The Limits of Efficiency for Open- and Closed-World Query Evaluation Under Guarded TGDs

Pablo Barceló
IMC, PUC Chile & IMFD Chile
pbarcelo@ing.puc.cl

Victor Dalmau
Universitat Pompeu Fabra
victor.dalmau@upf.edu

Cristina Feier
University of Bremen
feier@uni-bremen.de

Carsten Lutz
University of Bremen
clu@uni-bremen.de

Andreas Pieris
University of Edinburgh
apiерis@inf.ed.ac.uk

ABSTRACT

Ontology-mediated querying and querying in the presence of constraints are two key database problems where tuple-generating dependencies (TGDs) play a central role. In ontology-mediated querying, TGDs can formalize the ontology and thus derive additional facts from the given data, while in querying in the presence of constraints, they restrict the set of admissible databases. In this work, we study the limits of efficient query evaluation in the context of the above two problems, focusing on guarded and frontier-guarded TGDs and on UCQs as the actual queries. We show that a class of ontology-mediated queries (OMQs) based on guarded TGDs can be evaluated in FPT iff the OMQs in the class are equivalent to OMQs in which the actual query has bounded treewidth, up to some reasonable assumptions. For querying in the presence of constraints, we consider classes of constraint-query specifications (CQs) that bundle a set of constraints with an actual query. We show a dichotomy result for CQs based on guarded TGDs that parallels the one for OMQs except that, additionally, FPT coincides with PTime combined complexity. The proof is based on a novel connection between OMQ and CQ evaluation. Using a direct proof, we also show a similar dichotomy result, again up to some reasonable assumptions, for CQs based on frontier-guarded TGDs with a bounded number of atoms in TGD heads. Our results on CQs can be viewed as extensions of Grohe’s well-known characterization of the tractable classes of CQs (without constraints). Like Grohe’s characterization, all the above results assume that the arity of relation symbols is bounded by a constant. We also study the associated meta problems, i.e., whether a given OMQ or CQ is equivalent to one in which the actual query has bounded treewidth.

CCS CONCEPTS
• Theory of computation → Logic and databases; Database query processing and optimization (theory);

1 INTRODUCTION

Tuple-generating dependencies (TGDs) are a prominent rule-based formalism at the core of several areas that are of central importance to databases and artificial intelligence. They are first-order implications of the form $\forall x \forall y (\phi(x, y) \rightarrow \exists z \psi(x, z))$, where $\phi$ and $\psi$ are conjunctions of relational atoms, and they essentially state that some tuples (facts) in a relational instance imply the presence of some other tuples in that instance (hence the name “tuple-generating”). In query evaluation over relational databases, TGDs have two facets: (1) they can be used as ontology axioms that allow us to derive additional facts from the given data, which supports more complete query answers, and (2) they can be used as integrity constraints that restrict the set of admissible databases, which paves the way to constraint-aware query optimization. In fact, TGDs were originally introduced as a unifying framework for the large range of relational constraints introduced in the 1970s and the 1980s [1].

The Two Facets of TGDs. When a set of TGDs is used as an ontology, it is often bundled with an actual database query, typically a union of conjunctive queries (UCQ), to form a composite ontology-mediated query (OMQ) [11, 12]. As mentioned above, adding the ontology serves the purpose of delivering more complete answers to queries. It also enriches the vocabulary available for querying as it may introduce new relation symbols that are not part of the data schema. An OMQ language is a pair $(\mathbb{C}, \mathbb{Q})$, with $\mathbb{C}$ being a class of TGDs and $\mathbb{Q}$ a query language, which collects all the OMQs in which the ontology is formulated in $\mathbb{C}$ and the actual query comes from $\mathbb{Q}$ [12]. The main problem of concern for an OMQ language $(\mathbb{C}, \mathbb{Q})$ is OMQ evaluation under an open-world semantics, i.e., we are looking for answers to the actual query from $\mathbb{Q}$ that are logically entailed by the input database and the ontology from $\mathbb{C}$.
On the other hand, when a set of TGDs is used as relational integrity constraints, the TGDs together with the actual query can be bundled together into what we call a constraint-query specification (CQS). A class of CQSs is a pair \((\mathbb{C}, \mathbb{Q})\), where, as in OMQs, \(\mathbb{C}\) is a class of TGDs and \(\mathbb{Q}\) a query language. In this setting, however, the problem of interest is CQS evaluation under a closed-world semantics, i.e., we are looking for answers to the query from \(\mathbb{Q}\) over the input database, which is promised to comply with the set of constraints from \(\mathbb{C}\). In other words, we directly evaluate the query over the database, which we know to satisfy the set of constraints.

**TGDs and Guardedness.** It is well-known that OMQ evaluation in \((\text{TGD}, \text{UCQ})\), where \(\text{TGD}\) is the class of all TGDs, and \(\text{UCQ}\) the class of union of conjunctive queries, is an undecidable problem; see, e.g., [14]. This has led to intense research activity for identifying restrictions on TGDs that ensure the decidability of OMQ evaluation; see, e.g., [3, 14, 16, 28] – the list is by no means exhaustive. One of the most robust syntactic paradigms that emerged from this extensive effort, which is central to our work, is guardedness. A TGD is guarded if the left-hand of the implication, called the body, has an atom that contains (or guards) all the universally quantified variables [14]; let \(\mathbb{G}\) be the class of guarded TGDs. A natural generalization is to guard only the variables that appear also in the right-hand side, called the head, which leads to the class of frontier-guarded TGDs \(\text{FG}\) [3]. The complexity of OMQ evaluation in \((\mathbb{C}, \text{UCQ})\), where \(\mathbb{C} \in \{\mathbb{G}, \text{FG}\}\), is by now well-understood. In both cases, it is \(\text{2ExpTime}\)-complete [2, 14], and it remains hard even in the case of bounded-arity schemas, i.e., \(\text{2ExpTime}\)-complete in \((\text{FG}, \text{UCQ})\) [2], and \(\text{ExpTime}\)-complete in \((\mathbb{G}, \text{UCQ})\) [14].

Although guarded and frontier-guarded TGDs have been proposed as ontology languages, they can also naturally serve as classes of integrity constraints [9]. Note, for example, that the important class of referential integrity constraints (or inclusion dependencies) is a very special case of guarded TGDs. Now, the complexity of CQS evaluation in \((\mathbb{C}, \text{UCQ})\), where \(\mathbb{C} \in \{\mathbb{G}, \text{FG}\}\), coincides with that of query evaluation for \(\text{UCQ}\); it is \(\text{NP}\)-complete, even for schemas of bounded arity [17]. This holds since CQS evaluation in \((\mathbb{C}, \text{UCQ})\) is, in fact, a refinement of the query evaluation problem for \(\text{UCQ}\), with the additional promise that the input database satisfies a given (but potentially empty) set of integrity constraints coming from \(\mathbb{C}\).

**The Limits of Efficiency.** Thus both OMQ evaluation and CQS evaluation in \((\mathbb{C}, \text{UCQ})\), where \(\mathbb{C} \in \{\mathbb{G}, \text{FG}\}\), are computationally hard problems. However, there are subclasses of \((\mathbb{C}, \text{UCQ})\), seen either as an OMQ language or a class of CQSs, for which the evaluation problem is efficient in the sense of being tractable or being fixed-parameter tractable (\(\text{FPT}\)) where the parameter is the size of the OMQ (resp., CQS). In particular, with \(\text{UCQ}_k\) being the class of UCQSs of treewidth at most \(k\), we know that: (1) for each \(k \geq 1\), OMQ evaluation in \((\mathbb{G}, \text{UCQ}_k)\) is in \(\text{FPT}\) (actually, this is shown in this work – Proposition 3.3), and (2) for each \(k \geq 1\), CQS evaluation in \((\text{FG}, \text{UCQ}_k)\) is in \(\text{FPT}\). Both statements rely on the well-known result that query evaluation for UCQSs of bounded treewidth is tractable [18]. In view of this, it is natural to ask whether we can precisely characterize the limits of efficient OMQ and CQS evaluation. The goal of this work is to provide such efficiency characterizations.

A seminal result by Grohe precisely characterizes the (recursively enumerable) classes of CQs over schemas of bounded arity that can be evaluated in polynomial time (under the assumption that \(\text{FPT} \neq \text{W}[1]\)) [26]: this is the case if and only if for some \(k \geq 1\), every CQ in the class is equivalent to a CQ of treewidth \(k\). Grohe’s result also establishes that \(\text{PTIME}\) and \(\text{FPT}\) coincide for evaluating classes of CQs. This naturally generalizes to UCQs.

Efficiency characterizations in the same spirit have been recently obtained for OMQ languages based on description logics (DLs). In particular, [7] precisely characterizes the (recursively enumerable) classes of OMQs from \((\text{ELH}_1, \text{UCQ})\), where \(\text{ELH}_1\) is an important DL, essentially a fragment of guarded TGDs. Here, OMQ evaluation is in \(\text{FPT}\) (assuming \(\text{FPT} \neq \text{W}[1]\)) if and only if for some \(k \geq 1\), every OMQ in the class is equivalent to an OMQ of treewidth \(k\), i.e., an OMQ such that the UCQ in it is of treewidth \(k\). Note that the equivalence is now on the level of OMQs rather than on the level of UCQs. The same work [7] precisely characterizes the (recursively enumerable) classes of OMQs from \((\text{ELH}_1, \text{UCQ})\), where \(\text{ELH}_1\) is a key fragment of \(\text{ELH}_1\) that underpins the OWL 2 EL profile of the OWL 2 recommendation [30]. Here, evaluation is in \(\text{PTIME}\) (again assuming \(\text{FPT} \neq \text{W}[1]\)) if and only if for some \(k \geq 1\), every OMQ in the class is equivalent to an OMQ of treewidth \(k\). This also shows that \(\text{PTIME}\) and \(\text{FPT}\) coincide for evaluating classes of OMQs from \((\text{ELH}_1, \text{UCQ})\). The classification of \((\text{ELH}_1, \text{UCQ})\), however, is subject to the condition that the ontology does not introduce relations beyond those admitted in the database.

**Our Results.** Our main results are as follows, under the widely believed complexity-theoretic assumption that \(\text{FPT} \neq \text{W}[1]\):

1. For a recursively enumerable class of OMQs from \((\mathbb{G}, \text{UCQ})\) over a schema of bounded arity, evaluation is in \(\text{FPT}\) if and only if there is \(k \geq 1\) such that each OMQ in the class is equivalent to one from \((\mathbb{G}, \text{UCQ}_k)\) (Theorem 5.3).

