Fine-Grained Complexity of Ontology Mediated Queries

Cristina Feier
Technical University of Cluj-Napoca, Romania

Abstract

This article surveys some approaches for establishing fine-grained complexity results for evaluation of ontology mediated queries (OMQs). It accompanies a related talk given at the Reasoning Web Summer School 2024. It zooms into some characterizations of efficiency in a parameterized complexity framework for OMQs based on various description logics and guarded tgds. As such results were established using results from query evaluation on databases, it also discusses the relevant results from the database world. After surveying some successive results on OMQs which all leverage database results in custom ways, it describes an approach which provides a general fpt reduction from query evaluation in the database world to query evaluation in the OMQ world. The reduction enables porting hardness results from the DB world to the OMQ world in a black-box fashion. Along these mentioned approaches, it also provides a brief survey of other approaches which are concerned with fine-grained complexity of OMQs and are based on rewriting techniques.

2012 ACM Subject Classification Theory of computation → Logic; Theory of computation → Automated reasoning; Theory of computation → Description logics; Theory of computation → Parameterized complexity and exact algorithms; Information systems → Query languages

Keywords and phrases complexity analysis, guarded logics, guarded tgds, database theory, ontology mediated queries

Digital Object Identifier 10.4230/OASIcs.RW.2024/2025.2

Category Invited Paper

Funding This work is supported by the project "Romanian Hub for Artificial Intelligence-HRIA", Smart Growth, Digitization and Financial Instruments Program, MySMIS no. 334906.

Acknowledgements I want to thank the anonymous reviewer for the very helpful comments.

1 Introduction

Ontologies are formal structured representations of knowledge which capture interdependencies between knowledge items by means of logical rules and/or axioms. The term stems from philosophy where it refers to defining frameworks of reality [33]: what kind of entities exist, how they can be grouped, and so on. In Computer Science, ontologies [30] became popular as part of the Semantic Web vision which viewed the web as "an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [9]. While nowadays large language models (LLMs) are ubiquitous and immensely useful in retrieving information, becoming a de facto interface between human and machine, they are often prone to hallucinations due to the imprecise/statistical-based inference model. Ontologies, with their precisely defined semantics and accompanying algorithms which come with correctness and/or completeness guarantees, do not have this drawback. However, reasoning with ontologies can be costly from a computational perspective. Also, statically generated ontologies suffer from a modeling bottleneck: the conclusions of any algorithm are only as useful/complete/relevant as the world representation is.

While still in their infancy, *neurosymbolic* approaches combine the connectionist approach with the classical logic-based one. Starting with semantic parsing approaches which translate natural language questions into formal queries based on statistical approaches like probabilistic CFGs[8] or retrieval augmented generation (RAG) [37] which sets the ground for LLMs for

accessing other types of knowledge oracles and continuing with more specialized approaches like *ontology grounded retrieval* (OG-RAG) [47] or neural-based semantic parsing [31, 49], such approaches are constantly evolving. Thus, even in the era of gnerative AI, there seems to be a need for formal reasoning, i.e. reasoning with ontologies.

Some of the key components when talking about ontologies is their representation language, but also the underlying data model. One of the most popular ontology languages, the W3C standard OWL [45], with its various incarnations (it is in fact hierarchy of languages stemming from simple expressivity to very high expressivity), is based on the well-known family of Description Logics (DL) [1]. These logics, which can be seen as counterparts of modal logic [2] and also descendants of frame systems [44], are binary fragments of FOL, designed to have different expressivity features while maintaining decidability for popular inference tasks. Being binary logics, the underlying data model can be seen as being graph-based. In fact ABoxes, the DL specific databases, are nothing more than what nowadays is referred to as "knowledge graphs".

Description Logics are not the only languages used to represent ontologies. While the graph-based data model is simple and amenable to network-specific navigational querying and analytics, it also comes with limitations due to the constraint on the arity of relations. Thus, there was a need to also identify nicely-behaved languages (in the sense of at least being decidable) where the schema is not restricted to being binary. Such a language is that of *guarded tgds* [13] which is a fragment of FOL existential rules (tgds) with an additional restriction (guardedness) meant to enforce decidability. Guarded tgds can have arbitrary arity schemas and their underlying data model can be seen as that of relational databases (which in the setting of logics is abstracted as relational structures).

An ontology together with a database compatible with its underlying data model and schema is referred to as a *knowledge base* (KB). KBs can be queried by various means. Some popular query languages are that of *atomic queries* (where one is interested whether a single fact is entailed by the KB) and that of *conjunctive queries* (which ask whether a certain conjunction of atoms is entailed by the KB). An *ontology-mediated query* (OMQ) is simply an ontology together with a data schema (not all symbols from the ontology might be allowed to occur in the data) and a query over that ontology. One also speaks about *OMQ languages* as being tuples $(\mathcal{L}, \mathcal{Q})$, where $\mathcal{L}$ is an ontology language and $\mathcal{Q}$ is a query language.

The expressivity of OMQ languages offers a very rich landscape ranging from NP-completeness (or even tractability) to 2ExpTime/undecidability. Some examples of low-complexity OMQ languages are those based on the lightweight languages underlying the OWL 2 profiles (EL, RL, QL): all these languages are tractable w.r.t. atomic queries and NP-complete w.r.t. conjunctive queries (similar complexity as answering CQs over DBs). At the other end of the spectrum, answering CQs over DLs extending $\mathcal{ALCI}$ is 2ExpTime -complete. The same can be said about answering CQs over unrestricted-arity schema guarded TGDs.

Given an expressive OMQ language, a relevant question is: which particular OMQs from that language can be evaluated efficiently? Are there criteria that characterize classes of OMQs that can be evaluated efficiently? Similar questions have been posed in the past for the evaluation of CQs over databases. They received positive answers in two settings: that of bounded-arity schemas [28], and that of unrestricted arity schemas [43, 17], respectively. In the first case, it has been shown that the cut-off criteria is bounded treewidth (under some common assumption in parameterized complexity theory), while in the second it is bounded submodular width (again under some commpn assumptions in paramaterized complexity theory). In both cases, classes of CQs which exhibit such properties are fixed-parameter

tractable (fpt), with the parameter being the CQ size, i.e. there are algorithms which answer such queries q over databases D in time $f(|q|)p(|D|)$, where f is an arbitrary function and p is some polynomial. An interesting fact is that for databases of bounded-arity schemas tractable evaluation coincides with fixed parameter tractability.

For highly-expressive OMQs, tractable polynomial evaluation is a rather hard to achieve desiderata as the size of the ontology also comes into play (unlike the DB case where only the size of the query and that of the database matter). As such, it makes sense to study the complexity of OMQs into a parameterized framework where the parameter is the ontology size. Typically, the size of the ontology is much smaller than that of data, and thus one can allow arbitrary growth in that factor as long as it does not interact with the size of the data.

In this work, we survey several approaches which establish criteria for fpt evaluation of OMQs where the parameter is the ontology size. These range from OMQ languages like (ELHI, CQ) [4] to OMQs based on guarded tgds in the bounded arity case [3] and in the unbounded arity case [21]. It turns out that structural measures for CQs like treewidth (TW) and submodular width (SMW) again play an important work for efficiency of evaluation. However, here the measures are modulated by the ontology, as its presence has the potential to decrease the measures by enforcing the existence of equivalent OMQs with smaller measures for CQs. While the first two papers mentioned above establish the results by adapting proofs used in the database setting for establishing hardness results, the third introduces a direct reduction from evaluating CQs over DBs to evaluating OMQs (with structural measures preservation). This enables porting some hardness complexity results from the DB area in a black-box fashion, including retrieving previously established results. We will zoom into the latter approach and provide some more details both in the talk and the lecture notes.

While looking at OMQs complexity from a parameterized perspective is a principled way to obtain fine-grained complexity characterizations, this is by no means the only lens one can use for this task. Another "classical" way to show that seemingly hard queries are easy to evaluate is based on *rewriting*: queries are rewritten in simpler ones, from languages with lower complexity. This has also been applied in the OMQ area: along these lines [38] introduced a correspondence between OMQs based on the Description Logic $\mathcal{ALCI}$ and CSPs. This has been extended to a correspondence [10] between the expressive language MDDLog and MMSNP, the logical counterpart of CSPs. Such correspondences make it possible to port complexity results (which again are based on rewritings) from the CSP area to the OMQ area.

The plan of the paper is follows: Section 2 introduces necessary prerequisites and definitions related to OMQs and their evaluation. Then, Section 3 makes a database theory excurse: it describes existing results from the database area. In Section 4 we discuss two fpt characterization for OMQs: one based on a DL, $\mathcal{ELHI}$, the other based on guarded tgds with bounded-arity schema. Section 5 looks at fpt evaluation of guarded tgds with unrestriced arity schemas and describes in detail the approach, including the previously mentioned reduction. Before concluding in Section 7, we will also provide a short survey on other approaches in Section 6.

2 Ontology Mediated Queries

This section introduces some basic notions about ontology languages, query languages, and ontology mediated queries and their evaluation. It starts with some preliminaries about relational structures and homomorphisms as these will be needed in many places later.

2.1 Relational Structures, Homomorphisms

A *schema* $\mathbf{S}$ is a finite set of relation symbols with associated arities. An $\mathbf{S}$-*fact* has the form $r(\mathbf{a})$, where $r \in \mathbf{S}$, and $\mathbf{a}$ is a tuple of constants of size the arity of r. An $\mathbf{S}$-*structure* A is a set of $\mathbf{S}$-facts. The domain of a structure A, $\mathsf{dom}(A)$, is the set of constants that occur in facts in A. Given a structure A and a subset $C \subseteq \mathsf{dom}(A)$, the *sub-structure of A induced by C*, $A|_C$, is the structure containing all facts $r(\mathbf{b}) \in A$ such that $\mathbf{b} \subseteq C$. *The product of two structures A and B*, $A \times B$, is a structure with domain $\mathsf{dom}(A) \times \mathsf{dom}(B)$ consisting of all facts of the form $r((a_1, b_1), \ldots, (a_n, b_n))$, where $r(a_1, \ldots, a_n) \in A$ and $r(b_1, \ldots, b_n) \in B$.

Given two structures A and B, a function $f : \mathsf{dom}(A) \to \mathsf{dom}(B)$ is said to be a *homomorphism* from A to B, if for every fact $r(\mathbf{a}) \in A$, there exists a fact $r(\mathbf{b}) \in B$ such that $h(\mathbf{a}) = \mathbf{b}$. The image of A in B under f, $f(A)$, is the set of facts of the form $r(f(\mathbf{a}))$ in B, where $r(\mathbf{a})$ is from A. When such a homomorphism exists we say that A *maps into* B, denoted $A \to B$.

