Figure 4.

The first equality is immediate since D^m is $\mathcal{O}$-saturated. For the third equality, let $\bar{x} = x_1 \cdots x_n$ be the answer variables in q^s and for any $\bar{a} = a_1 \cdots a_n \in \mathsf{adom}(D^s)^n$, let $\bar{x} \times \bar{a}$ denote the tuple $(x_1, a_1) \cdots (x_n, a_n) \in \mathsf{adom}(P)^n$. Then $q^{s,m}(P^m) = \{\bar{x} \times \bar{a} \mid \bar{a} \in q(D^s)\}$. In fact, this follows from P being the product of D_{q^s} and D^s and from how the relation symbols R_x are used in $q^{s,m}$ and P^m.

It thus remains to deal with the second equality by showing that $q^m(D^m) = q^{s,m}(P^m)$. It is enough to observe that any function $h \colon \mathsf{var}(q^m) \to \mathsf{adom}(D^m)$ is a homomorphism from q^m to D^m if and only if it is a homomorphism from $q^{s,m}$ to P^m.

For the "if" direction, let h be a homomorphism from $q^{s,m}$ to P^m. First let $R(\bar{y})$ be an atom in q^m with $R \in \mathbf{S}$. There is a maximal guarded set S of D_{q^m} that contains all variables in $\bar{y}$. Then $R_S(\overline{S})$ is an atom in q^s and thus $R_S(h(\overline{S})) \in P$. By construction of D^m and since $\bar{y}$ is a tuple over S, this yields $R(h(\bar{y})) \in D^m$, as required. Now let $R_x(x)$ be an atom in q^m. Then $R_x(x)$ is also an atom in $q^{s,m}$ and thus $h(x) \in \{x\} \times \mathsf{adom}(D^s)$ due to the definition of P^m. But then $R_x(h(x)) \in D^m$ by definition of D^m.

For the "only if" direction, let h be a homomorphism from q^m to D^m. First consider atoms $R_S(\overline{S})$ in $q^{s,m}$. Then q^m contains an atom $R(\bar{y})$ where $\bar{y}$ contains exactly the variables in S and thus $R(h(\bar{y})) \in D^m$. By construction of D^m, $h(\bar{y})$ is thus a tuple over some guarded set in P, that is, P contains an atom $Q(\bar{a})$ where $\bar{a}$ contains all constants from $h(\bar{y})$. Let $V \subseteq \mathsf{var}(q^s)$ be the first components of the constants/pairs in $\bar{a}$. Since $Q(\bar{a}) \in P$ and by construction of P, V must be a guarded set in q^s. Now note that we must have $h(y) \in \{y\} \times \mathsf{adom}(D^s)$ for every variable y in $\bar{y}$ due to the use of the relation symbols R_y in q^m and D^m. Thus every variable from $\bar{y}$ (and thus every variable from S) occurs in V and thus $V = S$ because S is a maximal guarded set in D_{q^s}. It follows that $\bar{a}$ uses exactly the constants from $h(\bar{y})$ and not a proper superset. Also, the only atom that uses all variables from S in q^s is $R_S(\overline{S})$ and consequently $Q(\bar{a})$ must be $R_S(h(\overline{S}))$ which is thus in P, as required. It remains to deal with atoms $R_x(x)$ in $q^{s,m}$, which is straightforward as in the "if" direction. ◀

6 Meta Problems

Theorems 7 and 8 show that low values for the structural measures of treewidth, contract treewidth, starsize, and linked matching number are central to efficient answer counting. This suggests the importance of the meta problems to decide whether a given query is equivalent to a query in which some selected structural measures are small, and to construct the latter query if it exists. In the current section, we present some results on this topic both for ontology-mediated querying and for querying under constraints. The obtained results also shed some more light on the interplay between the ontology and the structural measures.