2. For a recursively enumerable class of CQs from \((\mathbb{G}, \text{UCQ})\) over a schema of bounded arity, evaluation is in \(\text{PTIME}\) if and only if evaluation is in \(\text{FPT}\) if and only if there is \(k \geq 1\) such that each CQ in the class is equivalent to one from \((\mathbb{G}, \text{UCQ}_k)\) (Theorem 5.7).

3. Our final result concerns CQs from \((\text{FG}, \text{UCQ})\) in which the TGDs have a bounded number of at most \(m \geq 1\) head atoms; let \(\text{FG}_m\) be the obtained class. For a recursively enumerable class of CQs from \((\text{FG}_m, \text{UCQ})\) over a schema of bounded arity, evaluation is in \(\text{PTIME}\) if and only if evaluation is in \(\text{FPT}\) if and only if there is \(k \geq 1\) such that each CQ in the class is equivalent to one from \((\text{FG}_m, \text{UCQ}_k)\) (Theorem 5.12).

It is not surprising that characterizing (1) talks only about \(\text{FPT}\) and not \(\text{PTIME}\) complexity since we know from [14] that the \(\text{ExpTime}\)-hardness of OMQ evaluation in \((\mathbb{G}, \text{UCQ})\) in the case of schemas of bounded arity holds even if the actual query is an atomic query of the simplest form, i.e., a propositional atom. Moreover, it turns out that the notion of being equivalent to an OMQ of bounded treewidth is not enough for characterizing evaluation in \(\text{FPT}\) for frontier-guarded TGDs. We can show that OMQ evaluation in \((\text{FG}, \text{UCQ})\) is \(\text{W}[1]\)-hard – this is actually easily inherited from the fact that Boolean CQ evaluation is \(\text{W}[1]\)-hard [31].

For all the above characterizations, if the efficiency condition fails, i.e., there is no integer \(k \geq 1\) such that each OMQ or CQ in the considered class is equivalent to one in which the actual query falls
within UCQₖ, then the evaluation problem is W[1]-hard. Showing these lower bounds is actually the most challenging task underlying our efficiency characterizations. Together with the assumption that FPT ≠ W[1], they establish that efficient evaluation implies the efficiency condition. For (1) and (3), this is done via an fpt-reduction from the parameterized version of the k-clique problem, a well-known W[1]-hard problem, building on Grohe’s result which also relies on an fpt-reduction from k-clique. For (2), we exploit a novel connection between OMQ evaluation and CQS evaluation, which is of independent interest. In fact, we provide an fpt-reduction from OMQ evaluation to CQS evaluation.

At this point, we would like to stress that the fpt-reductions underlyng (1) and (3) are not merely adaptations of the fpt-reduction by Grohe. For (1), the fact that the ontology can introduce additional relations beyond the data schema causes serious challenges. Moreover, unlike the characterizations of [7] for OMQs based on DLs, where only unary and binary relations are used, we need to deal with relations of arity beyond two, which also causes additional non-trivial complications that require novel ideas and techniques. For (3), unlike Grohe, we are more constrained in defining the right database as it must satisfy the given set of constraints. Moreover, the useful notion of the core of a CQ, which was crucial for Grohe’s proof, cannot be directly used in the presence of constraints.

We further study the complexity of the associated meta problems of deciding whether, for some fixed k, a given OMQ or CQS is equivalent to one in which the actual query falls within UCQₖ. We show that in all the considered cases, the problem is 2ExpTime-hard where each R with relations of arity beyond two, which also causes additional variables to J.

2 PRELIMINARIES

We consider the disjoint countably infinite sets C and V of constants and variables, respectively. We refer to constants and variables as terms. For an integer n ≥ 1, we write [n] for the set {1, ..., n}.

Relational Databases. A (relational) schema S is a finite set of relation symbols (or predicates) with associated arity. We write ar(R) for the arity of a predicate R, and ar(S) for the arity of S, that is, the number max_{R∈S} ar(R). An atom over S is an expression of the form R(t), where R ∈ S and t is an ar(R)-tuple of terms. An instance over S, or simply Σ-instance, is a (possibly infinite) set of atoms over S that contain only constants, while a database over S, or simply S-database, is a finite S-instance. We write dom(I) for the set of constants in an instance I. For a set T ⊆ dom(I), we denote by I|ₚ the restriction of I to atoms that mention only constants from T. A homomorphism from I to an instance J is a function h : dom(I) → dom(J) such that R(h(t)) ∈ J for every R(t) ∈ I. We write I → J for the fact that there is a homomorphism from I to J.

Conjunctive Queries. A conjunctive query (CQ) over a schema S is a first-order formula of the form q(̄x) := ∃y (R₁(̄x₁) ∧ ... ∧ Rₘ(̄xₘ)), where each Rᵢ(̄xᵢ), for i ∈ [m], is an atom over S that contains only variables, each variable mentioned in the ̄xᵢ appears either in ̄x or ̄y, and ̄x contains all the free variables of q called the answer variables. 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 viewing variables as constants. We may simply write q instead of D[q]. A homomorphism from a CQ q to an instance I is a homomorphism from D[q] to I. A tuple ̄c ∈ dom(I)|ₚ is an answer to Q over I if there is a homomorphism h from q to I with h(̄c) = ̄c.

Showing the evaluation of q(̄x) over I, denoted q(I), is the set of all answers to Q over I. We write CQ for the class of CQs. A union of conjunctive queries (UCQ) over a schema S is a first-order formula of the form q(̄x) := q₁(̄x) ∨ ... ∨ qₙ(̄x), where n ≥ 1, and qᵢ(̄x), for i ∈ [n], is a CQ over S. The evaluation of q over an instance I, denoted q(I), is the set of tuples |I|∈[n] qᵢ(I).

Let UCQ be the class of UCQs. The arity of a UCQ is the number of its answer variables. A UCQ of arity zero is called Boolean, and it can have as an answer only the empty tuple. For a Boolean UCQ q, we may write I |= q if q(I) = {()}, and I \models q, otherwise.

Treedepth. A central notion in our work is that of treedepth, which 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 δ = (Tδ, χ), where Tδ = (Vδ, Eδ) is a tree, and χ is a labeling function Vδ → 2δ, i.e., χ assigns a subset of V to each node of Tδ, such that:

1. U, V ∈ Tδ implies U ∩ V ∈ Tδ,
2. If U ∈ Tδ, then there exists a V ∈ Tδ such that U ⊆ V,
3. For each a ∈ V, the set of nodes {t ∈ Tδ | a ∈ χ(t)} induces a connected subtree of Tδ.

The width of δ is the maximum number max_{t ∈ Tδ} |{χ(t)|} − 1. If the edge-set E of G is non-empty, then the treedepth 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 = dom(I), and {a, b} ∈ E iff there is an atom R(i) ∈ I that mentions both a and b. The treedepth of I is the treedepth of G[I].

The evaluation problem for (U)CQs takes as input a (U)CQ q(̄x), a database D, and a candidate answer ̄c, and asks whether ̄c ∈ q(D). It is well-known that (U)CQ evaluation is NP-complete [17]. On the other hand, it becomes tractable by restricting the syntactic shape of CQs. One of the most widely studied such restrictions is bounded treedepth. Formally, a CQ q(̄x) = ∃y ϕ(̄x, ̄y) has treedepth k ≥ 1 if G[q] has treedepth k, where G[q] is the subgraph of G[q] induced by the elements of ̄y. Note that the treedepth of q is defined in a more liberal way than usual. The standard definition considers the treedepth of G[q], while here the treedepth of q is only measured with respect to the subgraph of G[q] induced by its existentially quantified variables. A UCQ q has treedepth k if each of its disjuncts has treedepth at most k. We write UCQₖ (resp., UCQₖ) for the class of CQs (resp. UCQs) of treedepth at most k ≥ 1. The next well-known result illustrates the usefulness of bounding the treedepth.

Proposition 2.1 ([18]). Fix k ≥ 1. Given a database D, an n-ary query q ∈ UCQₖ, and a tuple ̄c ∈ dom(D)^n, the problem of deciding whether ̄c ∈ q(D) can be solved in time O(|D|^k+1 ⋅ ||q||).¹

¹As usual, given a syntactic object O, we write ||O|| for its size.
∀x∀y (φ(x, y) → ∃zψ(χ, z)), where φ is a possibly empty conjunction of atoms over S, while ψ is a non-empty conjunction of atoms over S. For simplicity, we write σ as φ(χ, τ) → ∃zψ(χ, z), and use comma instead of ∧ for joining atoms. We call φ and ψ the body and head of σ, denoted body(σ) and head(σ), respectively. The frontier of σ, denoted fr(σ), is the set of variables x, i.e., the variables that appear both in the body and the head of σ. The TGD σ above is logically equivalent to the expression ∃x(∀yφ(χ, τ) → ψ(χ, z)), where qφ(x) and qψ(x) are the CQs ∃xφ(χ, τ) and ∃xψ(χ, z), respectively. Therefore, an instance l over S satisfies σ, denoted l ⊨ σ, if qφ(l) ⊆ qψ(l). An instance l satisfies a set Σ of TGDs, denoted l ⊨ Σ, if l ⊨ σ for each σ ∈ Σ. Henceforth, whenever we refer to a set of TGDs we mean a finite set. We write Π(Q) for the class of TGDs, that is, the family of all possible sets of TGDs.

Frontier-Guardedness. A TGD σ is guarded if either body(σ) is empty, or there exists an atom α in its body that contains all the variables occurring in body(σ) [14]. Such an atom α is the guard of σ, denoted guard(σ). We write ∅ for the class of guarded TGDs. A natural generalization of guardedness is frontier-guardedness, where only the frontier variables must be guarded. Formally, a TGD σ is frontier-guarded if either body(σ) is empty or there exists an atom α in its body that contains all the variables of fr(σ) [32], which we call again guard and denote as guard(σ). The class of frontier-guarded TGDs is denoted by ΠG. Clearly, ∅ ⊆ ΠG ⊆ ΠQ.

The Chase Procedure. The chase is a useful tool when reasoning with TGDs [14, 22, 27, 29]. We first define a single chase step. Let I be an instance over a schema S and a TGD of the form φ(χ, τ) → ∃zψ(χ, z) over S. We say that σ is applicable with in I if there exists a tuple (c, c′) of constants in I such that φ(c, c′) ⊆ I. In this case, the result of applying σ in I with (c, c′) is the instance J = I ∪ ψ(χ, z), where z is the tuple obtained from z by simultaneously replacing each variable z with a fresh distinct constant not occurring in I. For such a single chase step, we write I → I.