▶ **Example 1.** The structure $I = \{r(x, y), s(y, z), t(z, x)\}$ maps into the structure $J = \{r(a, b), s(b, b), t(b, a)\}$ via the homomorphism $h : \{x, y, z\} \to \{a, b\}$ with $h(x) = a$ and $h(y) = h(z) = b$. In fact, J is the image of I under h.

2.2 Ontology Languages

We assume that the reader is familiar with First Order Logic (FOL). Most ontology languages are decidable subsets of FOL (possibly with additional or alternative syntaxes). Among them, we mention the family of binary logics known as Description Logics, languages based on rules such as Datalog or disjunctive Datalog, or on tuple generating dependencies (tgds) such as *guarded tgds*.

In the following we will provide a short introduction to some of these logics. We start with the definition of tuple generating dependencies (tgds): a tgd is a first order sentence of the form $\forall \mathbf{x} \forall \mathbf{y}\ \phi(\mathbf{x}, \mathbf{y}) \to \exists \mathbf{z}\ \psi(\mathbf{x}, \mathbf{z})$, with ϕ and ψ conjunctions of atoms having as terms only variables from $\mathbf{x} \cup \mathbf{y}$, and from $\mathbf{x} \cup \mathbf{z}$, respectively. Such a sentence will be abbreviated as $\phi(\mathbf{x}, \mathbf{y}) \to \exists \mathbf{z}\ \psi(\mathbf{x}, \mathbf{z})$. The atoms in ϕ and the atoms in ψ are referred to as *body atoms* and *head atoms*, resp.

A *guarded tgd* is a tgd which has a body atom having all universally quantified variables occurring in the tgd. The language of guarded tgds will be denoted as GTGD. By further restricting tgds to rules with universally quantified variables only, one obtains the language Datalog. When all rules in a Datalog program are guarded, we speak about *guarded Datalog (GDLog)*.

▶ **Example 2** (tgds and guarded Datalog rules)**.** The program bellow contains one guarded tgd (axiom (1)) and two guarded Datalog rules (axioms (2) and (3)). The atom witnessing guardedness is emphasized in each rule.

$$\mathsf{machine}(m) \wedge \mathsf{time}(t) \wedge \mathbf{malfunc}(\mathbf{m}, \mathbf{t}) \to \exists e\ \mathsf{error}(e) \wedge \mathsf{timestamp}(e, t) \quad (1)$$

$$\mathbf{stops}(\mathbf{m}, \mathbf{t}) \to \mathsf{malfunc}(m, t) \quad (2)$$

$$\mathbf{rotate}(\mathbf{m}, \mathbf{t}) \to \mathsf{malfunc}(m, t) \quad (3)$$

The extension of Datalog with disjunction in the head (i.e. ψ is a disjunction of atoms and not a conjunction) is the language of *disjunctive Datalog*. W.r.t. (disjunctive) Datalog, it is customary to call *EDB* those predicates which do not occur in the head of any rule and *IDB* all the other predicates. When all IDBs in a disjunctive Datalog program are monadic, one speaks of monadic disjunctive Datalog (MDDLog).

2.2.1 Query Languages

A *conjunctive query (CQ)* is a positive existential formula of the form $q(\mathbf{x}) = \exists \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$, with $\mathbf{x}$ and $\mathbf{y}$ tuples of variables and $\phi(\mathbf{x}, \mathbf{y})$ a conjunction of atoms having as terms the variables from $\mathbf{x} \cup \mathbf{y}$. The set $\mathbf{x}$ is the set of *answer variables* of q, while the set $\mathbf{y}$ is the set of *existential variables* of q. When $\mathbf{x}$ is empty, the CQ is said to be *Boolean* (BCQ). We denote with $\mathsf{var}(q)$ the set of variables of q and with $D[q]$ the *canonical database of* q, i.e. the set of atoms which occur in ϕ. We will sometimes use BCQs or their canonical databases interchangeably. A sub-query of a BCQ q is a BCQ p such that $D[p] \subseteq D[q]$.

A *union of conjunctive queries (UCQ)* is a formula of the form $q(\mathbf{x}) = q_1(\mathbf{x}) \vee \ldots q_n(\mathbf{x})$, where each $q_i(\mathbf{x})$ is a CQ, for $i \in [n]$. An *atomic query (AQ)* is a CQ in which ϕ contains a single atom.

2.2.2 OMQs, OMQ Languages and OMQ Evaluation

Ontologies enhance pre-existing data with implicit knowledge. Thus, we usually query an ontology together with some database (unless interested in terminological reasoning, i.e. reasoning which concerns strictly the ontology). For languages of arbitrary arity like Datalog or guarded TGDs we can assume that the database comes in the form of a relational structure. For Description Logics, data comes in the form of *assertions* stored into *an ABox*. An assertion is simply a fact of the form $C(a)$ or $r(a, b)$, where C is a concept and r is a role and ABoxes are simply a type of knowledge graphs or they can be views as as relational structures over a binary signature (which might be a subset of the signature of the TBox).

It is important to note that no matter the formalism, the schema of the data might not be identical with the schema of the ontology. It might be that the ontology invents new predicates which are not present in the data/allowed to occur. As we will see later, schema restrictions on data have the potential to influence the complexity of query evaluation. For this reason, the definition of OMQs contain the schema of the data as a first-class citizen.

▶ **Definition 3** (OMQ, OMQ language). *An* ontology mediated query (OMQ) Q *is a triple* $(\mathcal{O}, \mathbf{S}, q(\mathbf{x}))$*, where* $\mathcal{O}$ *is an ontology,* $\mathbf{S}$ *is a schema, and* $q(\mathbf{x})$ *is a query.*

An OMQ language is a tuple $(\mathcal{L}, \mathcal{Q})$ *where* $\mathcal{L}$ *is an ontology language and* $\mathcal{Q}$ *is a query language.*

When Q *is such that* $\mathcal{O}$ *is from* $\mathcal{L}$*,* q *is from* $\mathcal{Q}$*, we say that* Q *belongs to* $(\mathcal{L}, \mathcal{Q})$*. We also say that* Q *is an* $\mathbf{S}$-OMQ*.*

Given the ontology languages and the query languages we introduced above, any combinations of these induce an OMQ language. For example, ($\mathcal{ELHI}$,CQ) and ($GTGD, AQ$) are OMQ languages.

We are interested in evaluating $\mathbf{S}$-OMQs over $\mathbf{S}$-structures. That is, given such an OMQ $Q = (\mathcal{O}, \mathbf{S}, q)$ and an $\mathbf{S}$-structure D, is it the case that $\mathcal{O} \cup D \models q$? If yes, we will also write: $D \models Q$. We denote the evaluation problem for the OMQ Q as $\mathsf{eval}(Q)$. For classes of OMQs $\mathbf{Q}$, the evaluation problem is denoted as $\mathsf{eval}(\mathbf{Q})$. The evaluation problem is undecidable even for OMQs based on tgds and AQs [12]. However, entailment of OMQs based on query languages like AQs and CQs can be checked by constructing a (potentially infinite) canonical model called *chase* [42, 19, 35]. Intuitively, the chase completes the database D w.r.t. which the OMQ is evaluated with atoms obtained by triggering tgds and successively adding their consequences. We will denote with $\mathsf{ch}_{\mathcal{O}}(D)$ the chase of $\mathcal{O}$ w.r.t. D. Then, $D \models Q$ iff $\mathsf{ch}_{\mathcal{O}}(D) \models q$, or in other words iff $D[q] \to \mathsf{ch}_{\mathcal{O}}(D)$.

There are several variants of the chase; here we describe the *oblivious chase*. Let $(\mathsf{ch}_k(\mathcal{O}, D))_{k\geq 0}$ be a sequence of structures such that $\mathsf{ch}_0(\mathcal{O}, D) = D$. Then, for every $i > 0$, $\mathsf{ch}_i(\mathcal{O}, D)$ is obtained from $\mathsf{ch}_{i-1}(\mathcal{O}, D)$ by considering all homomorphisms h from the body of some tgd $\phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\psi(\mathbf{x}, \mathbf{z})$ in $\mathcal{O}$ to $\mathsf{ch}_{i-1}(\mathcal{O}, D)$ s.t. at least one atom from ϕ is mapped by h into a fact from $\mathsf{ch}_{i-1}(\mathcal{O}, D) \setminus \mathsf{ch}_{i-2}(\mathcal{O}, D)$, and adding to $\mathsf{ch}_i(\mathcal{O}, D)$ all facts obtained from atoms in $\psi(\mathbf{x}, \mathbf{z})$ by replacing each $x \in \mathbf{x}$ with $h(x)$ and each $z \in \mathbf{z}$ with some fresh constant. Then, $\mathsf{ch}_{\mathcal{O}}(D) = \bigcup_{k\geq 0} \mathsf{ch}_k(\mathcal{O}, D)$. Note that $\mathsf{ch}_{\mathcal{O}}(D)$ might be infinite. Still, for languages based on Datalog, including guarded Datalog, the chase is finite.

▶ **Example 4.** Let $\mathcal{O}$ be the GTGD program containing the guarded tgds from Example 2 and let $D = \{\mathsf{stops}(m, 10); \mathsf{rotate}(m, 20)\}$. Then:

- $\mathsf{ch}_0(\mathcal{O}, D) = D$
- $\mathsf{ch}_1(\mathcal{O}, D) = D \cup \{\mathsf{malfunc}(m, 10), \mathsf{malfunc}(m, 20)\}$
- $\mathsf{ch}_2(\mathcal{O}, D) = \mathsf{ch}_1(\mathcal{O}, D) \cup \{\mathsf{error}(e_1), \mathsf{timestamp}(e_1, 10), \mathsf{error}(e_2), \mathsf{timestamp}(e_2, 20)\}$
- $\mathsf{ch}_3(\mathcal{O}, D) = \mathsf{ch}_2(\mathcal{O}, D)$

Then: $\mathsf{chase}(\mathcal{O}, D) = \mathsf{ch}_2(\mathcal{O}, D)$

While tgds are in general undecidable, all the restrictions we mentioned previously are decidable. As customary, we distinguish between data complexity and combined complexity, where:

- *combined complexity* refers to the setting where both the OMQ and data are part of the input, while
- *data complexity* refers to the setting where the OMQ is fixed and only data is seen as an input.