We start with querying under constraints. Our approach is as follows. For a given CQS $(\mathcal{T}, \mathbf{S}, q)$, we construct a certain CQ q' that approximates q from below under the constraints in $\mathcal{T}$ and that has small measures. Then, we show that if there is any CQ q'' that has small measures and is equivalent to q under the constraints in $\mathcal{T}$, then q' is equivalent to q. In this way, we are able to simultaneously solve the decision and computation version of the meta problem at hand. With 'approximation from below', we mean that the answers to q' are contained in those to q on all $\mathbf{S}$-databases. This should not be confused with computing a numerical approximation of the number of answers to a given query, which is a very interesting but entirely different problem.

A *set of measures* is a subset $M \subseteq \{\text{TW}, \text{CTW}, \text{SS}, \text{LMN}\}$ with the obvious meaning. For M a set of measures and $k \geq 1$, we say that a UCQ q is an M_k*-query* if for every CQ in q, every measure from M is at most k.

▶ **Definition 17.** *Let $(\mathcal{T}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ be a CQS, M a set of measures, and $k \geq 1$. An* M_k-approximation of q under $\mathcal{T}$ *is a UCQ q' such that*

1. *$q' \subseteq_{\mathcal{T}} q$,*
2. *q' is an M_k-query, and*
3. *for each UCQ q'' that satisfies Conditions 1 and 2, $q'' \subseteq_{\mathcal{T}} q'$.*

We next identify a simple way to construct M_k-approximations. Let $(\mathcal{T}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ be a CQS, M a set of meaures, and $k \geq 1$. Moreover, let ℓ be the maximum number of variables in any CQ in q. Assuming that $\mathcal{T}$ is understood from the context, we define q_k^M to be the UCQ that contains as a disjunct any CQ p such that $p \subseteq_{\mathcal{T}} q$, p is an M_k-query, and the number of variables in p is bounded by $\ell \cdot \mathsf{ar}(\mathbf{S})$. As containment between UCQs under constraints from $\mathbb{G}$ is decidable [4], given $(\mathcal{T}, \mathbf{S}, q)$ we can effectively compute q_k^M. We observe that q_k^M is an M_k-approximation of q under $\mathcal{T}$.

▶ **Lemma 18.** *Let $(\mathcal{T}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ be a CQS, M a set of measures, and $k \geq 1$. Then q_k^M is an M_k-approximation of q under $\mathcal{T}$.*

Proof. By construction, q_k^M satisfies Points 1 and 2 from Definition 17. We show that it satisfies also Point 3.

We start with some preparations. For an $\mathbf{S}$-database D, a set of TGDs $\mathcal{T}$, and a constant $a \in \mathsf{adom}(\mathsf{ch}_\mathcal{T}(D)) \setminus \mathsf{adom}(D)$, a guarded set X over D is a *generator* of a in $\mathcal{T}$ if $a \in \mathsf{adom}(\mathsf{ch}_\mathcal{T}(\mathsf{ch}_\mathcal{T}(D)|_X))$. Informally, X being a generator of a means that a is located in the tree rooted at X that the chase generates in $\mathsf{ch}_\mathcal{T}(D)$. Note that when $\mathcal{T}$ is a set of guarded TGDs, then a generator exists for every $a \in \mathsf{adom}(\mathsf{ch}_\mathcal{T}(D)) \setminus \mathsf{adom}(D)$ and furthermore, generators are *complete* in the sense that $R(\bar{a}) \in \mathsf{ch}_\mathcal{T}(D)$ with $a \in \bar{a}$ implies $R(\bar{a}) \in \mathsf{ch}_\mathcal{T}(\mathsf{ch}_\mathcal{T}(D)|_X)$ [12]. We also remark that generators need not be unique.

Let $q''(\bar{x})$ be a UCQ such that $q'' \subseteq_\mathcal{T} q$ and q'' is an M_k-query. Further, let p be a CQ in q''. We have to show that q_k^M contains a CQ p' with $p \subseteq_\mathcal{T} p'$.