Let I be an instance and Σ a set of TGDs. A chase sequence for I under Σ is a sequence of chase steps l0 → l1 → l2 → ... such that (1) l0 = I, (2) σi ∈ Σ for each i ≥ 0, and (3) J ⊆ l, where J = l ∪ l2 ∪ l3 ∪ ... . The instance J is the (potentially infinite) result of this chase sequence, which always exists. The chase sequence is fair if whenever a TGD σ ∈ Σ is applicable in some li with (c, c′), then lj → lj+1 is part of the sequence for some j ≥ i. Since we consider the oblivious chase, i.e., a TGD is triggered whenever its body is satisfied no matter whether its head is satisfied, every fair chase sequence for I under Σ leads to the same result (up to isomorphism). Thus, we can refer to the result of the chase for I under Σ, denoted chase(I, Σ). The key property of the chase follows:

Proposition 2.2. Consider an instance I and a set Σ of TGDs. For every instance J such that J ⊆ I and J ⊨ Σ, chase(I, Σ) → J via a homomorphism that is the identity on dom(I).

Parameterized Complexity. Parameterized complexity has a central role in our work. A parameterized problem over an alphabet Λ is a pair (P, κ), with P ⊆ Λ, a decision problem and κ a parameterization of P, that is, a PTIME computable function x : Λ → N. A prime example is p-Clique, where P is the set of all pairs (c, k) with an undirected graph that contains a k-clique and κ(c, k) = k.

A problem (P, κ) is fixed-parameter tractable (fpt) if there is a computable function f : N → N and an algorithm that decides P in time |x|O(1) · f(κ(x)), where x denotes the input. We use FPT to denote the class of all parameterized problems that are fixed-parameter tractable. Notice that FPT corresponds to a relaxation of the usual notion of tractability: if P is in PTIME, then (P, κ) is in FPT for any κ, but the latter might also be the case when P is NP-hard.

An fpt-reduction from a problem (P1, κ1) over Λ1 to a problem (P2, κ2) over Λ2 is a function ρ : Λ1 → Λ2 such that, for some computable functions f, g : N → N,

1. x ∈ P1 if ρ(x) ∈ P2, for all x ∈ Σ1;
2. ρ(x) is computable in time |x|O(1) · f(κ1(x)), for x ∈ Λ1;
3. κ2(ρ(x)) ≤ g(κ1(x)), for all x ∈ Λ1.

An important parameterized complexity class is W[1] ⊆ FPT. Hardness for W[1] is defined in terms of fpt-reductions. It is generally believed that FPT ≠ W[1], the status of this problem being comparable to that of PTIME ≠ NP. If a parameterized problem (P, κ) is in W[1]-hard, then (P, κ) is not fpt unless FPT = W[1]. A well-known W[1]-hard problem is p-Clique [21].

3 THE TWO FACETS OF TGDs IN QUERYING

As discussed in Section 1, TGDs can be used as:

1. Ontology axioms that enrich incomplete data with domain knowledge, which leads to more complete answers.
2. Integrity constraints that specify semantic properties satisfied by all databases, which paves the way to constraint-aware query optimization techniques.

Depending on how TGDs are used, we obtain different evaluation problems, which we now formalize.

3.1 TGDs as Ontology Axioms

When a set of TGDs is used as ontology axioms, or an ontology, it is typically seen, together with the actual query, as one composite query, called ontology-mediated query. Formally, an ontology-mediated query (OMQ) is a triple Q = (S, Σ, q), where S is a schema, called data schema, which indicates that Q will be evaluated over S-databases, Σ is a set of TGDs, called ontology, over an extended schema T ⊇ S, and q is a UCQ over T. In case S = T, we say that Q has full data schema. The arity of Q is defined as the arity of q. We write Q(χ) to emphasize that the answer variables of q are χ, and say that Q is over T to mean that the extended schema of Q is T.

The evaluation of an OMQ Q(χ) over a S-database D consists of all the tuples that are an answer to q for all models of D and Σ. A model of D and Σ is an instance I such that I ⊇ D and I ⊨ Σ. A tuple χ ∈ dom(D)[|χ|] is an answer to Q over D if χ ∈ q(I) for each model I of D and Σ. The evaluation of Q(χ) over D, denoted Q(D), is the set of all answers to Q over D. By Proposition 2.2, and the monotonicity of UCQs, we get the following useful result:

Proposition 3.1. For every OMQ Q = (S, Σ, q) ∈ (TGD, UCQ), Q(D) = q(chase(D, Σ)).

We write (C, Q) for the class of OMQs, called OMQ language, in which the ontology is formulated in the class of TGDs C and the
actual query is coming from the class of queries $Q$; for example, we may write $(G, QC), (FQ, UCQ), (G, UCQ_k)$, for some $k \geq 1$, etc. This brings us to the evaluation problem for OMQ languages $O$:

**PROBLEM**: OMQ-Evaluation($O$)

**INPUT**: An OMQ $Q = (S, \Sigma, q(x)) \in O$, an $S$-database $D$, and a tuple $\bar{c} \in \text{dom}(D)^{|\bar{x}|}$.

**QUESTION**: Is it the case that $\bar{c} \in q(D)$?

We are also interested in the parameterized version of the above problem, dubbed $p$-OMQ-Evaluation($O$), with the parameter being the size of the OMQ $Q$, as customary in the literature [31]. Thus $p$-OMQ-Evaluation($O$) is in FPT if it can be solved in time $|[D]|^{O(1)} \cdot f(|\bar{Q}|)$ for some computable function $f : \mathbb{N} \rightarrow \mathbb{N}$.

It is well-known that OMQ-Evaluation(TGD, $CQ$) is undecidable; see, e.g., [14]. On the other hand, if we focus on OMQs in which the ontology is formulated as a set of frontier-guarded TGDs, then the problem becomes decidable, in fact, it is 2ExpTime-complete [2]. At this point, one may wonder whether bounding the treewidth of the $CQ$s will have the same positive effect as in the case of $CQ$ evaluation (see Proposition 2.1). It is implicit in [14] that this is not the case. The next result summarizes some key facts about the evaluation problem for OMQs based on (frontier-)guarded TGDs.

**Proposition 3.2.** It holds that:

1. OMQ-Evaluation(FG, $UCQ$) is 2ExpTime-complete even for schemas of bounded arity.
2. OMQ-Evaluation($G$, $UCQ$) is 2ExpTime-complete, and becomes ExpTime-complete for schemas of bounded arity.
3. For each $k \geq 1$, OMQ-Evaluation($G$, $UCQ_k$) is 2ExpTime-complete, and still ExpTime-hard for schemas of bounded arity.

Since the parameterized version of the evaluation problem for $CQ$s is $W[1]$-hard [31], even for schemas of bounded arity, we can immediately conclude that $p$-OMQ-Evaluation($G$, $CQ$) is $W[1]$-hard, even for schemas of bounded arity. Do we gain something if we focus on $UCQ$s of bounded treewidth? The answer is negative for frontier-guarded TGDs, but affirmative for guarded TGDs.

**Proposition 3.3.** It holds that:

1. $p$-OMQ-Evaluation($G$, $CQ$) is $W[1]$-hard even for schemas of bounded arity.
2. $p$-OMQ-Evaluation($FG$, $CQ_k$) is $W[1]$-hard even for schemas of bounded arity.
3. For each $k \geq 1$, $p$-OMQ-Evaluation($G$, $UCQ_k$) is in FPT.

As discussed above, item (1) is a consequence of the fact that the parameterized version of $CQ$ evaluation is $W[1]$-hard, even for schemas of bounded arity. Item (2) is a consequence of the fact that the parameterized version of $CQ$ evaluation is $W[1]$-hard, even for Boolean $CQ$s over a schema of bounded arity, while a Boolean $CQ$ $\exists \phi(x)$ can be added as a frontier-guarded TGD to the ontology and replaced with the query $Ans$.

We now briefly explain how item (3) is shown. A TGD is called **linear** if it has only one atom in its body, while the class of linear TGDs is denoted $L$. The key ingredient underlying item (3) is that, given an $S$-database $D$ and an OMQ $Q = (S, \Sigma, q)$ from $(G, UCQ)$, $q(D)$ coincides with the evaluation of $q$ over an initial finite portion $\Sigma^*$ of chase($D^*, \Sigma^*$), where $D^*$ can be computed solely from $D$ and $\Sigma$, and $\Sigma^* \in L$ can be computed solely from $\Sigma$. Roughly, $C$ is the finite instance obtained by keeping only the atoms of chase($D^*, \Sigma^*$) up to a finite level that depends only on $\Sigma$ and $q$, while the notion of level indicates the distance of an atom in the chase from the starting database. Furthermore, the instance $C$ can be computed in time $|[D]|^{O(1)} \cdot f(|\bar{Q}|)$ for some computable triple exponential function $f : \mathbb{N} \rightarrow \mathbb{N}$. Therefore, to decide whether a tuple $\bar{c}$ over dom($D$) belongs to $q(D)$, it suffices to construct the finite instance $C$, and accept if $\bar{c}$ belongs to $q(C)$; otherwise, reject. Since $q \in UCQ_k$, by Proposition 2.1, the overall procedure takes time $|[D]|^{O(1)} \cdot g(|\bar{Q}|)$ for some computable triple exponential function $g : \mathbb{N} \rightarrow \mathbb{N}$.

### 3.2 TGDs as Integrity Constraints

When a set of TGDs is used as integrity constraints, the problem that we are interested in is simply a refinement of the standard query evaluation problem, with the additional promise that the input database satisfies the given set of TGDs. To this end, the evaluation problem is parameterized, not only with the query language in question, but also with the class of TGDs from which the constraints are coming. Formally, a constraint-query specification ($CQS$) over a schema $T$ is a pair $S = (\Sigma, q)$, where $\Sigma$ is a set of TGDs over $T$, the set of **integrity constraints**, and $q$ a $UCQ$ over $T$. We overload the notation and write $(C, Q)$ for the class of $CQS$s in which the set of integrity constraints is formulated in the class of TGDs $C$, and the query is coming from the class of queries $Q$. It will be clear from the context whether $(C, Q)$ is an OMQ language or a class of $CQS$s. The evaluation problem for $CQS$s follows:

**PROBLEM**: CQS-Evaluation($O$)

**INPUT**: A $CQS$ $S = (\Sigma, q(x)) \in O$ over a schema $T$, a $T$-database $D$ such that $D \models \Sigma$, and a tuple $\bar{c} \in \text{dom}(D)^{|\bar{x}|}$.