These are some results which are already known about the complexity of evaluating various classes of OMQs:

- evaluating OMQs from $(\mathcal{ELHI}, AQ)$, $(\mathcal{ALC}, CQ)$ or (Datalog, CQ) is EXPTIME-complete in combined complexity
- evaluating OMQs from $(\mathcal{ALCI}, AQ)$ or (GTGD,CQ) is 2EXPTIME in combined complexity
- evaluating OMQs from $(\mathcal{ALC}, AQ)$ or $(MDDLog, CQ)$ is coNP-complete in data complexity
- evaluating OMQs from $(\mathcal{ELHI}, CQ)$ or $(MDLog, CQ)$ is PTIME-complete in data complexity

Additionally, we look at the OMQ evaluation problem from the perspective of parameterized complexity.

▶ **Definition 5** (Parameterized problem)**.** *For Σ some finite alphabet, a* parameterized problem *is a tuple (P, κ), where $P \subseteq \Sigma^*$ is a problem, and $\kappa : \Sigma^* \rightarrow \mathbb{N}$ is a* PTIME *computable function called the* parameterization *of P.*

The idea behind parameterized complexity is that the parameter can be safely ignored when assessing hardness of a problem as long as it does not interact with the rest of the problem. Thus, it comes with a relaxed notion of tractability defined as follows:

▶ **Definition 6.** *Fixed-parameter tractability A parameterized problem is* fixed-parameter tractable *if there exists an algorithm for deciding P for an input $x \in \Sigma^*$ in time $f(\kappa(x))poly(|x|)$, where f is a computable function and poly is a polynomial. The class of all fixed-parameter tractable problems is denoted as* FPT.

Downey and Fellows [20] established a hierarchy of parameterized complexity classes $\mathrm{W}[0] \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \ldots$, where $\mathrm{W}[0] = \mathrm{FPT}$ and each inclusion is believed to be strict.

As discussed in the introduction, when interested in efficient evaluation of OMQs, it makes sense to see the evaluation problem as a parameterized problem with parameter the size of the ontology. The parameterized evaluation problem for an OMQs Q with parameter the size of the ontology is denoted as p-eval(Q). As expected, the notation lifts to classes $\mathbf{Q}$ of OMQs: p-eval($\mathbf{Q}$). In general, the p-eval($\mathbf{Q}$) is W[1]-hard for any class of OMQs based on CQs.

3 A Database Theory Excurse

As mentioned in the Introduction, it has been a long-standing question when conjunctive queries can be efficiently evaluated on databases. For Boolean CQs this is the problem of checking whether the query maps homomorphically into the database. The same problem is also a presentation of so-called uniform Constraint Satisfaction Problems, except that in this setting the query is called a *template*. The problem is in general NP-complete[15]. One of the earliest results in this direction stems from Yannakakis [48] which states that acyclic conjunctive queries can be evaluated in polynomial time. An acyclic CQ is one whose underlying hypergraph can be seen as a tree. The algorithm uses a dynamical programming approach by evaluating the tree bottom-up and propagating the constraints.

This has been refined in successive works which established tractability results for queries of bounded treewidth [16], bounded (fractional) hypertreewidth [27, 29], and complete characterizations of fixed-paramater tractability in the bounded [28] and unbounded arity seting [43, 17]. Interestingly enough, for the bounded-arity case, tractability in a classical sense (polynomial time evaluation) coincides with fixed-parameter tractability. The tractability results rely on different structural measures pertaining to the CQ considered: treewidth, hypertreewidth, etc. Before stating the formal characterizations, we will give some preliminaries concerning such measures.

A *hypergraph* is a pair $H = (V, E)$ with V a set of *nodes* and $E \subseteq 2^V \setminus \{\emptyset\}$ a set of *hyperedges*. For a relational structure I, $H_I = (V, E)$, the hypergraph underlying I is such that $V = \mathsf{var}(q)$ and E contains a hyperedge $\mathbf{x}$ for every atom of the form $r(\mathbf{x})$ in I. For a CQ q, we denote with H_q the hypergraph underlying $D[q]$ (thus, technically $H_{D[q]}$).

▶ **Definition 7** (Tree decomposition). *Given a hypergraph H, a* tree decomposition *of H is a pair $\delta = (T_\delta, \chi)$, with $T_\delta = (V_\delta, E_\delta)$ a tree, and χ a labeling function $V_\delta \to 2^V$ such that:*

1. $\bigcup_{t \in V_\delta} \chi(t) = V$.
2. *If $e \in E$, then $e \subseteq \chi(t)$ 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.*

▶ **Definition 8** (Treewidth). *The* treewidth *of a hypergraph H, $\mathsf{TW}(H)$, is the smallest k such that there exists a tree decomposition (T_δ, χ) of H, with $T_\delta = (V_\delta, E_\delta)$, such that for every $t \in V_\delta$, $|\chi(t)| \leq k$.*

A function $f : 2^V \to \mathbb{R}_{\geq 0}$ is *submodular* if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$. It is *edge-dominated* if $f(e) \leq 1$ for all $e \in E$.

▶ **Definition 9** (Submodular width). *The* submodular width *of a hypergraph H, $\mathsf{SMW}(H)$, is the smallest k such that for every monotone submodular edge-dominated function f, for which $f(\emptyset) = 0$, there exists a tree decomposition (T_δ, χ) of H, with $T_\delta = (V_\delta, E_\delta)$, such that $f(\chi(t)) \leq k$ for all $t \in V_\delta$.*

The notions of treewidth (TW) and submodular width (SMW) extend naturally to relational structures and CQs via their hypergraph. Let X range over $\{\mathsf{TW}, \mathsf{SMW}\}$. For a CQ q, $\mathsf{X}(q) = \mathsf{X}(H_q)$, while for a UCQ q', $\mathsf{X}(q') = \mathsf{sup}_q$ is a CQ in $_{q'}$ $X(q)$. For classes of CQs/UCQs $\mathbb{Q}$, $\mathsf{X}(\mathbb{Q}) = \mathsf{sup}_{Q \in \mathbb{Q}}$ $(X(Q))$. Also, for a relational structure I: $\mathsf{X}(I) = \mathsf{X}(H_I)$ and for a class of relational structures $\mathbb{I}$, $\mathsf{X}(\mathbb{I}) = \mathsf{sup}_{I \in \mathbb{I}}$ $X(I)$. Then, a class of CQs/UCQs/relational structures has bounded (semantic) X-width if there exists a k such that every CQ/UCQ/structure from the class has X-width at most k.

We are now ready to provide the promised characterizations concerning the tractability results. Similar to the OMQ case, for a CQ q we introduce the notations eval(q) and p-eval(q) to denote the evaluation problem and the parameterized evaluation problem of q on arbitrary databases (where the parameter is the size of q). The notation extends to classes of CQs as expected.

We start with the bounded arity case:

▶ **Theorem 10** ([28]). *Assume that* FPT $\neq$ W[1]. *Then for every recursively enumerable class* $\mathbb{C}$ *of CQs the following are equivalent:*

1. *eval*$(\mathbb{C}, \_)$ *is in polynomial time.*
2. *p-eval*$(\mathbb{C}, \_)$ *is fixed-paramater tractable.*
3. $\mathbb{C}$ *has bounded semantic treewidth (TW).*

If either statement is false, then p-eval$(\mathbb{C}, \_)$ *is* W[1]*-hard.*

First, we notice that the theorem holds under the assumption that FPT $\neq$ W[1] which is a widely held assumption in parameterized complexity theory. Second, the criteria for tractability (both classical and parameterized) is "bounded semantic treewidth". Here semantic refers to bounded treewidth modulo equivalence, i.e. the semantic TW of a CQ q, sem-TW(q), is the minimum TW of some equivalent CQ: $\mathsf{min}_{q' \equiv q}\mathrm{TW}(q')$.

In the unbounded arity case Marx [43] established in a seminal result the border for fpt evaluation of uniform CSPs of the form p-CSP($\mathbb{A}$), where $\mathbb{A}$, the class of templates, is closed under underlying hypergraphs i.e. if H is the hypergraph corresponding to some structure $A \in \mathbb{A}$ all structures A' with underlying hypergraph H are part of $\mathbb{A}$. The restriction has been lifted in [17], yielding a full characterization for parameterized uniform CSPs of unrestricted arity. Both results are based on a widely held conjecture, the Exponential Time Hypothesis [34], and rely on a new structural measure, *submodular width*:

▶ **Theorem 11** (Theorem 1, [17]). *Let* $\mathbb{C}$ *be a r.e. class of CSPs. Assuming the Exponential Time Hypothesis, p-CSP*$(\mathbb{C})$ *is fixed-parameter tractable if and only if* $\mathbb{C}$ *has bounded semantic submodular width.*

As the (parameterized) evaluation problem for CSPs is just another incarnation of the (parameterized) evaluation problem of CQs on databases, a direct corollary of the above theorem is:

▶ **Corollary 12.** *Let* $\mathbb{C}$ *be a r.e. class of CQs. Assuming the Exponential Time Hypothesis, p-eval*$(\mathbb{C})$ *is fixed-parameter tractable if and only if* $\mathbb{C}$ *has bounded semantic submodular width.*

Again, the tractability border is characterized by a semantic structural measure, this time submodular width (SMW). Similar to the case for TW, the semantic SMW of a CQ is the minimum SMW of some equivalent CQ.

An interesting question, given the role of semantic structural measures, is for a given CQs what are the witnesses for such measures? That is, for a given measure X and a CQ q, how does one find the equivalent CQ q' such that sem-X(q) = X(q') where sem-X(q) is

the semantic X-measure for q? It has been established in [17] that such witnesses are the homomorphic *cores* of the original CQ. For readers unfamiliar with the notion of cores we next provide a definition and some explanations.

▶ **Definition 13** (Core). *The core [32] of a relational structure I is a sub-structure J of I which is homomorphically equivalent to I and it is subset minimal (w.r.t. the number of vertices and/or atoms). In general, a structure is said to be a core if it is the core of some relational structure.*

▶ **Example 14.** The structure $A = \{R(a,b), R(c,b), S(c,d), S(a,d), T(d,b)\}$ is not a core as its substructure $A' = \{R(a,b), S(a,d), T(d,b)\}$ is homomorphically equivalent. A maps into A' via a homomorphism h which is the identity on $\{a,b,d\}$ and which maps c into a.

However, $B = \{R(a,c), R(c,a), S(c,d), S(a,d)\}$ is a core. There is no substructure which maps into B. While there are homomorphisms from C to itself which are not the identity function (they either map a to c or vice versa) they are bijections and their image is C itself.