We apply Theorem 10 to the database D_p, the set of TGDs $\mathcal{T}$, and the integer ℓ, defined to be the maximum number of variables of CQs in q. We obtain a database D_p^* which has the properties that $D_p^* \models \mathcal{T}$, $D_p \subseteq D_p^*$, and thus $\bar{x} \in p(D_p^*)$. By containment, $\bar{x} \in q(D_p^*)$, and thus there must be CQ q_i in q such that $\bar{x} \in q_i(D_p^*)$. From Point 2 of Theorem 10 and $|q_i| \leq \ell$, it follows that $\bar{x} \in Q(D_p)$ for the OMQ $Q = (\mathcal{T}, \mathbf{S}, q_i)$. Consequently, q_i maps into $\mathsf{ch}_\mathcal{T}(D_p)$ via some homomorphism h that is the identity on $\bar{x}$. We construct a new CQ p' as follows. For each atom $R(\bar{a})$ in q_i,

1. if all constants in $h(\bar{a})$ are from $\mathsf{adom}(p)$, then add $R(h(\bar{a}))$ to p';
2. if $h(\bar{a})$ contains some constant $a \notin \mathsf{adom}(p)$, then take a generator X for a in $\mathcal{T}$ and add all facts in $\mathsf{ch}_\mathcal{T}(p)|_X$ as atoms to p'.

The answer variables of p' are exactly those of p. It follows from the construction of p' that the identity is a homomorphism from p' to $\mathsf{ch}_\mathcal{T}(p)$. Thus $p \subseteq_\mathcal{T} p'$ and it remains to show that p' is a CQ in q_k^M. This follows from the following properties:

1. p' is an M_k-query. Follows from the fact that all guarded sets in p' are also guarded sets in p and thus the Gaifman graph of p' is a subgraph of the Gaifman graph of p, and the fact that all measures are monotone regarding subgraphs.
2. $p' \subseteq_\mathcal{T} q_i$. Due to the completeness of generators, the homomorhism h from q_i to $\mathsf{ch}_\mathcal{T}(D_p)$ is a also a homomorphism from q_i to $\mathsf{ch}_\mathcal{T}(D'_p)$. Consequently, $p' \subseteq_\mathcal{T} q_i$.
3. $|\mathsf{adom}(p')| \leq \ell \cdot \mathsf{ar}(\mathbf{S})$. For every variable in q_i, at most $\mathsf{ar}(\mathbf{S})$ variables are introduced during the construction of p'. ◀

By definition of M_k-approximations, it is clear that if $(\mathcal{T}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ is a CQS such that q is equivalent under $\mathcal{T}$ to a UCQ q' that is an M_k-query, then any M_k-approximation of q under $\mathcal{T}$ also satisfies these properties. The following is thus an immediate consequence of Lemma 18 and the fact that containment between UCQs under constraints from $\mathbb{G}$ is decidable.

▶ **Theorem 19.** *Let M be a set of measures. Given a CQS $(\mathcal{T}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ and $k \geq 1$, it is decidable whether q is equivalent under $\mathcal{T}$ to a UCQ q' that is an M_k-query. Moreover, if this is the case, then such a q' can be effectively computed.*

A particularly relevant case is $M = \{\mathrm{TW}, \mathrm{CTW}\}$, as it is linked to fixed-parameter tractability. Let $(\mathcal{T}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{CQ})$ and assume that we have computed an equivalent UCQ q' that is an M_k-query as per Theorem 19. Since the original query q is a CQ and $q \equiv_\mathcal{T} q'$, there must be a single disjunct q^* of q such that $q \equiv_\mathcal{T} q^*$. We can effectively identify q^* and count answers to q^* on any $\mathbf{S}$-database in FPT based on Theorem 4. It remains an interesting open question whether the same is true when the original query is a UCQ and, related to this, whether M_k-approximations always admit answer counting in FPT, that is, even when the original query is not equivalent under $\mathcal{T}$ to an M_k-query.