**QUESTION**: Is it the case that $\bar{c} \in q(D)$?

We are also interested in the parameterized version of the above problem, which we call $p$-CQS-Evaluation($O$), with the parameter being the size of the $CQS$ $S$. Therefore, $p$-CQS-Evaluation($O$) is in FPT if it can be solved in time $|[D]|^{O(1)} \cdot f(|\bar{Q}|)$ for some computable function $f : \mathbb{N} \rightarrow \mathbb{N}$.

Recall that the evaluation problem for $CQS$s is $NP$-hard [17], while its parameterized version is $W[1]$-hard [31], even for schemas of bounded arity. These lower bounds trivially transfer to CQS-Evaluation($C, Q$) for any $C$ as we can choose the set of constraints to be empty. Evaluation of UCQs is in $NP$, while the parameterized version is in $W[1]$, and also these bounds transfer since we can simply ignore the constraints for evaluating the query. It should thus also be clear that bounding the treewidth of $CQS$s has the same positive effect on evaluating $CQS$s as on $CQ$ evaluation (see Proposition 2.1), i.e., CQS-Evaluation($C, UCQ_k$) is in PTime.

### 4 Semantic T likeness

A seminal result by Grohe precisely characterizes the (recursively enumerable) classes of $CQS$s over schemas of bounded arity that can be evaluated in polynomial time (under the assumption that $P \neq W[1]$). In fact, this result shows that the classes of $CQS$s over schemas of bounded arity that can be evaluated in polynomial time
are precisely those that are semantically tree-like, or, more formally, are of bounded treewidth modulo equivalence. Moreover, the result states that fpt does not add anything to standard tractability in the considered setting. Before giving the statement, let us formalize the notion of bounded treewidth modulo equivalence. Recall that two CQSs q, q' over a schema S are equivalent if, for every S-database D, \( q(D) = q'(D) \). For each \( k \geq 1 \), let \( \text{CQ}_k^\text{eq} \) be the class of all CQSs that are equivalent to a CQ from \( \text{CQ}_k \). Grohe's result follows:

**Theorem 4.1 (Grohe's Theorem [26]).** Fix \( r \geq 1 \). Let \( Q \) be a recursively enumerable class of CQSs over schemas of arity \( r \). The following are equivalent, assuming \( \text{FPT} \neq \text{W}[1] \):

1. The evaluation problem for \( Q \) is in PTIME.
2. The evaluation problem for \( Q \) in \( \text{FPT} \).
3. There is \( k \geq 1 \) such that \( Q \subseteq \text{CQ}_k^\text{eq} \).

If either statement is false, then evaluation for \( Q \) is \( \text{W}[1] \)-hard.

Interestingly, it is decidable whether a CQ is equivalent to one of treewidth \( k \). This is shown by exploiting the notion of core. Recall that the core of a CQ \( Q \) is a \( \subseteq \)-minimal subquery of \( Q \) that is equivalent to \( Q \). It is known that, for each \( k \geq 1 \), a CQ \( q \) belongs to \( \text{CQ}_k^\text{eq} \) iff its core is in \( \text{CQ}_k \), and that deciding this property is \( \text{NP} \)-complete [20]. There is also a natural generalization of this characterization and of Theorem 4.1 to the class of UCQs.

At this point, it is natural to ask whether it is possible to obtain a characterization of the classes of OMQs (resp., CQs) based on (frontier-)guarded TGDs that can be efficiently evaluated, in the same spirit as Grohe's Theorem. In fact, our main question is whether the natural generalization of the notion of bounded treewidth modulo equivalence from CQs to OMQs (resp., CQs) yields a decidable notion that exhausts tractability or fixed-parameter tractability of OMQ-Evaluation (resp., CQs-Evaluation), as in the case of Grohe’s Theorem.

### 4.1 Semantic Tree-likeness for OMQs

We first concentrate on OMQs and introduce the notion of \( \text{UCQ}_k^\text{eq} \)-equivalence, meaning that an OMQ can be rewritten into an equivalent one where the UCQ belongs to \( \text{UCQ}_k \). Let us clarify that for OMQs the notion of equivalence is not applied at the level of the actual query, but to the whole OMQ. Formally, given two OMQs \( Q \) and \( Q' \), both with data schema \( S \), \( Q \) is contained in \( Q' \), written \( Q \subseteq Q' \), if \( Q(D) \subseteq Q'(D) \) for every \( S \)-database \( D \). We then say that \( Q \) and \( Q' \) are equivalent, denoted \( Q \equiv Q' \), if \( Q \subseteq Q' \) and \( Q' \subseteq Q \).

In what follows, let \( C \) be a class of TGDs, i.e., \( C \subseteq \text{TGD} \).

**Definition 4.2 (UCQ\(_k^\text{eq}\)-equivalence for OMQs).** An OMQ \( Q = (S, \Sigma, q) \in (C, \text{UCQ}) \) is \( \text{UCQ}_k^\text{eq} \)-equivalent, for \( k \geq 1 \), if there exists an OMQ \( Q' = (S, \Sigma', q') \in (C, \text{UCQ}) \) such that \( Q \equiv Q' \). Given an OMQ language \( O = (C, \text{UCQ}) \) with \( k \geq 1 \), let \( O_k^\text{eq} \) be the class of all OMQs from \( O \) that are \( \text{UCQ}_k^\text{eq} \)-equivalent.

According to the above definition, we are allowed to rewrite both the ontology and the UCQ. It is conceptually meaningful though to consider also the setting where the ontology cannot be altered. This leads to the uniform version of \( \text{UCQ}_k^\text{eq} \)-equivalence.

**Definition 4.3 (Uniform UCQ\(_k^\text{eq}\)-equivalence for OMQs).** An OMQ \( Q = (S, \Sigma, q) \in (C, \text{UCQ}) \) is uniformly \( \text{UCQ}_k^\text{eq} \)-equivalent, for \( k \geq 1 \), if there is \( Q' = (S, \Sigma', q') \in (C, \text{UCQ}) \) such that \( Q \equiv Q' \). For an OMQ language \( O = (C, \text{UCQ}) \) and \( k \geq 1 \), let \( O_k^\text{eq} \) be the class of all OMQs from \( O \) that are uniformly \( \text{UCQ}_k^\text{eq} \)-equivalent.

The next example shows that both the ontology and the data schema can have an impact on the treewidth.

**Example 4.4.** We first illustrate that the ontology can have an impact on the treewidth. Consider the OMQ \( Q_1 = (S, \Sigma, q) \), where

\[
\begin{align*}
S &= \{ R_1, R_2, R_3, R_4, P \} \\
\Sigma &= \{ R_2(x) \rightarrow R_4(x) \} \\
qu() &= P(x_2, x_1) \wedge P(x_4, x_1) \wedge P(x_2, x_3) \wedge P(x_4, x_3) \wedge R_1(x_1) \wedge R_2(x_2) \wedge R_3(x_3) \wedge R_4(x_4).
\end{align*}
\]

Let us clarify that for \( q \in \text{UCQ}_2 \), we get that \( Q_1 \subseteq \text{UCQ}_2 \), and thus can easily be seen to not belong to \( \text{UCQ}_2^\text{eq} \). However, \( Q_1 \) is equivalent to the OMQ \( (S, \Sigma, q') \), where

\[
qu'(x) = P(x_2, x_1) \wedge P(x_2, x_3) \wedge R_1(x_1) \wedge R_2(x_2) \wedge R_3(x_3).
\]

Since \( q' \in \text{UCQ}_2 \), we get that \( Q_1 \in (C, \text{UCQ})_2^\text{eq} \).

We proceed to show that the data schema can also have an impact. Consider the OMQ \( Q_2 = (S', \Sigma', q) \) with full data schema, where

\[
\Sigma' = \{ S(x) \rightarrow R_1(x), \ S(x) \rightarrow R_3(x) \}.
\]

It is not hard to see that \( Q_2 \) does not belong to \( (C, \text{UCQ})_2^\text{eq} \). If, however, the predicate \( R_1 \) is omitted from the signature, then \( Q_2 \) is equivalent to the OMQ \( (S, P, R_2, R_4, \Sigma') \), where \( q''() = P(x_2, x_1) \wedge P(x_2, x_3) \wedge R_2(x_2) \wedge R_4(x_4) \) and thus, it belongs to \( (C, \text{UCQ})_2^\text{eq} \).

### 4.2 Semantic Tree-likeness for CQs

We now introduce \( \text{UCQ}_k^\text{eq} \)-equivalence for CQs. For this setting, only the uniform version is relevant. Indeed, given a CQS \( S = (\Sigma, q) \), by altering the set of integrity constraints \( \Sigma \), we effectively change the semantics of \( S \) which is, of course, not our intention. Given two CQs \( S = (\Sigma, q) \) and \( S' = (\Sigma', q') \) over a schema \( T \), we say that \( S \) is contained in \( S' \), denoted \( S \subseteq S' \), if \( q(D) \subseteq q'(D) \) for every \( T \)-database \( D \) that satisfies \( S \). We then say that \( S \) is equivalent to \( S' \) if \( S \subseteq S' \) and \( S' \subseteq S \). We may also write \( q \equiv q' \) (resp., \( q \equiv q' \)) for the fact that \( S \subseteq S' \) (resp., \( S' \subseteq S \)). We recall a known result about containment among CQs, which will be useful for our analysis.

**Proposition 4.5.** Let \( S_1 = (\Sigma, q_1(x)) \) and \( S_2 = (\Sigma, q_2(y)) \) be CQs from \( \text{TGD, UCQ} \) with \( |x| = |y| \). Then \( S_1 \subseteq S_2 \) if and only if for each \( p_1 \in q_1 \), there exists \( p_2 \in q_2 \) such that \( x \in p_2(\text{chase}(p_1, \Sigma)) \).

The notion of uniform \( \text{UCQ}_k^\text{eq} \)-equivalence for CQs follows:

**Definition 4.6 (Uniform UCQ\(_k^\text{eq}\)-equivalence for CQs).** A CQ \( S = (\Sigma, q) \) from \( (C, \text{UCQ}) \) is uniformly \( \text{UCQ}_k^\text{eq} \)-equivalent, for \( k \geq 1 \), if there exists a CQS \( S' = (\Sigma, q') \) from \( (C, \text{UCQ}) \) such that \( S \equiv S' \). Given a class of CQs \( O \), for \( k \geq 1 \), let \( O_k^\text{eq} \) be the class of all CQs from \( O \) that are uniformly \( \text{UCQ}_k^\text{eq} \)-equivalent.