It is NP-complete to compute the core of a relational structure [11]. Cores have some important properties:

1. they are unique up to isomorphism
2. all the homomorphisms to itself (endomorphisms) are automorphisms (bijections)

4 FPT Characterization for OMQs Based on $\mathcal{ELHI}$ and Bounded-Arity Guarded TGDs

In this section we describe some results which establish fpt borders for evaluating OMQs from ($\mathcal{ELHI}_\bot$, UCQ) and (GTGD, UCQ) (bounded arity case). It turns out that structural measures like TW and SMW again play an important role in characterizations. To this end, we lift such measures from CQs/UCQs to OMQs based on CQs/UCQs: given such a measure X for an OMQ $Q = (\mathcal{O}, \mathbf{S}, q)$, with q a UCQ, $\mathsf{X}(Q) = \mathsf{X}(q)$. Again, semantic measures are defined via equivalence, this time with an OMQ: $\mathsf{sem}-\mathsf{X}(Q) = \mathsf{min}_{Q' \equiv Q}\ \mathsf{X}(Q')$, i.e. the semantic X-width of some OMQ Q is the minimum X-width over all equivalent OMQs Q'. The notations lift to classes of OMQs $\mathbb{Q}$ as expected: $\mathsf{X}(\mathbb{Q}) = \mathsf{sup}_{Q \in \mathbb{Q}}\ X(Q)$ and $\mathsf{sem}-\mathsf{X}(\mathbb{Q}) = \mathsf{sup}_{Q \in \mathbb{Q}}\ \mathsf{sem}-\mathsf{X}(Q)$. With these notations in place, we say that a class of OMQs $\mathbb{Q}$ has bounded (semantic) X-width if there exists a $k > 0$ such that each OMQ from Q has (semantic) X-width at most k.

We start with some results concerning OMQs from ($\mathcal{ELHI}_\bot$, UCQ):

▶ **Theorem 15** (Theorem 4 [4]). *For any recursively enumerable class of OMQs $\mathbf{Q} \subseteq (\mathcal{ELHI}_\bot, UCQ)$, the following are equivalent, unless* FPT = W[1]*:*

1. *p-*$\mathsf{eval}(\mathbf{Q})$ *is in* FPT*;*
2. $\mathbf{Q}$ *has bounded semantic treewidth*

If either statement is false, p-$\mathsf{eval}(\mathbf{Q})$ *is* W[1]*-hard.*

As can be seen from the theorem above, the result nicely generalizes the result for the DB case. The most challenging part in establishing the result is the lower bound, i.e. the hardness result. To this end, the authors use an fpt-reduction from the k-clique problem. The problem when parameterized by k is a standard problem in parameterized complexity theory which is known to be $W[1]$-hard. An fpt-reduction, whose definition is given below, is a type of reduction under which each class from the Downey and Fellows hierarchy is closed, and thus they are employed to show parameterized hardness results:

▶ **Definition 16** (Parameterized reduction). *Given two parameterized problems* (P_1, κ_1) *and* (P_2, κ_2) *over alphabets* Σ_1 *and* Σ_2*, an* fpt-reduction *from* (P_1, κ_1) *to* (P_2, κ_2) *is a function* $R : \Sigma_1^* \to \Sigma_2^*$ *with the following properties:*

1. $x \in P_1$ *iff* $R(x) \in P_2$*, for every* $x \in \Sigma_1^*$*,*
2. *there exists a computable function* f *and a polynomial function* p *such that* $R(x)$ *is computable in time* $f(\kappa_1(x))p(|x|)$*,*
3. *there exists a computable function* g *such that* $\kappa_2(R(x)) \leq g(\kappa_1(x))$*, for all* $x \in \Sigma_1^*$*.*

The reduction from [4] adapts a similar reduction from the k-clique problem which was used to show the hardness result in the database case [28]. However, the lifting of the constructions from [28] is very laborious and it involves full introspection into the proof from [28]. A central concept in the proof is that of injective homomorphisms between query contractions and chase of databases. In order for the proof to go through it is necessary to identify databases that entail the OMQ but at the same time are weak enough in the homomorphism order to admit injective only homomorphisms from some query contractions. This concept will also be used in the approach in the next section and be made more precise there.

Another result from [4] looks at a fragment of $\mathcal{ELHI}_\bot$, $\mathcal{ELH}_\bot^{dr}$, and shows that in the absence of inverse roles and for *full schemas* (schemas which allow all symbols from the ontology to occur in the data), tractability and fpt actually coincide, similarly to the case of databases:

▶ **Theorem 17.** *For any recursively enumerable class of OMQs* $\mathbf{Q} \subseteq (\mathcal{ELH}_\bot^{dr}, \mathit{UCQ})$ *based on the full schema, the following are equivalent, unless* FPT = W[1]*:*

1. $\mathsf{eval}(\mathbf{Q})$ *is in* PTime *combined complexity;*
2. $p\text{-}\mathsf{eval}(\mathbf{Q})$ *is in* FPT*;*
3. $\mathbf{Q}$ *has semantic bounded treewidth.*

If either statement is false, $\mathsf{p\text{-}eval}(\mathbf{Q})$ *is* W[1]*-hard.*

Moving on to guarded tgds over bounder-arity schemas, a similar characterization as for $\mathcal{ELHI}_\bot$ holds:

▶ **Theorem 18** (Theorem 5.3 [3]). *Let* $\mathbb{Q}$ *be a r.e. class of OMQs from (GTGD, UCQ) over bounded arity schemas. Assumming* FPT ≠ W[1]*, the following are equivalent :*

1. $p\text{-}\mathsf{eval}(\mathbb{Q})$ *is fixed-paramater tractable.*
2. $\mathbb{Q}$ *has bounded semantic treewidth.*

If either statement is false, then p-OMQ$(\mathbb{Q})$ *is* W[1]*-hard.*

Establishing the hardness result required a careful adaptation of the constructions from [28]. For the upper bound, the paper shows a bounded TW OMQ from (GTGD, UCQ) can be be evaluated in OMQ by means of rewriting into the language (L, UCQ) (this includes database rewriting), where L is the language of linear TGDs. Evaluating OMQs from (L, UCQ) can be done by computing a finite bounded portion of the chase which can be done in fpt and then it follows from database results that the overall evaluation is in fpt given the bounded TW of the query.

5 FPT Characterization for OMQs based on Unbounded-Arity Guarded TGDs

In this section we describe another fpt characterization for evaluation of OMQs from (GTGD, UCQ), this time in the setting of unrestricted schema arity. As we have seen in the previous section, the characterization in the bounded-arity setting generalizes the one from the

database world, both being based on bounded semantic TW, of classes of OMQs, and of classes of CQs, resp. Similarly, the characterization descrined in this section generalizes the one from the database world, both characterizations being based on bounded semantic SMW. The characterization is as follows:

▶ **Theorem 19** ([21]). *Let $\mathbb{Q}$ be a r.e. class of OMQs from $(\boldsymbol{GTGD}, UCQ)$. Under the Exponential Time Hypothesis, $\mathbb{Q}$ has bounded semantic submodular width iff p-OMQ($\mathbb{Q}$) is fpt. If either statement is false, then p-OMQ($\mathbb{Q}$) is* W[1]*-hard.*

In the following, we will go into detail into how the result has been established, particularly regarding the lower bound, as this introduces an fpt reduction from query evaluation on DBs to OMQ evaluation which makes it possible to lift hardness results from the DB setting into the OMQ one. Before launching into the proof, we discuss some issues related to witnesses of bounded semantic measures (Section 5.1). Then, we provide some intuition and a roadmap in Section 5.2. Section 5.3 introduces the main ingredients of the proof – the notions of query initial databases, guarded unravelings and diversifications. Proof sketches for two characterizations, one for (GDLog, UCQ), the other for (GTGD, UCQs) are provided in Sections 5.4 and 5.5. Then, we wrap-up in Section 5.6.

5.1 Defining Cores for OMQs? The Issue with Structural Measures

As discussed in Section 3, the witnesses for semantic X-width of some CQ q, for $X \in \{\mathsf{TW}, \mathsf{SMW}\}$ are nothing else than the homomorphic cores of q. In an attempt to establish such witnesses for semantic measures for OMQs, one would be tempted to simply consider cores of CQs/UCQs which occur in the OMQ. However, as the following examples show, this is problematic. The ontology can actually lower the semantic measure, both in the case of TW, and in the case of SMW:

▶ **Example 20.** Let $Q = (\mathcal{O}, \mathbf{S}, q)$ be an OMQ with $\mathcal{O} = \{R(x, y, z, \dots) \to R(x, x, x, \dots)\}$. Then for any R-only CQ q (of arbitrary treewidth):

$$Q \equiv (\mathcal{O}, \mathbf{S}, \exists x\ R(x, x, \dots))$$

Why is this the case? Because the following identity holds:

$$Q \equiv (\mathcal{O}, \mathbf{S}, \mathsf{chase}_{\mathcal{O}}(D_q)) \equiv (\mathcal{O}, \mathbf{S}, \mathsf{core}(\mathsf{chase}_{\mathcal{O}}(D_q)), \quad (\dagger)$$

and $\mathsf{core}(\mathsf{chase}_{\mathcal{O}}(D_q)) = \{R(x, x, \dots)\}$. Note that in (†) we slightly abused notation as we regard $\mathsf{chase}_{\mathcal{O}}(D_q)$ and $\mathsf{core}(\mathsf{chase}_{\mathcal{O}}(D_q))$ as Boolean CQs. Thus, by chasing the query and considering the core (when feasible, if the chase is finite) we obtain a much simpler query, in this case $\exists x\ R(x, x, \dots)$.

The previous example raises the question: what if we consider the core of the chase of the query, $\mathsf{core}(\mathsf{chase}_{\mathcal{O}}(D_q))$ (when finite)? Again, this is not satisfactory:

▶ **Example 21.** Let $Q = (\mathcal{O}, \mathbf{S}, q)$ with:

$$\mathcal{O} = \{R(x, y) \to S(x, y), R(x, y) \to T(x, y)\};$$
$$\mathbf{S} = \{R\};$$
$$q = \exists x, y, z, u\ S(y, x) \wedge T(y, z) \wedge T(u, z) \wedge T(u, x)$$

Then, q is a homomorphic core of TW 2 ($\mathsf{TW}(H_q) = 2$) and thus, $\mathsf{TW}(Q) = 2$. It is also the case that $\mathsf{core}(\mathsf{chase}_{\mathcal{O}}(D_q)) = D_q$, the strategy mentioned above does not decrease TW.

Is the semantic TW of Q 2? No, it is 1! Let's consider the following OMQ: $Q' = (\mathcal{O}, \mathbf{S}, q')$ with $q' = \exists x, y\ R(x, y)$. Then, $Q \equiv Q'$. To see why this is the case, one has to pay attention to the data schema $\mathbf{S}$; any database which is queried by Q will contain only R atoms; thus, the only way to obtain the S-atom and the T-atom from the query is to actually apply the rules from $\mathcal{O}$. But then, q can be rewritten into $q'' = \exists x, y, z, u\ R(y, x) \wedge R(y, z) \wedge R(u, z) \wedge R(u, x)$ which is not a core and whose core is q'.