We now turn to ontology-mediated querying, using essentially the same approach to the meta problems as in the case of CQSs. We say that OMQ $Q_1(\bar{x}) = (\mathcal{O}_1, \mathbf{S}, q_1)$ is *contained* in OMQ $Q_2(\bar{x}) = (\mathcal{O}_2, \mathbf{S}, q_2)$, written $Q_1 \subseteq Q_2$, if $Q_1(D) \subseteq Q_2(D)$ for every $\mathbf{S}$-database D. Q_1 and Q_2 are *equivalent*, written $Q_1 \equiv Q_2$, if $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$. We say that an OMQ $Q = (\mathcal{O}, \mathbf{S}, q)$ is an M_k*-query* if q is.

▶ **Definition 20.** *Let $Q = (\mathcal{O}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ be an OMQ, M a set of measures, and $k \geq 1$. An M_k-*approximation of Q *is an OMQ $Q' = (\mathcal{O}', \mathbf{S}, q') \in (\mathbb{G}, \mathbb{UCQ})$ such that*

1. *$Q' \subseteq Q$,*
2. *Q' is an M_k-query, and*
3. *for each $Q'' = (\mathcal{O}'', \mathbf{S}, q'') \in (\mathbb{G}, \mathbb{UCQ})$ that satisfies Conditions 1 and 2, $Q'' \subseteq Q'$.*

We say that Q' is an M_k-approximation of Q while preserving the ontology *if it is an M_k-approximation and $\mathcal{O}' = \mathcal{O}$.*

We show in the appendix of the long version that M_k-approximations of OMQs $(\mathcal{O}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ based on the full schema and while preserving the ontology are identical to M_k-approximations of $(\mathcal{O}, \mathbf{S}, q)$ viewed as a CQS. We can thus reuse the approximations from Lemma 18 as a basis for showing the following counterpart of Theorem 19.

▶ **Theorem 21.** *Let M be a set of measures. Given an OMQ $Q = (\mathcal{O}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ based on the full schema and $k \geq 1$, it is decidable whether Q is equivalent to an OMQ $Q' = (\mathcal{O}, \mathbf{S}, q') \in (\mathbb{G}, \mathbb{UCQ})$ that is an M_k-query. Moreover, if this is the case, then such a Q' can be effectively computed.*

We next consider approximations of OMQs that need not preserve the ontology and might not assume the full schema, focussing on single structural measures rather than sets thereof. To simplify notion instead of, say, $\{\text{CTW}\}_k$-approximations, we speak of CTW_k-approximations. We only have full results for contract treewidth and starsize.

A *collapsing* of a CQ $q(\bar{x})$ is a CQ $p(\bar{x})$ that can be obtained from q by identifying variables and adding equality atoms (on answer variables). When an answer variable x is identified with a non-answer variable y, the resulting variable is x; the identification of two answer variables is not allowed. The *CTW_k-approximation* of an OMQ $Q = (\mathcal{O}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$, for $k \geq 1$, is the OMQ $Q_k^{\text{CTW}} = (\mathcal{O}, \mathbf{S}, q_k^{\text{CTW}})$ where q_k^{CTW} is the UCQ that contains as CQs all collapsings of q that have contract treewidth at most k. The *SS_k-approximation* of Q is defined accordingly, and denoted with Q_k^{SS}.

▶ **Theorem 22.** *Let $(\mathcal{O}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ be an OMQ and $k \geq 1$. Then Q_k^{CTW} is a CTW_k-approximation of Q. Moreover, if $k \geq \mathsf{ar}(\mathbf{S})$, then Q_k^{SS} is an SS_k-approximation of Q.*

The proof of Theorem 22 is non-trivial and relies on careful manipulations of databases that are tailored towards the structural measure under consideration. It gives rise to decidability results that, in contrast to Theorem 21, neither require the ontology to be preserved nor the schema to be full.