Observe that the first part of Example 4.4 also works when \( (\Sigma, q) \) is viewed as a CQS. This illustrates the fact that integrity constraints can have an impact on the treewidth.

\textsuperscript{2} We overload the notation again. It will be clear from the context whether \( O_k^\text{eq} \) is an OMQ language or a class of CQs. We also avoid the superscript \( k \) since for CQs we only consider the uniform version of \( \text{UCQ}_k^\text{eq} \)-equivalence.
5 OUR RESULTS IN A NUTSHELL

Having the relevant notions in place, we can now provide an answer to our main question, that is, whether (uniform) \( UCQ \)-equivalence of OMQs (resp., uniform \( UCQ \)-equivalence of CQSs) exhausts tractability and fixed-parameter tractability of OMQ-Evaluation (resp., CQS-Evaluation) in the same spirit as Grohe’s Theorem. In this section, we give an overview of our results, observe interesting connections between the OMQ and CQS settings, and study the associated meta problems. Further details are deferred to the subsequent sections.

5.1 The Guarded Case

Ontology-mediated Queries. We start with our results on OMQs based on guarded TGDs. We first ask whether (uniform) \( UCQ \)-equivalence is decidable for \((G, UCQ)\).

**Theorem 5.1.** For each \( k \), deciding whether a given OMQ \( Q \) from \((G, UCQ)\) over \( T \) with \( k \geq \text{ar}(T) - 1 \) is (uniformly) \( UCQ \)-equivalent is \( 2\text{ExpTime} \)-complete. If existant, an OMQ \( Q' \in (G, UCQ) \) such that \( Q \equiv Q' \) can be computed in double exponential time.

The above complexity result exploits a characterization of when an OMQ \( Q \) from \((G, UCQ)\) is (uniformly) \( UCQ \)-equivalent, which in turn relies on what we call a \( UCQ_k \)-approximation, that is, an approximation of \( Q \) from below in terms of an OMQ from \((G, UCQ)\). In a nutshell, a \( UCQ_k \)-approximation of \( Q = (S, \Sigma, q) \) is an OMQ \( Q_k = (S', \Sigma_k, q_k) \), where \( q_k \in \mathcal{Q}_k \), that behaves like \( Q \) over \( S \)-databases of treewidth at most \( k \), i.e., for every \( S \)-database \( D \) of treewidth at most \( k \), \( Q(D) \equiv Q_k(D) \). It follows that \( Q_k \) is equivalent to \( Q \) if and only if \( Q \) is \( UCQ_k \)-equivalent. Let us say that the definition of \( UCQ_k \)-approximation in our setting is significantly more involved than in the case of description logics [7] because there the chase only generates structures of treewidth one.

**Proposition 5.2.** Let \( Q \) be an OMQ from \((G, UCQ)\) over \( T \), and let \( k \geq \text{ar}(T) - 1 \). The following are equivalent:

1. \( Q \) is \( UCQ_k \)-equivalent.
2. \( Q \) is uniformly \( UCQ_k \)-equivalent.
3. \( Q \equiv Q_k \).

Since \( UCQ_k \)-equivalence and uniform \( UCQ_k \)-equivalence turn out to be equivalent, we henceforth only use \( UCQ_k \)-equivalence.

Note that Proposition 5.2 also provides an approach to deciding \( UCQ_k \)-equivalence, and thus to establishing Theorem 5.1: compute the \( UCQ_k \)-approximation \( Q_k \) of \( Q \) and accept if \( Q \subseteq Q_k \) (note that \( Q_k \subseteq Q \) holds always); otherwise, reject. We can show that \( Q_k \) can be computed in double exponential time and it is known that OMG containment for \((G, UCQ)\) can be decided in double exponential time [6]. A naive use of these observations yields only a \( 4\text{ExpTime} \) upper bound, but it can be improved to \( 2\text{ExpTime} \).

The lower bound is inherited from [7], where the same problem for OMQs based on DLs has been studied.

We remark that the case \( k < \text{ar}(T) - 1 \), excluded in Theorem 5.1 and Proposition 5.2, is somewhat esoteric as it involves relation symbols whose arity is so high that they cannot be part of the UCQ in a \( UCQ_k \)-approximation (unless variables are reused). In fact, one can provide a concrete example which illustrates that this case is significantly different from the case \( k \geq \text{ar}(T) - 1 \); in particular, Proposition 5.2 is provably wrong.

We now state our main result concerning guarded OMQs which shows that \( UCQ_k \)-equivalence characterizes \( \text{FPT} \) for classes of OMQs from \((G, UCQ)\) over schemas of bounded arity.

**Theorem 5.3 (Main Result I).** Fix \( r \geq 1 \). Let \( O \) be a recursively enumerable class of OMQs from \((G, UCQ)\) over a schema of arity \( r \). The following are equivalent, assuming \( \text{FPT} \neq W[1] \):

1. \( p\text{-OMQ-Evaluation}(O) \) is in \( \text{FPT} \).
2. There is \( k \geq 1 \) such that \( O \subseteq (G, UCQ)_{k} \).

If either statement is false, then \( p\text{-OMQ-Evaluation}(O) \) is \( W[1] \)-hard.

The easy direction is (2) implies (1), which exploits Theorem 5.1, and the third item of Proposition 3.3. The hard task is to show that (1) implies (2). To this end, we assume that \( \text{FPT} \neq W[1] \), it suffices to show the following lower bound, which is our main technical result on OMQs based on guarded TGDs:

**Theorem 5.4.** Fix \( r \geq 1 \). Let \( O \) be a recursively enumerable class of OMQs from \((G, UCQ)\) over a schema of arity \( r \), and, for each \( k \geq 1 \), \( O \subseteq (G, UCQ)_{k} \). Then, \( p\text{-OMQ-Evaluation}(O) \) is \( W[1] \)-hard.

In view of Proposition 3.2, which states that, for each \( k \geq 1 \), OMQ-Evaluation\((G, UCQ)_{k}\) is \( \text{ExpTime} \)-hard, it is not surprising that Theorem 5.3 does not state a pure tractability result. One may think that the above result can be easily obtained by using Grohe’s construction underlying the \( \text{fpt} \)-reduction from \( p\text{-Clique} \) that establishes the lower bound of Theorem 4.1. However, the fact that the ontology can introduce additional relations that are not part of the data schema causes serious challenges. A detailed sketch of our proof is presented in Section 6.

Constraint Query Specifications. We now turn our attention to CQSs based on guarded TGDs. Interestingly, there is a strong connection between CQSs and OMQs with full data schema, which allows us to transfer results from the OMQ to the CQS setting. Recall that a full data schema consists of all the predicates occurring in the ontology and the UCQ. Note that we can naturally convert a CQS \( S = (\Sigma, q) \) over a schema \( S \) into the OMQ \((S, \Sigma, q)\) that has full data schema, denoted \( \text{omq}(S) \). The following result relates the \( UCQ \)-equivalence of CQSs to the \( UCQ_k \)-equivalence of OMQs.

**Proposition 5.5.** Consider a CQS \( S \in (G, UCQ) \) over \( T \). For each \( k \geq \text{ar}(T) \), the following are equivalent:

1. \( S \) is uniformly \( UCQ_k \)-equivalent.
2. \( \text{omq}(S) \) is \( UCQ_k \)-equivalent.

From Theorem 5.1 and Proposition 5.5, we get a \( 2\text{ExpTime} \) upper bound for deciding uniform \( UCQ_k \)-equivalence for guarded CQSs, while the lower bound is inherited from [8]. As in Theorem 5.1.1, \( k \) should be greater than the arity of the schema, minus one.

**Theorem 5.6.** Let \( S \) be a CQS from \((G, UCQ)\) over a schema \( T \). For each \( k \geq \text{ar}(T) \), deciding whether \( S \) is uniformly \( UCQ_k \)-equivalent is \( 2\text{ExpTime} \)-complete.

Our main result concerning guarded CQSs shows that uniform \( UCQ_k \)-equivalence characterizes tractability and \( \text{fpt} \) for classes of CQSs from \((G, UCQ)\) over schemas of bounded arity.

**Theorem 5.7 (Main Result II).** Fix \( r \geq 1 \). Let \( O \) be a recursively enumerable class of CQSs from \((G, UCQ)\) over a schema of arity \( r \). The following are equivalent, assuming \( \text{FPT} \neq W[1] \):


(1) CQS-Evaluation(∅) is in PTIME.
(2) p-CQS-Evaluation(∅) is in FPT.
(3) There is $k \geq 1$ such that $\emptyset \subseteq (G, UCQ)^k$.

If either statement is false, then p-CQS-Evaluation(∅) is $W[1]$-hard.

The fact that (3) implies (1) is shown in [8], while (1) implies (2) holds by definition. The interesting task is to show that (2) implies (3). To this end, we exploit the subsequent result which relates the evaluation of CQSs to the evaluation of OMQs. Given a class $\mathcal{O}$ of CQSs, we write $omq(O)$ for the class of OMQs $\{omq(S) \mid S \in \mathcal{O}\}$.

**Proposition 5.8.** Consider a class $\mathcal{O}$ of CQSs from $(G, UCQ)$. There exists an fpt-reduction from p-OMQ-Evaluation$(omq(O))$ to p-CQS-Evaluation$(O)$.

The reduction exploits the fact that guarded TGDs are finitely controllable, which means that $OMQ$-Evaluation$(G, UCQ)$ coincides with $OMQ$-Evaluation$_{\text{fin}}(G, UCQ)$ that considers only finite models [4]. Note that the reduction does not extend to frontier-guarded TGDs since it requires the evaluation of TGD bodies to be fpt. Actually, no such reduction is possible for frontier-guarded TGDs (unless FPT = $W[1]$) since there exists a class $\mathcal{O}$ of CQSs from $(FG, UCQ)$ such that p-CQS-Evaluation$(omq(O))$ is $W[1]$-hard, while p-CQS-Evaluation$(O)$ is in FPT: $\mathcal{O}$ consists of CQSs $(\Sigma, 3\exists \mathcal{R}(x))$, where $\Sigma$ contains TGDs of the form $\varphi(x, y) \rightarrow R(x)$, i.e., frontier-guarded TGDs with only one frontier variable. Now, by Theorem 5.4, Proposition 5.5, and Proposition 5.8, we get the following result, which establishes the direction (2) implies (3) of Theorem 5.7 (since we assume that FPT $\neq W[1]$).

**Theorem 5.9.** Fix $r \geq 1$. Let $\mathcal{O}$ be a recursively enumerable class of CQSs from $(G, UCQ)$ over a schema of arity $r$, and, for each $k \geq 1$, $\emptyset \subseteq (G, UCQ)^k$. Then, p-CQS-Evaluation$(O)$ is $W[1]$-hard.