The example above is one of the cases hinted at in Section 2 where the schema restriction in conjunction with the ontology dilute the hardness of an OMQ, in this case it decreases its TW. We next show a similar phenomenon, but for the other structural measure, SMW.

▶ **Example 22.** For every $i \in \mathbb{N}$ with $i > 0$, let:

$$\psi_i^R(\mathbf{x}_i) = R(x_1, x_2) \wedge R(x_1, x_3) \wedge \cdots \wedge R(x_{i-1}, x_i)$$

It is implicit in the formula above that $\mathbf{x}_i = (x_1, x_2, \ldots x_i)$. Also, let $Q_i = (\mathcal{O}_i, \mathbf{S}_i, q_i)$ with:

$$\mathcal{O}_i = \{S_i(\mathbf{x}_i) \to \psi_i^R(\mathbf{x}_i)\}$$
$$\mathbf{S}_i = \{S_i, T\};$$
$$q_i = \exists \mathbf{x}_i\ \psi_i^R(\mathbf{x}_i) \wedge \psi_i^T(\mathbf{x}_i)$$

Then, the class of OMQs $\mathbb{Q} = (Q_i)_{i \geq 1}$ has unbounded SMW. To see why this is the case, let's consider the function $f(X) = |X|/2$ which is monotone sub-modular and edge-dominated w.r.t. H_{q_i}; then $f(\mathbf{x}_i) = i/2$. Thus, $\mathsf{SMW}(Q_i) \geq i/2$.

At the same time, there exists a class of OMQs $\mathbb{Q}'$ such that $\mathbb{Q}' \equiv \mathbb{Q}$ and such that $\mathsf{SMW}(\mathbb{Q}') = 1$, and thus $\mathsf{sem} - \mathsf{SMW}(\mathbb{Q}) = 1$ as well. We will next construct the class $\mathbb{Q}'$.

For every $i \in \mathbb{N}$, with $i > 1$, let $Q_i' = (\mathcal{O}_i, \mathbf{S}_i, q_i')$ with:

$$q_i' = \exists \mathbf{x}_i\ S_i(\mathbf{x}_i) \wedge \psi_i^R(\mathbf{x}_i) \wedge \psi_i^T(\mathbf{x}_i)$$

It is the case that $\mathsf{SMW}(Q_i') = \mathsf{SMW}(q_i') = 1$ as q_i' is "guarded" by $S_i(\mathbf{x}_i)$: every function f which is edge dominated will have the property that $f(\mathbf{x}_i) \leq 1$ and thus its cost for all of q_i' cannot be higher.

5.2 Intuition and Roadmap for the Characterization

As we have seen in the previous section, establishing cores of OMQs (with the hope of finding witnesses for bounded SMW) is not at all straightforward. Still, a useful observation is that sometimes adding atoms to the CQs in an OMQ has the effect to decrease its semantic SMW. This is exactly the phenomenon described in Example 22. In the following, we will use this strategy to obtain such witnesses of bounded SMW.

Where from can such atoms be added to CQs such that they decrease SMW and at the same time OMQ equivalence is preserved? In the example discussed above, and in general, atoms will be added from DBs which entail the OMQ. As CQs are specialized during this process, in order to maintain equivalence, multiple atoms from different DBs may be needed to be added: a cut in the space of the homorphism order of DBs which entail the OMQ has to be considered for such CQ adornments. At the same time based on previous experience with establishing hardness results for OMQs (as described in the previous section), it was known that injectivity (of mapping CQs into the chase of a database) plays a very important role, and that, to this end, databases which are small w.r.t. homomorphism order are desirable. Thus, it makes sense to look at DBs which entail the OMQ and which are as small as possible

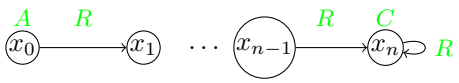
in the homomorphism order. A natural question is whether there are DBs which entail the OMQ and are minimal w.r.t. the homomorphism order (is the homomorphism order over DBs which entail the OMQ well-founded)? As next example shows, the answer is negative.

▶ **Example 23** (No hom-minimal DBs which entail an OMQ). Let $Q = (\mathcal{O}, \mathbf{S}, q)$ with:

$$\mathcal{O} = \{A(x) \wedge R(x,y) \to A(y),\ A(x) \to B(x)\}$$
$$\mathbf{S} = \{A, R, C\}$$
$$q = \exists x\ B(x) \wedge R(x,x) \wedge C(x)$$

Also, for every n let:

$$D_n = \{A(x_0), C(x_n), R(x_n, x_n)\} \cup \{R(x_i, x_{i+1}) \mid 0 \leq i < n\}$$



Then, for every $n \in \mathbf{N}$: $D_n \models Q$, D_n is a core, and $D_{n+1} \to D_n$. Also, for every $\mathbf{S}$-database D: $D \models Q \Rightarrow \exists n \in \mathbf{N} :\ D_n \to D$. Thus, D is not minimal w.r.t. $\to$.

Again, schema restrictions play a role in the fact that there are no minimal DBs for Q. In the case of *full* database schema: $D[q] = \{B(x), R(x,x), C(x)\}$ is a minimal database w.r.t. $\to$ which entails Q.

Still, as it will be shown in next sections, for OMQs from (GDLog, UCQ) it is possible to identify a finite bounded set of databases which can extended in a principled way to DBs that entail the OMQ. We will call this *characteristic databases*. Atoms from these characteristic DBs will be added to CQs in the original OMQ to obtain witnesses for bounded SMW, called *covers*. Covers are not optimal witnesses in the sense of always exhibiting optimal SMW as cores do in the case of CQs, but they are good enough, i.e. they preserve boundedness of the measure. Based on these notions, for OMQs from (GDLog, UCQ) one obtains the following characterization:

▶ **Theorem 24** ([21]). *For $\mathbb{Q}$, being a r.e. enumerable class of OMQs from (GDLog, UCQ), under the ETH, the following are equivalent:*

1. *p-OMQ($\mathbb{Q}$) is fixed-parameter tractable*
2. *$\mathbb{Q}_c$, the class of covers of OMQs from $\mathbb{Q}$, has bounded SMW*
3. *$\mathbb{D}_{\mathbb{Q}}$, the class of characteristic DBs associated to OMQs from $\mathbb{Q}$, has bounded SMW.*

Finally, to obtain the characterization for OMQs from (GTGD, UCQ), one exploits the fact that every such OMQ can be rewritten into an OMQ from (GDLog, UCQ) and the result above.

5.3 Query Initial Databases, Unravelings, Diversifications

As explained earlier, databases which are small in the homomorphism order and entail the OMQ are desirable for constructions as they are more likely to admit injective homomorphisms from CQs in the OMQ. This section makes a first step towards identifying such databases. A *contraction* of a CQ q is a CQ q' obtained from q by variable identification.

▶ **Definition 25** (Query initial DBs). *For $Q = (\mathcal{O}, \mathbf{S}, q) \in$ (GDLog, UCQ) and D an $\mathbf{S}$-database s.t. $D \models Q$, we say that D is query-initial (qi) w.r.t. Q if: for every $\mathbf{S}$-database D' such that $D' \to D$ and $D' \models Q$, and every contraction p of a CQ in q: $p \to \mathsf{ch}_{\mathcal{O}}(D')$ iff $p \to \mathsf{ch}_{\mathcal{O}}(D)$.*

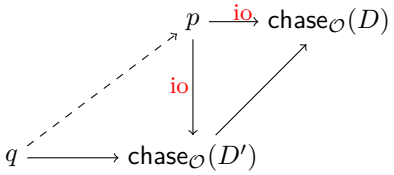


Figure 1 D is a qi DB: if $p \xrightarrow{io} \mathsf{ch}_{\mathcal{O}}(D')$ then $p \xrightarrow{io} \mathsf{ch}_{\mathcal{O}}(D)$.

To capture the connection of qi databases with injective homomorphisms we introduce the notion of *injectively only* mapping (io): a CQ/structure A maps io into another structure B, written $A \xrightarrow{io} B$, if $A \to B$ and every homomorphism from A to B is injective. Then the following holds:

▶ **Lemma 26.** *For every qi database D and pre-database D' such that $D' \models Q$, and every contraction p of a CQ q such that $p \xrightarrow{io} \mathsf{ch}_{\mathcal{O}}(D')$, it is the case that $p \xrightarrow{io} \mathsf{ch}_{\mathcal{O}}(D)$.*

Figure 1 depicts in a schematic way Definition 25 and the connection of such DBs with injectively only (i.o.) homomorphisms. Intuitively, the lemma holds, as from a non-injective homomorphism from p to $\mathsf{ch}_{\mathcal{O}}(D)$ one could identify a contraction of q (and p) which maps into $\mathsf{ch}_{\mathcal{O}}(D)$, but not into $\mathsf{ch}_{\mathcal{O}}(D')$. Note that this property of qi DBs also forces some injectivity between a part of D' and D (between the images of contractions which map io into both).

Another property of qi DBs is as follows:

▶ **Lemma 27.** *If D is qi w.r.t. Q, then for every $D' \to D$ s.t. $D' \models Q$, D' is qi w.r.t. Q.*

▶ **Example 28.** Let $Q = (\mathcal{O}, \mathbf{S}, q)$ with:

$$\mathcal{O} = \{W(x,y,z) \to S(y,z), U(x,y,z) \wedge V(x,z) \to T(x,z)\}$$
$$\mathbf{S} = \{R, U, V, W\}$$
$$q = \exists x, y, z \; R(x,y) \wedge S(y,z) \wedge T(z,x)$$

The, $q' = \exists x, y \; R(x,y) \wedge S(y,x) \wedge T(x,x)$ is a contraction of q. Also, let:

- $D_1 = \{R(a,b), W(d,b,a), U(a,d,a), V(a,a)\}$ and
- $D_2 = \{R(a,b), W(d,b,c), U(c,d,a), V(c,a)\}$ be two $\mathbf{S}$-databases.

It can be seen that $D_2 \to D_1$. It is also the case that:

- $\mathsf{ch}_{\mathcal{O}}(D_1) = D_1 \cup \{S(b,a), T(a,a)\}$, and
- $\mathsf{ch}_{\mathcal{O}}(D_2) = D_2 \cup \{S(b,c), T(c,a)\}$

We have that:

- $q, q' \to \mathsf{ch}_{\mathcal{O}}(D_1)$, and
- $q \to \mathsf{ch}_{\mathcal{O}}(D_2)$, but $q' \not\to \mathsf{ch}_{\mathcal{O}}(D_2)$

Thus, D_1 is not qi w.r.t. Q: D_2 and q' are counter-witnesses. But D_2 is!