▶ **Corollary 23.** *Given an OMQ $Q = (\mathcal{O}, \mathbf{S}, q) \in (\mathbb{G}, \mathbb{UCQ})$ and $k \geq 1$, it is decidable whether Q is equivalent to an OMQ $Q' \in (\mathbb{G}, \mathbb{UCQ})$ of contract treewidth at most k. Moreover, if this is the case, then such a Q' can be effectively computed. The same is true for starsize in place of contract treewidth.*

For treewidth, we leave open decidability of the meta problem and only observe that it behaves differently from contract treewidth and starsize in that obtaining approximations might require a modification of the ontology. This is even true when the schema is full.

▶ **Example 24.** For $n \geq 3$, let $Q_n() = (\emptyset, \mathbf{S}_n, q_n \vee p_n)$ where $\mathbf{S}_n = \{W, R_1, \ldots, R_n\}$ with W of arity n and each R_i binary and where

$$q_n = \exists x_1 \cdots \exists x_n \, W(x_1, \ldots, x_n) \text{ and } p_n = \exists x_1 \cdots \exists x_n \exists y \, R_1(x_1, y), \ldots, R_n(x_n, y).$$

Then $Q'_n() = (\mathcal{O}, \mathbf{S}_n, p_n)$ with $\mathcal{O} = \{W(\bar{x}) \rightarrow p_n(\bar{x})\}$ is a TW_1-approximation of Q_n. In fact, it is equivalent to Q_n. However, Q_n has no TW_k-approximation Q^* based on the same (empty) ontology as Q_n for any $k < n$ since $Q'_n \not\subseteq Q^*$ for any $Q^* = (\emptyset, \mathbf{S}_n, q^*)$ such that $Q^* \subseteq Q$ and $q*$ is of treewidth $k < n$. In fact, any Q^* with the latter property does not return any answers on the database $\{W(a_1, \ldots, a_n)\}$.

In the appendix of the long version, we provide a further set of examples which does not require the arity of relation names to grow. It does, however, use a data schema that is not full. It remains an interesting and non-trival open problem to prove a counterpart of Corollary 23, even for the case of the full schema.

7 Conclusions

We have provided a complexity classification for counting the number of answers to UCQs in the presence of TGDs that applies both to ontology-mediated querying and to querying under constraints. The classification also applies to ontology-mediated querying with the OMQ language $(\mathcal{ELIH}, \text{UCQ})$ where $\mathcal{ELIH}$ is a well-known description logic [2]. In fact, this is immediate if the ontologies in OMQs are in a certain well-known normal form that avoids nesting of concepts [2]. In the general case, it suffices to observe that all our proofs extended from guarded TGDs to frontier-guarded TGDs [3] with bodies of bounded treewidth, a strict generalization of $\mathcal{ELIH}$. In contrast, a complexity classification for OMQs based on frontier-guarded TGDs with unrestricted bodies is an interesting problem for future work.

There are several other interesting questions that remain open, we mention only a few. In querying under constraints that are guarded TGDs, does answer counting in FPT coincide with answer counting in PTime? Do our results extend to ontology-mediated querying when the data schema is not required to be full? What about OMQs and CQSs based on other decidable classes of TGDs? And how can we decide the meta problems for the important structural measure of treewidth when the ontology needs not be preserved, with full data schema or even with unrestricted data schema?

References

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

2 Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. `doi:10.1017/9781139025355`.

3 Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011. `doi:10.1016/j.artint.2011.03.002`.

4 Pablo Barceló, Gerald Berger, and Andreas Pieris. Containment for rule-based ontology-mediated queries. In *PODS*, pages 267–279, 2018. `doi:10.1145/3196959.3196963`.

5 Pablo Barceló, Victor Dalmau, Cristina Feier, Carsten Lutz, and Andreas Pieris. The limits of efficiency for open- and closed-world query evaluation under guarded TGDs. In *Proc. of PODS*, 2020. doi:10.1145/3375395.3387653.