### 5.2 The Frontier-guarded Case

By item (2) of Proposition 3.3, the notion of (uniform) $UCQ_k$-equivalence does not provide a characterization of fpt for classes of OMQs from $(FG, UCQ)$. Such a characterization results for OMQs based on frontier-guarded TGDs via an alternative notion of “being equivalent to an OMQ of low treewidth” remains an interesting open problem. The situation is different for querying in the presence of constraints. In fact, uniform $UCQ_k$-equivalence allows us to characterize tractability and fpt for classes of CQSs from $(FG, UCQ)$ over schemas of bounded arity $r$ provided that the number of atoms in the head of TGDs is bounded by an integer $m$. The latter is required in our proof for reasons that are explained in Section 7.

The first question is whether uniform $UCQ_k$-equivalence for CQSs based on frontier-guarded TGDs, with at most $m \geq 1$ head atoms, over schemas of arity $r \geq 1$, is decidable. We write $FG_m$ for the class of frontier-guarded TGDs with at most $m$ head atoms.

**Theorem 5.10.** Fix $r, m \geq 1$. Let $S$ be a CQS from $(FG_m, UCQ)$ over a schema of arity $r$. For each $k \geq (r \cdot m - 1)$, deciding whether $S$ is uniformly $UCQ_k$-equivalent is $2\text{ExpTime}$-complete.

The above result exploits a characterization, analogous to Proposition 5.2, of when a CQS from $(FG_m, UCQ)$ over a schema of arity $r$ is uniformly $UCQ_k$-equivalent via a suitable notion of $UCQ_k$-approximation. The fact that we consider schemas of arity $r$, and frontier-guarded TGDs with at most $m$ atoms in the head, allows us to show that chasing a database of treewidth $k$, with $k \geq r \cdot m - 1$, will lead to an instance that has treewidth $k$. This in turn allows us to adopt a notion of $UCQ_k$-approximation that is significantly simpler than the one for guarded OMQs in Proposition 5.2. A contraction of a CQ $q$ is a CQ obtained from $q$ by identifying variables. When an answer variable $x$ is identified with a non-answer variable $y$, the resulting variable is $x$, while the identification of two answer variables is not allowed. The $UCQ_k$-approximation of a CQS $S = (\Sigma, q) \in (FG, UCQ)$ is the CQS $S^a_k = (\Sigma, q^a_k)$, where $q^a_k$ is the UCQ that consists of all contractions of a CQ from $q$ that belong to $CQ_k$. We can then show the following:

**Proposition 5.11.** Fix $r, m \geq 1, k \geq r \cdot m - 1$. Let $S$ be a CQS from $(FG_m, UCQ)$ over a schema of arity $r$. The following are equivalent:

1. $S$ is uniformly $UCQ_k$-equivalent.
2. $S \equiv S^a_k$.

From the above result, we get a procedure for deciding uniform $UCQ_k$-equivalence: compute the $UCQ_k$-approximation $S^a_k = (\Sigma, q^a_k)$ of $S = (\Sigma, q)$, and accept if $S \subseteq S^a_k$ (the other direction holds by construction); otherwise, reject. Clearly, $S^a_k$ can be computed in exponential time, while $q^a_k$ consists, in general, of exponentially many CQs, each of polynomial size. By Proposition 4.5, $S \subseteq S^a_k$ iff for each CQ $p(\bar{x}) \in q$, there exists $p' \in q^a_k$ such that $\bar{x} \in p'(\text{chase}(p, \Sigma))$. Moreover, by Proposition 3.1 and item (1) of Proposition 3.2, we get that checking whether $\bar{x} \in p'(\text{chase}(p, \Sigma))$ is feasible in double exponential time. Summing up, to check whether $S \subseteq S^a_k$ we need to perform exponentially many checks, while each check is feasible in double exponential time. This leads to the desired $2\text{ExpTime}$ upper bound, while a matching lower bound is inherited from [8].

Our main result concerning frontier-guarded CQSs follows:

**Theorem 5.12 (Main Result III).** Fix $r, m \geq 1$. Let $\mathcal{O}$ be a recursively enumerable class of CQSs from $(FG_m, UCQ)$ over a schema of arity $r$. The following are equivalent, assuming FPT $\neq W[1]$:

1. CQS-Evaluation$(O)$ is in PTIME.
2. p-CQS-Evaluation$(O)$ is in FPT.
3. There is $k \geq 1$ such that $\emptyset \subseteq (FG_m, UCQ)^k$.

If either statement is false, then p-CQS-Evaluation$(O)$ is $W[1]$-hard.

As for Theorem 5.7, the fact that (3) implies (1) is shown in [8], while (1) implies (2) holds by definition. Thus, the non-trivial task is to show that (2) implies (3). To this end, since we assume that FPT $\neq W[1]$, it suffices to show the following, which is our main technical result on CQSs based on frontier-guarded TGDs:

**Theorem 5.13.** Fix $r, m \geq 1$. Let $\mathcal{O}$ be a recursively enumerable class of CQSs from $(FG_m, UCQ)$ over a schema of arity $r$, and, for each $k \geq 1$, $\emptyset \subseteq (FG, UCQ)^k$. Then, p-CQS-Evaluation$(O)$ is $W[1]$-hard.

Unlike Theorem 5.9, the above result cannot benefit from results on OMQs since (uniform) $UCQ_k$-equivalence does not characterize fpt for OMQs based on frontier-guarded TGDs. It is shown via a reduction from p-Clique by following the same approach as for the lower bound of Grohe’s Theorem, and exploiting the fact that frontier-guarded TGDs are finitely controllable. Let us stress, however, that our fpt-reduction is not a merely adaptation of Grohe’s...
We provide a proof sketch of Theorem 5.4, illustrating only the \( \mathcal{O} \) to simplify this proof sketch, assume that no rewriting is needed heads. Although all \( \mathcal{O} \) taken into account by a minor change in the construction. He considers the case for the homomorphism \( G \) defined in the usual way, see, e.g., [26]. When \( k \) is understood from the context, we use \( K \) to denote \( \mathcal{O} \). We say that a database \( D \) is connected if its graph \( G_D \) is connected. A constant \( a \in \text{dom}(D) \) is isolated in \( D \) if there is only a single atom \( R(a) \in D \) with \( a \in \bar{a} \).

Theorem 6.1 (Grohe). Given an undirected graph \( G \), a \( k \geq 1 \), a connected \( S \)-database \( D \), and a set \( A \subseteq \text{dom}(D) \) such that the restriction \( G_D^{A} \) of the Gaifman graph of \( D \) to vertices \( A \) contains the \( k \times k \)-grid as a minor, one can construct in time \( f(k) \cdot \text{poly}(|G|, |D|) \) an \( S \)-database \( D_G \) with \( \text{dom}(D_G) \setminus \bar{A} \subseteq \text{dom}(D_G) \) such that:

1. there is a surjective homomorphism \( h_0 \) from \( D_G \) to \( D \) that is the identity on \( \text{dom}(D) \setminus \bar{A} \),
2. \( G \) contains a \( k \)-clique if \( f \) is a homomorphism \( h \) from \( D \) to \( D_G \) such that \( h \) is the identity on \( \text{dom}(D) \setminus \bar{A} \) and \( h_0(h(\cdot)) \) is the identity, and
3. if \( R(a_1, \ldots, a_n) \in D, h_0(h_i) = a_i \) for each \( i \in [n] \), and \( \{h_i \mid a_i \text{ non-isolated in } D \} \) is a clique in the Gaifman graph of \( G_D \), then \( D_G \) contains an atom \( R(c_1, \ldots, c_n) \) where \( c_i = h_i \) if \( a_i \) is non-isolated in \( D \) and \( h_0(c_i) = a_i \) for each \( i \in [n] \).

The condition in item (3) is not considered in [26], but can easily be verified to hold. It can be viewed as an ontorequirement for the homomorphism \( h_0 \) from item (1) as it says that for certain atoms \( R(\bar{a}) \in D \), we must find certain atoms \( R(\bar{e}) \in D_G \) with \( h_0(\bar{e}) = \bar{a} \). The set \( A \) is not present in Grohe’s construction (that is, he considers the case \( A = \text{dom}(D) \)), but we can show that it can be taken into account by a minor change in the construction.

For the proof of Theorem 5.4, we are given a recursively enumerable class of OMIs \( \mathcal{O} \subseteq (G, \text{UCQ})^k \) such that \( \mathcal{O} \not\subseteq (G, \text{UCQ})^\kappa \) for all \( k \geq 1 \). As a preliminary, we show that for every \( \mathcal{O} \in \mathcal{O} \), we can effectively find an equivalent OMQ \( Q' \in (G \cup \text{FULL}, \text{UCQ}) \) over the same extended schema, where \( \text{FULL} \) is the class of full TGDs, i.e., TGDs without existentially quantified variables. In other words, we can rewrite away the existential quantifiers in TGD heads. Although \( Q' \) need not be in \( \mathcal{O} \), we can switch between \( Q \) and \( Q' \) in the proof whenever we only care about \( Q' \)’s semantics. To simplify this proof sketch, assume that no rewriting is needed as all ontologies in \( \mathcal{O} \) are already from \( \mathcal{O} \cap \text{FULL} \), and assume that \( \mathcal{O} \) contains only Boolean OMIs where the actual queries are CQs.

The proof is by pft-reduction from \( p \)-Clique, a \( W[1] \)-hard problem. Assume that \( G \) is an undirected graph and \( k \geq 1 \) a clique size. We aim to construct a database \( D_G \) and find an OMQ \( Q \in \mathcal{O} \) where

\[ \text{(*) } G \text{ has a } k \text{-clique iff } D_G \models Q. \]

We first describe how to find \( Q \). By Robertson and Seymour’s Excluded Grid Theorem, there exists an \( f \) such that every graph of treewidth exceeding \( f \) contains the \( k \times k \)-grid as a minor [32]. We may assume, w.l.o.g., that \( f \) exceeds the fixed arity \( r \) of relation symbols that occur in \( \mathcal{O} \). Since \( \mathcal{O} \not\subseteq (G, \text{UCQ})^\kappa \), there is an OMQ \( Q = (S, \Sigma, q) \) from \( \mathcal{O} \) that is not in \( (G, \text{UCQ})^\kappa \), and we can find \( Q \) by enumerating \( \mathcal{O} \) and relying on Theorem 5.1. It thus remains to construct \( D_G \). In the special case where \( \Sigma \) is empty and \( S \) is full, this can be done by replacing \( q \) with its core and applying Theorem 6.1, using \( D[q] \) for \( D \). Then, item (2) of that theorem corresponds to (*).