Query initial databases can be seen as containing some kind of kernels (the images of contractions which map i.o. into their chase) together with facts which are needed just to satisfy individual atoms from queries and possibly irrelevant facts. The so-called kernels are relevant more from a topological perspective, they need to preserve a certain topology for the underlying hypergraph of the query to be embeddable into the chase of the database.

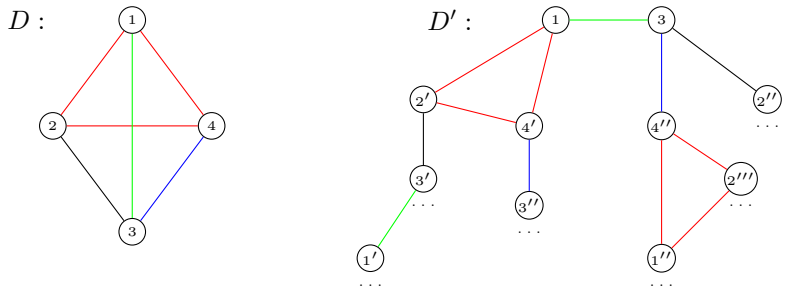


Figure 2 Constructing the guarded unraveling of D at $(1, 3)$.

As concerns individual atoms in the query and their satisfaction, it is enough to consider databases which have a particular shape, namely *guarded tree decompositions* – a tree decomposition where each bag is guarded by some atom. Thus, any sub-database needed to satisfy a particular atom can be unravelled safely into such a guarded decomposition and will still satisfy the atom. The operation which does this is called *guarded unravelling*. As its purpose is to preserve entailment of specific atoms, it is always applied w.r.t. guarded sets in database (a guarded set is a set of constants which occurs in a given atom). For a database D, and a guarded set $\mathbf{a}$ in D, the *guarded unraveling of D at $\mathbf{a}$*, is a database $D^{\mathbf{a}}$, which has the following properties:

1. for every OMQ Q' of the form $(\mathcal{O}, \mathbf{S}, p(\mathbf{x}))$, where $p(\mathbf{x})$ is an atomic query and every tuple $\mathbf{a}' \subseteq \mathbf{a}$: $D \models Q'(\mathbf{a}')$ implies $D^{\mathbf{a}} \models Q'(\mathbf{a}')$.
2. $\mathsf{SMW}(D^{\mathbf{a}}) = 1$ (due to the existence of a guarded tree decomposition)

We leave out the details of the construction as it is rather technical and at the same time straightforward: it unfolds the original database by creating copies of constants into a structure which admits a guarded tree decomposition. It achieves this by actually creating copies of guarded sets (to get a new node of the composition) and by adding the copies of the corresponding atoms over the original guarded set (the get the bag attached to that node). For further details, The interested reader can consult [21].

▶ **Example 29.** Let D be the database: $\{\mathsf{red}(1, 2, 4), \mathsf{green}(1, 3), \mathsf{blue}(4, 3), \mathsf{black}(2, 3)\}$. The database is pictured in Figure 2, where each atom is depicted by its homonym color. The left hand side of the figure depicts a part of the database $D_0 = D^{(1,3)}$, the guarded unraveling of D at $(1, 3)$ (the first steps of the unraveling construction). Note that for this particular case each bag of the guarded tree decomposition will contain exactly one atom.

How are unravelings used? For a given OMQ $Q = (\mathcal{O}, \mathbf{S}, q)$ from (GDLog, CQ) and $\mathbf{S}$-database D s.t. $D \models Q$, we will replace some parts of D with corresponding guarded unravelings. We will keep unchanged only the part of D needed to entail the (topological) structure of the query which cannot be mapped into guarded unravelings. This leads us to next notion, that of *diversification*.

For a database D, a constant $c \in \mathsf{dom}(D)$ is *isolated* if it occurs in a single fact in D. We denote with $\mathsf{ker}(D)$ the set of non-isolated constants in D.

▶ **Definition 30** (diversification). *A pair $(D, \uparrow)$ is a* diversification *of a database D_0 (written $D \preceq D_0$) if $\uparrow$ is a homomorphism from D to D_0 for which:*

- $\uparrow_{|\mathsf{ker}(D)}$ *is injective;*
- $\uparrow_{|\mathbf{a}}$ *is injective, for every guarded set $\mathbf{a}$ in D (denoted i.g.s.)*

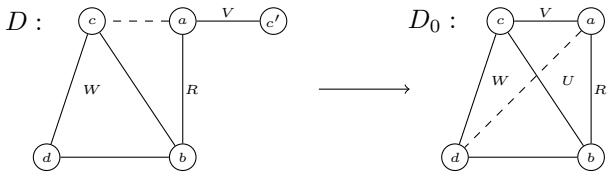


Figure 3 D is a diversification of D_0.

Intuitively, the diversification operation drops some atoms from a database and it does a one-step unfolding as concerns other atoms (i.e it allows the renaming of some constants occurring in atoms if the new constants occur only in a single atom); in other words, it diversifies a DB by making it more sparse.

▶ **Example 31.** Figure 3 depicts two databases $D = \{R(a,b), W(d,b,c), V(c',a)\}$ and $D_0 = \{R(a,b), W(d,b,c), V(c,a), U(c,d,a)\}$ together with a homomorphism $\uparrow\colon \mathsf{dom}(D) \to \mathsf{dom}(D_0)$ which is the identity on $\{a,b,c,d\}$ and such that $\uparrow(c') = c$.

As $\mathsf{ker}(D) = \{a,b\}$ and $\uparrow$ is injective on guarded sets, it follows that $(D,\uparrow)$ is a diversification of D_0: $D \preceq D_0$.

In this case, D diversifies D_0 by dropping the U-atom and by renaming c in $V(c,a)$ as c'.

In order to get databases as small as possible w.r.t. the homomorphism order, starting with any arbitrary database D_0 such that $D_0 \models Q$, we dilute D_0 as much as possible by combining the diversification operation with the addition of guarded unravelings. Intuitively:

- diversification preserves (some) structure (in the sense of underlying skeleton, hypergraph)
- guarded unravelings preserve atomic consequences

Databases obtained from the combination of the two operations are called *extended databases*.

▶ **Definition 32** (extended database). *For $(D,\uparrow)$ a diversification of D_0, $\mathsf{ext}(D,\uparrow,D_0)$ is the database obtained from D by adding to D for every guarded set $\mathbf{a}$ in D the database $D_0^{\uparrow(\mathbf{a})}$, where $\uparrow(\mathbf{a})$ is renamed as $\mathbf{a}$.*

The procedure is depicted in Figure 4.

An important observation is that given an injective homomorphism h from q to $\mathsf{chase}_{\mathcal{O}}(D_0)$, it is possible to construct a diversification (D,h') of D_0, where $D \approx D_0 \cup h(q)$, and $h' \approx h$, by considering the h-image of q in D_0. We denote with $\mathsf{div}(D_0,Q)$, the set of diversifications $(D,\uparrow)$ of D_0 for which $\mathsf{ext}(D,\uparrow,D_0) \models Q$.

As we want to find weak databases, i.e. small in the homomorphism order we introduce the notion of minimality for diversifications.

▶ **Definition 33** (Minimal diversification). *A diversification $(D,\uparrow)$ of D_0 is* minimal w.r.t. *Q if D is a core and there is no diversification $(D',\downarrow)$ in $\mathsf{div}(D_0,Q)$ s.t. $D' \preceq D$, and $D \not\preceq D'$.*

The set of all minimal diversifications of D_0 w.r.t. Q is denoted $\mathsf{mdiv}(D_0,Q)$.

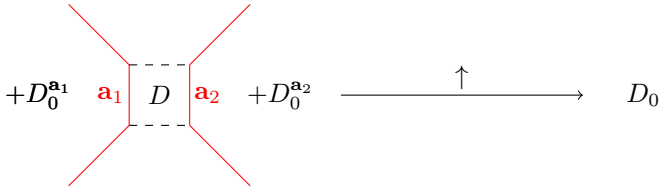


Figure 4 Constructing an extended database from D_0 using diversification $(D,\uparrow)$.

Finally:

▶ **Definition 34** (characteristic DBs). *For $Q \in (GDLog, UCQ)$, the set of characteristic databases for Q, $\mathbb{D}_Q$, is the set $\{D \mid (D, \uparrow) \in \mathsf{mdiv}(D_0, Q), D_0$ is qi w.r.t. $Q\}$*

Thus, characteristic databases are exactly those DBs which occur in minimal diversifications. We also introduce the notion of extended characteristic databases as those databases that entail the OMQ and which can be reconstructed from minimal diversifications:

▶ **Definition 35** (extended characteristic DBs). *For $Q \in (GDLog, UCQ)$, the set of extended characteristic databases for Q, $\mathbb{D}^+_Q$ is the set*

$$\{\mathsf{ext}(D, \uparrow, D_0) \mid (D, \uparrow) \in \mathsf{mdiv}(D_0, Q), D_0 \text{ is qi w.r.t. } Q\}$$

As promised earlier, we use the set of characteristic databases (which are basically subsets of databases which entail the OMQ) to add atoms to CQs in OMQs with the hope of obtaining OMQs with lower SMW. Such OMQs are called *covers*.

▶ **Definition 36** (Covers). *For an OMQ $Q = (\mathcal{O}, \mathbf{S}, q)$, we denote with Q_c a new OMQ $(\mathcal{O}, \mathbf{S}, q_c)$, where q_c is the union of all CQs p_c of the form $h(p) \cup S$, where:*

1. *$h(p)$ is the image of some CQ p from q into $\mathsf{ch}_{\mathcal{O}}(D^+)$, for D^+ an extended characteristic database of Q of the form $\mathsf{ext}(D, \uparrow, D_0)$ $(D \subseteq D^+)$*
2. *S is a minimal set of atoms such that:*
   a. *$D \subseteq S \subseteq D^+$;*
   b. *for every atom $r(\mathbf{a})$ from $h(p)$, there exists an atom $r'(\mathbf{a}')$ from S such that $\mathbf{a} \subseteq \mathbf{a}'$ and $\mathbf{a}'$ is a maximal guarded set in D^+.*

Q_c is said to be the cover of Q.