6 Pablo Barceló, Cristina Feier, Carsten Lutz, and Andreas Pieris. When is ontology-mediated querying efficient? In *Proc. of LICS*, pages 1–13, 2019. doi:10.1109/LICS.2019.8785823.

7 Pablo Barceló, Diego Figueira, Georg Gottlob, and Andreas Pieris. Semantic optimization of conjunctive queries. *J. ACM*, 67(6), 2020. doi:10.1145/3424908.

8 Pablo Barceló, Georg Gottlob, and Andreas Pieris. Semantic acyclicity under constraints. In *PODS*, pages 343–354, 2016. doi:10.1145/2902251.2902302.

9 Meghyn Bienvenu, Quentin Manière, and Michaël Thomazo. Answering counting queries over DL-Lite ontologies. In *Proc. of IJCAI*, 2020. doi:10.24963/ijcai.2020/223.

10 Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Proc. of Reasoning Web*, volume 9203 of *LNCS*, pages 218–307. Springer, 2015. doi:10.1007/978-3-319-21768-0_9.

11 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014. doi:10.1145/2661643.

12 Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013. doi:10.1613/jair.3873.

13 Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012. doi:10.1016/j.artint.2012.08.002.

14 Diego Calvanese, Julien Corman, Davide Lanti, and Simon Razniewski. Counting query answers over DL-Lite knowledge base. In *Proc. of IJCAI*, 2020. doi:10.24963/ijcai.2020/230.

15 Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS*, pages 149–158. ACM Press, 1998. doi:10.1145/275487.275504.

16 Hubie Chen and Stefan Mengel. A trichotomy in the complexity of counting answers to conjunctive queries. In *Proc. of ICDT*, pages 110–126, 2015. doi:10.4230/LIPIcs.ICDT.2015.110.

17 Hubie Chen and Stefan Mengel. Counting answers to existential positive queries: A complexity classification. In *Proc. of PODS*, pages 315–326, 2016. doi:10.1145/2902251.2902279.

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

19 Holger Dell, Marc Roth, and Philip Wellnitz. Counting answers to existential questions. In *Proc. of ICALP*, pages 113:1–113:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.113.

20 Arnaud Durand and Stefan Mengel. The complexity of weighted counting for acyclic conjunctive queries. *J. Comput. Syst. Sci.*, 80(1):277–296, 2014. doi:10.1016/j.jcss.2013.08.001.

21 Arnaud Durand and Stefan Mengel. Structural tractability of counting of solutions to conjunctive queries. *J. Theory Comput. Syst.*, 57(4):1202–1249, 2015. doi:10.1007/s00224-014-9543-y.

22 Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *J. Theor. Comput. Sci.*, 336(1):89–124, 2005. doi:10.1016/j.tcs.2004.10.033.

23 Cristina Feier, Carsten Lutz, and Marcin Przybyłko. Answer counting under guarded TGDs, 2021. arXiv:2101.03058.

24 Diego Figueira. Semantically acyclic conjunctive queries under functional dependencies. In *Proc. of LICS*, page 847–856. ACM, 2016. doi:10.1145/2933575.2933580.

25 Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.

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

27 David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984. doi:10.1016/0022-0000(84)90081-3.

28 Bogdan Kostov and Petr Kremen. Count distinct semantic queries over multiple linked datasets. *OJSW*, 5(1):1–11, 2018. URL: http://nbn-resolving.de/urn:nbn:de:101:1-201712245426.

29 Egor V. Kostylev and Juan L. Reutter. Complexity of answering counting aggregate queries over DL-Lite. *J. Web Semant.*, 33:94–111, 2015. doi:10.1016/j.websem.2015.05.003.

30 Nicola Leone, Marco Manna, Giorgio Terracina, and Pierfrancesco Veltri. Fast query answering over existential rules. *ACM Trans. Comput. Log.*, 20(2):12:1–12:48, 2019. doi:10.1145/3308448.

31 David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979. doi:10.1145/320107.320115.

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

33 Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 4900:133–173, 2008. doi:10.1007/978-3-540-77688-8_5.