In the general case, however, we cannot use \( D[q] \) for \( D \) since \( D[q] \) might use relation symbols that are not in \( S \) and thus so would the resulting \( D_G \). This, in fact, is what makes our proof rather delicate and difficult. The general idea is to replace \( D[q] \) with an \( S \)-database \( D_0 \) such that \( D_0 \models Q \) and \( (D_0)^r \not\subseteq Q \), where \( (D_0)^r \) denotes the unraveling of \( D_0 \) into a structure of treewidth \( f \). Ideally, we would like to apply the Grohe construction to \( D_0 \) instead of \( D[q] \). However, we of course still have to attain (*), which is formulated in terms of \( Q \) (and thus, in terms of \( \Sigma \) and \( q \)) rather than \( D_0 \). This means that we need more from \( D_0 \) than just the stated properties and, in fact, we want chase \((D_0, \Sigma) \) and \( q \) to be structurally as similar as possible. The following example illustrates a structural dissimilarity.

Example 6.2. Assume that \( \Sigma \) is empty, that \( S \) is full, and that \( q \) is the 3 \( \times \) 4-grid, that is,

\[ \bigwedge_{i \in \{1,2,3\}} X(x_i,j,x_{i+1,j}) \land \bigwedge_{i \in \{1,2,3\}} Y(x_i,j,x_{i,j+1}). \]

Moreover, let \( D_0 \) be the following database, the 3\( \times \)3-grid augmented with reflexive \( X \)-loops in the rightmost column:

\[ \{X(a_{1,j}, a_{1,j+1}) \mid i \in [1,2], j \in \{1,2,3\}\} \cup \{Y(a_{i,j}, a_{i+1,j}) \mid i \in [1,2,3], j \in \{1,2\}\} \cup \{X(a_{3,j}, a_{3,j}) \mid j \in \{1,2,3\}\}. \]

Clearly, \( D_0 \models Q \) and \( (D_0)^r \not\subseteq Q \).

Example 6.2 is a particularly simple case: the ontology is empty, the data schema is full, and the query is a core. We could in fact choose \( D_0 = D[q] \). In general, however, it might not be possible to make \( D_0 \) and \( D[q] \) isomorphic, and there might not even be some \( D_0 \) such that \( q \) maps injectively to \( D_0 \). What we do is choosing \( D_0 \) such that the following holds, where \( \bar{p} \) means that \( D \models \bar{p} \) and all homomorphisms from \( p \) to \( D \) are injective (‘io’ stands for ‘injectively only’):

\[ \text{(**) if chase}(D_0, \Sigma) \models \bar{p}_q, \text{ where } q_\Sigma \text{ is a contraction of } q, \text{ then there is no } S \text{-database } D \text{ and contraction } q' \text{ of } q \text{ such that } D \text{ homorphically maps to } D_0 \text{, chase}(D, \Sigma) \models \bar{p}_q', \text{ and } q_\Sigma \neq q'_\Sigma \text{ is a contraction of } q'_\Sigma. \]

In Example 6.2, \( D_0 \) does not satisfy this condition as witnessed by choosing \( D = D[q] \), where \( q_{\Sigma} \) is the contraction of \( q \) that identifies
Further, let \( f \) be the database
\[
\{ (a, b), (b, c), (c, a) \}
\]
and thus there is a contraction \( \bar{q} \) of \( f \). We remark that our proof is considerably more involved than the original proof for the guarded case. In particular, the presence of \( f \) might contain parts that are superfluous for our purposes.

\section{Constraint Query Specifications}

We now discuss the fpt-reduction of Proposition 5.8, which is underlying Theorem 5.9. Recall that this reduction relies on the fact that guarded TGDs are finitely controllable. Recall also that the notion of finite controllability is crucial for establishing our main technical results on CQs based on frontier-guarded TGDs (Theorem 5.13), for which more details are given in the next section. Let us then give some useful details about the TGDs.

Finite Controllability. Given an \( S \)-database \( D \) and a set of TGDs over \( S \), we write \( \text{fmods}(D, S) \) for the set of all finite models of \( D \) and \( S \). Finite controllability guarantees that, for computing the set of tuples that are answers to a UCQ \( q \) over every finite model of \( D \) and \( S \), i.e., the set \( \bigcap_{M \in \text{fmods}(D, S)} q(M) \), it suffices to evaluate \( q \) over the (possibly infinite) instance chase \( (\bar{D}, \Sigma) \).

Definition 6.4 (Finite Controllability). A class \( C \) of TGDs is finitely controllable if, for every \( S \)-database \( D \), set \( \Sigma \in C \) of TGDs over \( S \), and UCQ \( q \) over \( S \), \( q(\text{chase}(D, \Sigma)) = \bigcap_{M \in \text{fmods}(D, S)} q(M) \).

Actually, as we shall see, in our proofs we rely on a seemingly stronger property than finite controllability. Roughly speaking, this property states that, once we fix the number of variables that can appear in UCQs, let say \( n \geq 0 \), given a database \( D \), and a set \( \Sigma \) of TGDs, there exists a finite model \( M \) of \( D \) and \( \Sigma \), which depends on \( n \), that allows us to compute the set of tuples \( q(\text{chase}(D, \Sigma)) \), i.e., \( q(\text{chase}(D, \Sigma)) = q(M) \), no matter what the UCQ looks like.

Definition 6.5 (Strong Finite Controllability). Consider a class \( C \) of TGDs. We say that \( C \) is strongly finitely controllable if, for every \( S \)-database \( D \), set \( \Sigma \in C \) of TGDs over \( S \), and integer \( n \geq 0 \), there exists an instance \( M(D, \Sigma, n) \in \text{fmods}(D, S) \) such that, for each UCQ \( q \) that has at most \( n \) variables, \( q(\text{chase}(D, \Sigma)) = q(M(D, \Sigma, n)) \). If there is a computable function that takes as input \( D, \Sigma, n \), and outputs \( M(D, \Sigma, n) \), then we say that \( C \) is finitely realizable.

The next technical result, which is of independent interest, shows that finite and strong finite controllability are equivalent properties.

Lemma 6.6. Consider a class \( C \) of TGDs. It holds that \( C \) is finitely controllable iff \( C \) is strongly finitely controllable.

By using the above lemma and results on the guarded negation fragment of first-order logic (GNFO) from [5], we can show that:

Theorem 6.7. The class \( \mathcal{FQ} \) is strongly finitely controllable. Moreover, finite witnesses are realizable.

In particular, every satisfiable GNFO sentence has a finite model. This allows us to show that the class of frontier-guarded TGDs is finitely controllable, and thus, by Lemma 6.6, it is also strongly finitely controllable. Moreover, we know that if a GNFO sentence \( \phi \) has a finite model, then it has a finite model of size \( 2^{\text{poly}(|\phi|)} \). This allows us to show, in addition, that finite witnesses are realizable.

The FPT-Reduction. Let us now come back to Proposition 5.8. We want to show the following: given a class of CQs \( \mathcal{F} \) from \( \mathcal{G}, \) UCQ, there exists an fpt-reduction from \( p\text{-OMQ}\)-Evaluation (omq(\( \mathcal{G} \))) to \( p\text{-CQ}\)-Evaluation (omq(\( \mathcal{G} \))). Consider \( \mathcal{Q} = (\Sigma, \phi, \overline{\psi}) \) from omq(\( \mathcal{G} \)), an \( S \)-database \( D \), and a tuple \( \bar{a} \in \text{dom}(D)^{|\overline{\psi}|} \). We are going to construct an \( S \)-database \( D' \) such that \( D' \models \Sigma \) and \( \bar{a} \in \text{dom}(D)^{|\overline{\psi}|} \). We define \( D' \) as the database
\[
D \cup \{ (a, b) \in \text{chase}(D, \Sigma) \mid a \subseteq \text{dom}(D) \}.
\]
Let \( \bar{a} \) be the family of all maximal tuples \( \bar{b} \) over \( \text{dom}(D) \) that are guarded in \( D' \), i.e., there is an atom \( R(\bar{b}) \) in \( D' \) such that \( \bar{a} \subseteq \bar{b} \).
Fix an arbitrary tuple $\bar{a} \in A$. Since, by Theorem 6.7, the class $\mathbb{O}$ is strongly finitely controllable, and also finite witnesses are realizable, we can compute an instance $M(D^+_{|\bar{a}|}, \Sigma, n) \in \text{fmos}(D^+_{|\bar{a}|}, \Sigma)$, where $n$ is the number of variables in $q$, such that for each CQ $q'$ of arity $|\bar{a}|$ with at most $n$ variables, it holds that
\begin{align*}
(\ast) \quad \bar{a} \in q'(M(D^+_{|\bar{a}|}, \Sigma, n)) \iff \bar{a} \in q'(\text{chase}(D^+_{|\bar{a}|}, \Sigma)).
\end{align*}
W.l.o.g., we assume that $\text{dom}(M(D^+_{|\bar{a}|}, \Sigma, n)) \cap \text{dom}(M(D^+_{|\bar{a}|}, \Sigma, n)) \subseteq \text{dom}(D)$, for every two distinct tuples $\bar{a}, \bar{b} \in A$. The database $D^*$ is
\begin{align*}
D^* = \bigcup_{\bar{a} \in A} M(D^+_{|\bar{a}|}, \Sigma, n).
\end{align*}

The next lemma shows that $D^*$ is the desired database.

**Lemma 6.8.** It holds that:
\begin{enumerate}
\item $D^* \models \Sigma$.
\item $\exists \bar{c} \in q(\text{chase}(D, \Sigma)) \iff \exists \bar{c} \in q(D^*)$.
\item There exists a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $D^*$ can be constructed in time $|D|^{|p(1)|} \cdot f(||Q||)$ for some computable function $g : \mathbb{N} \rightarrow \mathbb{N}$, and on the observation that $|A|$, as well as the size of $D^*_{|\bar{a}|}$ for some $\bar{a} \in A$, do not depend on $D$.
\end{enumerate}

### 7 FRONTIER-GUARDEDNESS

In this final section, we focus on the proof of Theorem 5.13. Consider a recursively enumerable class $\mathbb{O}$ of CQs from $(\mathbb{P}G_m, \mathbb{UCQ})$ over a schema of arity $r$ such that, for each $k \geq 1$, $\mathbb{O} \notin (\mathbb{P}G_m, \mathbb{UCQ})^k$. Our goal is to show that CQS-Evaluation($\mathbb{O}$) is W[1]-hard. Note that, unlike in the proof of Theorem 5.4, we cannot eliminate the existential quantifiers. The reason is that in the OMQ case we only need an equivalent OMQ while in the CQS case we would have to replace the set of constraints $\Sigma$ from $\mathbb{P}G$ with an equivalent set $\Sigma'$ from $\mathbb{P}G \cap \text{FULL}$, but clearly such a $\Sigma'$ need not exist. As in the proof of Grohe’s Theorem, we provide an fpt-reduction from $p$-Clique (in fact, a restricted version of it as explained later). For the sake of clarity, here we discuss the case where $\mathbb{O}$ consists only of CQs $Q = (\Sigma, q)$ where $q$ is a Boolean CQ whose Gaifman graph is connected. This can be extended to non-Boolean UCQs consisting of non-connected CQs.