Note that in the definition above, S is the set of added atoms. It "covers" every atom from $h(p)$. Some important properties of covers are that they are equivalent to the original OMQs and that they are computable (this is shown in [21] via a Guarded Second Order encoding)

5.4 Proof sketches for characterization for OMQs from (GDLog, UCQ)

We recall the characterization for OMQs based on (GDLog, UCQ):

▶ **Theorem.** *For $\mathbb{Q}$, being a r.e. enumerable class of OMQs from $(GDLog, UCQ)$, under the ETH, the following are equivalent:*

1. *p-OMQ($\mathbb{Q}$) is fixed-parameter tractable*
2. *$\mathbb{Q}_c$, the class of covers of OMQs from $\mathbb{Q}$, has bounded SMW*
3. *$\mathbb{D}_{\mathbb{Q}}$, the class of characteristic DBs associated to OMQs from $\mathbb{Q}$, has bounded SMW.*

We start with the counter-positive of direction "1 $\Rightarrow$ 3", i.e. unbounded SMW for characteristic databases implies non-fpt. To this end, the following reduction is introduced:

▶ **Theorem 37.** *Let $\mathbb{Q}$ be a r.e. enumerable class of OMQs from $(GDLog, UCQ)$. Then, there exists an fpt-reduction from p-eval($\mathbb{D}_{\mathbb{Q}}$) to p-OMQ($\mathbb{Q}$).*

Note that the reduction talks about the problem of evaluating CQs from $\mathbb{D}_{\mathbb{Q}}$, but in that class we have characteristic databases. Thus, at this point of the proof such databases are seen as CQs. Once we have such a reduction, the result follows through straightforward: if

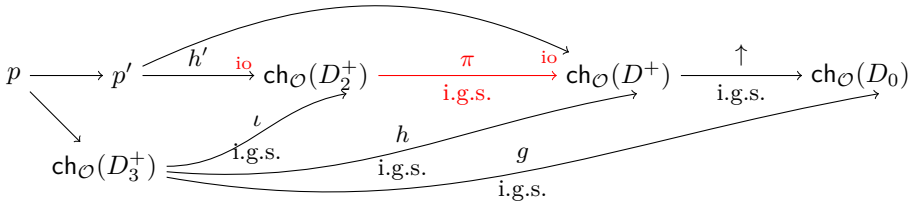


Figure 5 Proving the fpt reduction from CQ evaluation to OMQ evaluation.

$\mathbb{D}_{\mathbb{Q}}$ has unbounded SMW then p-eval($\mathbb{D}_{\mathbb{Q}}$) is not fpt according to DB theory results [18] (as mentioned also in Section 3), and then according to the reduction p-OMQ($\mathbb{Q}$) is not fpt as well.

We do not provide the full proof of the reduction, only some intuition, but the interested reader can find it in [21].

Proof. Let (D, B) be an instance of p-eval($\mathbb{D}_{\mathbb{Q}}$). Then $D \in \mathbb{D}_Q$, for $Q \in \mathbb{Q}$, and there exists $(D, \uparrow) \in \mathsf{mdiv}(D_0)$, for some database D_0 which is qi w.r.t. Q. A new database $D_2 \subseteq D \times B$ is constructed to which we add guarded unravelings of D_0 to obtain D_2^+. It can be shown that: $D_2^+ \models Q$ iff $D \to B$.

"⇒": Assume $D_2^+ \models Q$. It is possible to construct a diversification (D_3, ι) of D_2 and to show that it is also a diversification of D and that $(D_3, g) \in \mathsf{div}(D_0, Q)$, where $g = \iota \circ \pi \circ \uparrow$. The multiple databases and homomorphisms are depicted in Figure 5. As $(D, \uparrow) \in \mathsf{mdiv}(D_0, Q)$, $D \preceq D_3$. Thus, $D \to D_3 \to D_2 \to D \times B \to B$. Injectivity, and in particular qi DBs play an important role: D_0 is qi and so are D^+, D_2^+, and D_3^+. Injectivity is forced on π and then propagated to h.

"⇐": $D \to B$ implies $D \to D \times B \approx D_2$ implies $D_2^+ \models Q$ (as $D^+ \models Q$) ◀

As concerns direction "3 ⇒ 2" of the characterization, the following lemma is employed:

▶ **Lemma 38.** *Let $Q = (\mathcal{O}, \mathbf{S}, q)$ be an OMQ from $(GDLog, UCQ)$ and $Q_c = (\mathcal{O}, \mathbf{S}, q_c)$ be its cover. Then,* $\mathsf{SMW}(Q_c) \leq \mathsf{SMW}(\mathbb{D}_Q)$.

Intuitively, the lemma follows from the fact that the hypergraphs associated to CQs in Q_c are very closely related to those associated to databases from $\mathbb{D}_Q$: every maximal guarded set in the former is a maximal guarded set in the latter.

There is still one direction to be shown, direction "2 ⇒ 1":

▶ **Lemma 39.** *Let Q be an OMQ from (**GTGD**, UCQ) of bounded submodular width. Then, p-*$\mathsf{eval}(\mathsf{Q})$ *is fixed-parameter tractable.*

Note that the lemma is for OMQs from (GTGD, UCQ), thus stronger than what we need at this point. However, the lemma will prove useful in next section. To prove it, one can use a similar strategy as the one used in [3] to show a similar result for OMQs of bounded TW. Namely, one rewrites such OMQs into OMQs from (L, UCQ) with UCQ preservation, and thus also SMW preservation. Then, OMQs from (L, UCQ) are evaluated by materializing a finite portion of the chase in fpt; finally, the result follows from the complexity of evaluating CQs of bounded SMW which is known to be fpt [43].

5.5 Proofs for the characterization for OMQs from (GDTGD, UCQ)

Recall the previously announced characterization:

▶ **Theorem.** *Let $\mathbb{Q}$ be a r.e. class of OMQs from $(\boldsymbol{GTGD}, UCQ)$. Under the Exponential Time Hypothesis, $\mathbb{Q}$ has bounded semantic submodular width iff p-OMQ($\mathbb{Q}$) is fpt. If either statement is false, then p-OMQ($\mathbb{Q}$) is* W[1]*-hard.*

As discussed earlier we use the fact that $\mathbb{Q} \equiv \mathbb{Q}'$, for some $\mathbb{Q}' \in (\text{GDLOG}, \text{UCQ})$.

Proof. "⇐": p-OMQ($\mathbb{Q}$) is fpt implies that p-OMQ($\mathbb{Q}'$) is fpt as well. From the (GDLOg, UCQ) characterization we know that ${\mathbb{Q}'}_c$, the class of covers of OMQs from p-OMQ($\mathbb{Q}'$), has bounded SMW. As $\mathbb{Q} \equiv {\mathbb{Q}'}_c$, it means that $\mathbb{Q}$ has bounded semantic SMW.

"⇒": $\mathbb{Q} \equiv \mathbb{Q}_k$, where $\mathsf{SMW}(\mathbb{Q}_k) \leq k$. Then, from Lemma 39, it follows that p-OMQ($\mathbb{Q}_k$) is fpt, and so is p-OMQ($\mathbb{Q}'_k$), $\mathbb{Q}'_k$, the class (GDLog, UCQ) rewritings of $\mathbb{Q}_k$. Then, $\mathbb{D}_{\mathbb{Q}'_k}$ has bounded SMW.

The problem is that we do not have $\mathbb{Q}_k$ and neither do we have $\mathbb{Q}'_k$! Still, we have $\mathbb{Q}'$. We show how that can be exploited. We know that $\mathbb{Q} \equiv \mathbb{Q}' \equiv \mathbb{Q}'_k$, where both $\mathbb{Q}'$ and $\mathbb{Q}_k$ are from (GDLog, UCQ). Still, there is no guarantee that $\mathbb{D}_{\mathbb{Q}'}$ has bounded SMW! A new class of OMQs is constructed: $\mathbb{Q}_\cap \equiv \mathbb{Q}$ (cover-like) based on $\mathbb{D}_{\mathbb{Q}'_k} \cap \mathbb{D}_{\mathbb{Q}'}$. This new class has bounded SMW! Then p-OMQ($\mathbb{Q}_\cap$) is fpt and thus, p-OMQ($\mathbb{Q}$) is fpt as well. ◀

5.6 Wrap-up

The fpt characterization for (GTGD, UCQ) in the unbounded arity case introduced new tools for analyzing and manipulating OMQs: *extended characteristic databases* and *covers*.

The fpt-reduction from CQ evaluation to OMQ evaluation which came along with the approach makes it possible to port hardness results from database theory to the OMQ world in a blackbox fashion. For example, the hardness result for (GTGD, UCQ) over bounded arity schemas can be retrieved using this reduction, as well. By focusing on that setting, one can also establish using the constructs introduced in this section aa new syntactical characterization for OMQs from (GDLog, UCQ) over bounded arity schema.

The fpt-reduction has potential as a new technique to obtain complexity characterizations for other problems for OMQs inspired from database theory: counting, enumeration, and so on. The new constructs such as covers and characteristic databases could also be used to attack other OMQ problems, especially in those case where there are schema restrictions: in these cases canonical databases which entail the OMQ are always needed for proofs and are not necessarily easy to obtain.

6 Other Approaches Related to Fine-Grained Complexity Analysis

6.1 Other fpt Characterizations

Firstly, there are fpt characterizations concerning other reasoning tasks associated to OMQs like *enumeration* and *counting*.

For an **S**-OMQ Q and an **S**-database D, the *enumeration task* consists in listing all tuples $\mathbf{a}$ from $\mathsf{dom}(D)^n$ such that $D \models Q(\mathbf{a})$. In this direction, [39] studies enumeration for OMQs based on (GTGD, CQ) and ($\mathcal{ELI}$, CQ) and establishes the following:

- a CDLin algorithm for enumerating results to OMQs which are free-connex acyclic and acyclic, i.e. in algorithm which takes linear time preprocessing and enumerates results with *constant delay*
- some partial matching bounds for OMQs based on (EL, CQ)

The results mentioned above are lifted in [40] to the case with functional dependencies. Both works lift similar results from DB theory, and similarly to those results, the lower bounds apply only to self join free queries, i.e. queries with distinct relational symbols. An additional restriction in the OMQ setting is that chasing such queries must preserve self join freeness.

As concerns counting, which is the task of counting the number of answers to an OMQ over a given database, [24] introduces an fpt characterization for counting answers to OMQs from (GTGD, UCQ) over full database schemas.

Returning to *evaluation*, this time of *more expressive OMQs* based on DLs which also allow for disjunction like $\mathcal{ALC}$ and $\mathcal{ALCI}$, [41] establishes some fpl (fixed parameter linear) evaluation results for OMQs of bounded clique-width.

A similar line of research has been launched also for evaluation of *unions of conjunctive two-way regular path queries (UC2RPQs)*, which are navigational queries based on regular expressions, in [5, 6]. It has been established in [7] that:

- acyclic UC2RPQs can be evaluated in polynomial time and so do semantically acyclic UC2RPQs;
- deciding semantic acyclicity is EXPSPACE-complete.

As concerns the notion of semantic treewidth of (U)C2RPQs, [46] introduced two notions of C2RPQ equivalence: one based on homomorphisms, and another based on logical equivalence. While under the first notion semantic bounded TW leads to tractability, under the logical notion, fpt for C2RPQS of bounded TW follows by materializing regular paths queries (RPQs) and applying DB results.

As in the case of databases, an important question is that of semantic TW witnesses and deciding semantic TW for C2RPQs. While witnesses for bounded semantic TW have already been constructed in [46] they had non-optimal TW, in the sense that they might up to double the semantic TW of the original query; the evaluation of such witnesses takes then time $O(f(\Phi)|\mathcal{G}|^{2k+1})$ where f is some exponential function. Witnesses of optimal TW of size double exponential have been introduced in [26] and thus the existence of such witnesses settled also the question of deciding TW; as the witnesses have double exponential size this lead to an algorithm for evaluating C2RPQs of semantic bounded TW which runs in $O(f(\Phi)|\mathcal{G}|^{k+1})$ where f is again a double exponential function. It has been shown in [22] that it is possible to evaluate C2RPQs in time $O(g(\Phi)|\mathcal{G}|^{k+1})$, with g an exponential function. A precise characterization for the fpt border for C2RPQs is not yet established.

6.2 Approaches Based on Rewriting

Rewritability of OMQs based on CQs and OMQs based on instance queries has been investigated in [25]. *Instance Queries (IQs)* are queries of the form $C(x)$, with C a DL concept – they are generally easier to evaluate w.r.t. ontologies than CQs. For example, $(\mathcal{ALC}, CQ)$ is EXPTIME-complete, $(\mathcal{ALCI}, UCQ)$ is 2EXPTIME-complete, while the combined complexity is EXPTIME-complete in both cases for IQs. Furthermore, from a practical perspective, conjunctive query answering in expressive non-Horn DLs is a major area of theoretical investigation but which does not have much practical support, while answering OMQs based on IQs is widely supported by reasoners like Hermit, Pellet, etc.

While tree-shaped CQs can generally be rewritten into IQs w.r.t. an ontology, [36] showed that for expressive DLs (like $\mathcal{ALC}$ and above) the same holds also for some cyclic CQs. Starting from this observation, [25] asked when exactly is such a rewriting possible for expressive DLs such as $\mathcal{ALC}$ and $\mathcal{ALCI}$ and unary CQs (as IQs are unary queries, i.e.

queries with one answer variable, rewritability is only feasible for unary CQs as well). It obtained precise characterizations for rewritability based on witnesses (equivalent queries) which are x-acyclic, where this refers to the fact that cycle has to go through the answer variable.

Among other rewriting approaches, we mention [23] which investigates rewritability of OMQs from (MDDLog, UCQ) into OMQs from (MDLog, UCQ) (Datalog, UCQ), and UCQs using the connection of MDDLog with MMSNP and CSPs and rewriting results and techniques from the Constraint Satisfaction Problems (CSP) area. Note for OMQs for which there exist such rewritings, data complexity decreases: while this is coNP-complete for MDDLog it is PTIME-complete for MDLog and Datalog and AC^0 for UCQs. The combined complexity is affected as well: while (MDDLog, UCQ) is NEXPTIME-complete, the combined complexity decreases to EXPTIME-complete for Datalog-based languages [14] and NP-complete for UCQs [15].

7 Conclusions

The area of fine-grained complexity analysis offers a rich landscape of complexities and techniques to identify easy OMQs (with low complexity) which belong to an overall hard (more complex) language. Among such techniques we have seen fpt characterizations which target efficient evaluation from a parameterized perspective and approaches based on rewritings which target directly data and combined complexity. There are lots of open questions in this domain, e.g. establishing optimal witnesses in the general case for OMQs for semantic TW and semantic SMW or establishing a precise characterization of the fpt border for UC2RPQs.

References

1 Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2003.

2 Franz Baader, Carsten Lutz, Heike Sturm, and Frank Wolter. Fusions of description logics and abstract description systems. *arXiv*, 2011. `arXiv:1106.1802`.

3 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 *39th ACM Symposium on Principles of Database Systems (PODS-20)*. ACM Press, 2020.

4 Pablo Barceló, Cristina Feier, Carsten Lutz, and Andreas Pieris. When is ontology-mediated querying efficient? In *Thirty-Fourth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019.

5 Pablo Barceló, Leonid Libkin, Anthony W. Lin, and Peter T. Wood. Expressive languages for path queries over graph-structured data. *ACM Trans. Database Syst.*, 37(4), 2012. `doi:10.1145/2389241.2389250`.

6 Pablo Barceló, Jorge Pérez, and Juan L. Reutter. Relative expressiveness of nested regular expressions. In *Proc. of the 6th Alberto Mendelzon International Workshop on Foundations of Data Management, June 27-30, 2012*, volume 866, pages 180–195. CEUR-WS.org, 2012. URL: `https://ceur-ws.org/Vol-866/paper13.pdf`.

7 Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. Semantic acyclicity on graph databases. *SIAM Journal on Computing*, 45(4):1339–1376, 2016. `doi:10.1137/15M1034714`.

8 Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544. Association for Computational Linguistics, 2013. `doi:10.18653/V1/D13-1160`.

9 Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):28–37, 2001.

10 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.

11 Manuel Bodirsky. Cores of structures and constraint satisfaction problems. *Theoretical Computer Science*, 412(18):1898–1910, 2011. doi:10.1016/j.tcs.2010.12.017.

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 Thomas Lukasiewicz. Datalog±: a unified approach to ontologies and integrity constraints. In *ICDT*, pages 14–30, 2009. doi:10.1145/1514894.1514897.

14 Ashok K. Chandra and David Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25(1):99–128, 1982. doi:10.1016/0022-0000(82)90004-7.

15 Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977. doi:10.1145/800105.803397.

16 Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.*, 239(2):211–229, 2000. doi:10.1016/S0304-3975(99)00220-0.

17 Hubie Chen, Georg Gottlob, Matthias Lanzinger, and Reinhard Pichler. Semantic width and the fixed-parameter tractability of constraint satisfaction problems. In Christian Bessiere, editor, *IJCAI*, pages 1726–1733, 2020. doi:10.24963/IJCAI.2020/239.

18 Hubie Chen and Stefan Mengel. A Trichotomy in the Complexity of Counting Answers to Conjunctive Queries. In *(ICDT 2015)*, volume 31, pages 110–126, 2015. doi:10.4230/LIPICS.ICDT.2015.110.

19 Alin Deutsch, Alan Nash, and Jeff B. Remmel. The chase revisisted. In *PODS*, pages 149–158, 2008.

20 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: basic results. *SIAM Journal on Computing*, 24:873–921, 1995. doi:10.1137/S0097539792228228.

21 Cristina Feier. Characterising Fixed Parameter Tractability for Query Evaluation Over Guarded TGDs. In *ICDT*, volume 220, pages 12:1–12:22, 2022.

22 Cristina Feier, Tomasz Gogacz, and Filip Murlak. Evaluating graph queries using semantic treewidth. In *27th International Conference on Database Theory*, volume 290, pages 22:1–22:20, 2024. doi:10.4230/LIPICS.ICDT.2024.22.

23 Cristina Feier, Antti Kuusisto, and Carsten Lutz. Rewritability in monadic disjunctive datalog, mmsnp, and expressive description logics. *Logical Methods in Computer Science*, 15(2):15:1–15:46, 2019. doi:10.23638/LMCS-15(2:15)2019.

24 Cristina Feier, Carsten Lutz, and Marcin Przybyłko. Answer Counting Under Guarded TGDs. In Ke Yi and Zhewei Wei, editors, *24th International Conference on Database Theory (ICDT 2021)*, volume 186, pages 11:1–11:22, 2021. doi:10.4230/LIPICS.ICDT.2021.11.

25 Cristina Feier, Carsten Lutz, and Frank Wolter. From conjunctive queries to instance queries in ontology-mediated querying. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-ECAI 2018)*. AAAI Press, 2018.

26 Diego Figueira and Rémi Morvan. Approximation and Semantic Tree-width of Conjunctive Regular Path Queries. In *26th International Conference on Database Theory*, Ioannina, Greece, 2023. URL: https://hal.archives-ouvertes.fr/hal-03883042.

27 Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002. doi:10.1006/JCSS.2001.1809.

28 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.

29 Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. *ACM Trans. Algorithms*, 11(1), August 2014. doi:10.1145/2636918.

30 Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993. doi:10.1006/knac.1993.1008.

31 Lingxiao Guo and et al. Text2sparql: Neural semantic parsing for knowledge graph question answering. In *Proceedings of AAAI*, 2022.

32 Wilfrid Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1993.

33 Thomas Hofweber. Ontology. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2021 edition, 2021.

34 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/JCSS.2001.1774.

35 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.

36 Stanislav Kikot and Evgeny Zolin. Modal definability of first-order formulas with free variables and query answering. *J. Applied Logic*, 11(2):190–216, 2013. doi:10.1016/J.JAL.2013.03.007.

37 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2005.11401.

38 Carsten Lutz. Non-uniform data complexity of query answering in description logics. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 276–286, 2012.

39 Carsten Lutz and Marcin Przybylko. Efficiently enumerating answers to ontology-mediated queries. In *PODS '22: International Conference on Management of Data*, pages 277–289, 2022. doi:10.1145/3517804.3524166.

40 Carsten Lutz and Marcin Przybylko. Efficient answer enumeration in description logics with functional roles. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, pages 6483–6490, 2023. doi:10.1609/AAAI.V37I5.25797.

41 Carsten Lutz, Leif Sabellek, and Lukas Schulze. Ontology-mediated querying on databases of bounded cliquewidth. In *Proc. of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR*, 2022.

42 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.

43 Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In *STOC*, pages 735–744, 2010. doi:10.1145/1806689.1806790.

44 Marvin Minsky. *A Framework for Representing Knowledge*. MIT AI Laboratory, Memo 306, 1975. Also published in *The Psychology of Computer Vision*, P.H. Winston (Ed.), 1975, McGraw-Hill.

45 W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 2009. Available at http://www.w3.org/TR/owl2-overview/.

46 Miguel Romero, Pablo Barceló, and Moshe Y. Vardi. The homomorphism problem for regular graph patterns. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005106.

47 Kartik Sharma, Peeyush Kumar, and Yunqing Li. Og-rag: Ontology-grounded retrieval-augmented generation for large language models. *arXiv preprint arXiv:2412.15235*, 2024. doi:10.48550/arXiv.2412.15235.

48 Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94, 1981.

49 Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Question answering over freebase with multi-column convolutional neural networks. *arXiv preprint arXiv:1609.07843*, 2016.