**A Variation of Grohe’s Database.** For technical reasons that we clarify later, we do not use Grohe’s database used in [26] to establish Theorem 4.1, nor the one that we used for showing Theorem 5.4. Instead, we use a subset of Grohe’s database that satisfies several good properties summarized in the next result.

**Theorem 7.1 (Grohe).** Given an undirected graph $G = (V, E)$, $k \geq 1$, databases $D, D'$ with $D \subseteq D'$, and a set $A \subseteq \text{dom}(D)$ such that the restriction $G^{D}_{|A|}$ of the Gaifman graph of $D$ to vertices $A$ contains the $k \times k$-grid as a minor, one can construct in time $f(k) \cdot \text{poly}(||G||, ||D'||)$ a database $D'' = D''(G, D, D', A)$ such that:
\begin{enumerate}
\item there is a surjective homomorphism $h_0$ from $D''$ to $D'$,
\item $G$ contains a $k$-clique iff there is a homomorphism $h$ from $D$ to $D''$ such that $h_0(h(\cdot))$ is the identity on $A$, and
\item if $D' \models \Sigma$, and every clique of size at most $3 \cdot r$ in $G$ is contained in a clique of size $3 \cdot r \cdot m$, then $D'' \models \Sigma$.
\end{enumerate}

The condition in item (3) is not considered by Grohe in [26], but can easily be verified to hold for our modified version of his database. It is crucial for our proof as we need to ensure that the database that the fpt-reduction constructs satisfies the given set $\Sigma$ of frontier-guarded TGDs.

**A Crucial Lemma.** Our proof borrows several ideas and techniques from the proof of Grohe’s Theorem. However, the adaptation of such techniques is non-trivial for a couple of reasons. First, we cannot construct an arbitrary database, but need to construct one that satisfies the given set $\Sigma$ of frontier-guarded TGDs. Second, the notion of core of a CQ, which is crucial for Grohe’s proof, cannot be directly used in our context. Recall that, for $k \geq 1$, a CQ $q$ belongs to $\mathbb{CQ}^k$ iff its core is in $\mathbb{CQ}_k$ [20]. This allows Grohe to work with the core of $q$ instead of $q$, and thus, exploit the property that a homomorphism from the core of $q$ to itself is injective. This is far from being true in the presence of constraints. Instead, we need a technical result, which, intuitively speaking, states that some subsets of our CQs behave like cores for the sake of our proof.

**Lemma 7.2.** Fix $t \geq r - m$. There exists a computable function that takes as input a CQS $S = (\Sigma, q) \in (\mathbb{P}G_m, \mathbb{CQ})$ that is not uniformly $\mathbb{CQ}^t$-equivalent, and outputs two CQs $p$ and $p'$, and a subset $X$ of the variables of $p$, such that the following hold:
\begin{enumerate}
\item $q \equiv_{\mathbb{CQ}} p$.
\item $D[p'] \models \Sigma$.
\item $D[p] \subseteq D[p']$.
\item $\forall(X) = X$, for every homomorphism $h$ from $p$ to $p'$.
\item The treewidth of $G^p_{\mathbb{CQ}}$ is larger than $t$.
\end{enumerate}

Let us now briefly comment on the reason why we have to bound the number of head atoms in frontier-guarded TGDs for our proof to work. In very rough terms, this is because the way in which $p'$ is constructed is by generating from $p$ the finite model $M = (D[p], \Sigma, n)$, for some $n \geq 0$, that is obtained from Definition 6.5 given the strong finite controllability of $\mathbb{P}G$ as established in Theorem 6.7. A property of this construction that is crucial for the proof of Lemma 7.2 is that the treewidth of $M$ is not larger than that of $D[p]$. The only way in which we can ensure this property to hold is by fixing the number of atoms in TGD heads.

Remarkably, this additional bound on the number of head atoms is not needed for obtaining Theorem 5.7. This is because there exists an fpt-reduction from the evaluation of CQs from $(G, \mathbb{UCQ})$ to CQs from the same class in which there are are no existential quantifiers in TGD heads – this is implicit in our proofs. When such a reduction is possible, the CQ $p'$ can be obtained in a simpler way: it suffices to chase $p$ using $\Sigma$, with $p$ being the CQ with a minimal number of atoms among all the CQs that are equivalent to $q$ w.r.t. $\Sigma$. Notice that no bound on the number of head atoms in the TGDs of $\Sigma$ is required in this case. Unfortunately, when $\Sigma \in \mathbb{P}G$, a reduction like the one described above is not possible. This prevents us from applying the previous idea since chase($p, \Sigma$) may be infinite.

**The FPT-Reduction.** We now proceed to explain how Lemma 7.2 is applied in order to prove Theorem 5.13. Let $(\Sigma, k)$ be an instance of $p$-Clique. It is easy to see that we can assume, w.l.o.g., that every
We have studied the limits of efficient query evaluation in the context of ontology-mediated querying and querying in the presence of constraints, focusing on guarded and frontier-guarded TGDs, and on UCQs as the actual queries. We have obtained novel efficiency characterizations in the spirit of Grohe’s well-known characterization of the tractable classes of CQs (without constraints).

An interesting open problem that emerges from this work is whether our main result on CQs based on frontier-guarded TGDs (Theorem 5.12) the bound on the number of head atoms can be dropped. Recall that our results, similarly to Grohe’s characterization, assume that the schema is of bounded arity. Establishing similar characterizations for schemas of unbounded arity is an open problem.

Acknowledgments. Barceló was supported by the Millennium Institute for Foundational Research on Data (IMFD Chile) and Fondecyt grant 1171019. Dalmau was supported by MICINN grant TIN2016-76573-C2-1P and Maria de Maeztu Units of Excellence Programme MDM-2015-0502. Feier and Lutz were supported by the ERC consolidator grant 647289 "CODA". Pieris was supported by the EPSRC grant EP/S003800/1 "EQUID".

REFERENCES

A PROOF OF PROP 3.3

The first two items are easy, and have been already discussed in the main body of the paper. Here we focus on item (3). The proof relies on two technical lemmas.

The first such lemma, which is implicit in [15], states that, given an S-database D and an OMQ Q = (S, Σ, q) from (L, UCQ), Q(D) coincides with the evaluation of q over an initial finite portion C of chase(D, Σ). To formalize this, we need then notion of chase level. Let s = 0 0 → 1 l → I 2 . . . be a chase sequence for a database D under a set Σ of TGDs, where σ_1 = φ_1(ā, ȳ) → ∃xφ_2(ā, x). The s-level of an atom α ∈ chase(D, Σ) is inductively defined as follows: level_s(α) = 0 if α ∈ D, and level_s(α) = t if max{level_r(β) | β ∈ φ_1(ā_i, ȳ_i−1)) = t for 0 < i < r, where i > 0 is such that σ_r ∈ I_t and σ_r /∈ ∪_{0≤j<i} I_j. Let chase^t(D, Σ) be the instance \{α ∈ \cup_{i≥0} I_i | level_s(α) ≤ t\}. A chase sequence for D under Σ is level-wise if, for i > 0, it produces all the atoms of s-level i before generating an atom of s-level i + 1.

**Lemma A.1.** Consider a database D, a set Σ ⊆ L, and a UCQ q. There exists a computable function g : N → N such that

q(chase(D, Σ)) = q\left(\text{chase}_s^{g(|Σ|+|q|)}(D, Σ)\right),

where s is a level-wise chase sequence for D under Σ. Furthermore, the instance chase_s^{g(|Σ|+|q|)}(D, Σ) can be computed in time |D| · f(|Q|) for some computable function f : N → N.

**Proof.** The existence of the computable function g, which is actually an exponential function, has been shown in [15]. It remains to show that chase_s^{g(|Σ|+|q|)}(D, Σ) can be computed in time |D| · f(|Q|) for some computable function f : N → N; recall that s is a level-wise chase sequence for D under Σ. We first provide an upper bound on the size of chase_s^{g(|Σ|+|q|)}(D, Σ). Let H_Σ be the maximum number of atoms in the head of a TGD of Σ.

**Lemma A.2.** It holds that

|chase_s^{g(|Σ|+|q|)}(D, Σ)| ≤ |D| · (|Σ| · H_Σ + 1)^{g(|Σ|+|q|)}.

**Proof.** We proceed by induction on the s-level i ≥ 0.

**Base case.** It is clear that chase_0^0(D, Σ) = D, and the claim follows.

**Inductive step.** Observe that, due to linearity, each TGD of Σ can be triggered by an atom of chase_{i−1}^0(D, Σ) at most once, and generate H_Σ new atoms. Hence,

|chase_i^0(D, Σ)| ≤ |chase_{i−1}^0(D, Σ)| · |Σ| · chase_i^0(D, Σ) · H_Σ.

By induction hypothesis,

|chase_{i−1}^0(D, Σ)| ≤ |D| · (|Σ| · H_Σ + 1)^{i−1}.

Therefore,

|chase_i^0(D, Σ)| ≤ |D| · (|Σ| · H_Σ + 1)^{i−1} · |Σ| · |D| · (|Σ| · H_Σ + 1)^{i−1} · H_Σ

= |D| · (|Σ| · H_Σ + 1)^{i−1} · (|Σ| · H_Σ + 1)

= |D| · (|Σ| · H_Σ + 1)^i,

and the claim follows.

---

We can now show that chase_s^{g(|Σ|+|q|)}(D, Σ) can be computed in time |D| · f(|Q|) for some computable function f : N → N. It is easy to see that, for each i > 0, chase_i^0(D, Σ) can be computed from chase_{i−1}^0(D, Σ) in time \( (|D| · (|Σ| · H_Σ + 1)^i · (|Σ| · H_Σ + 1)) \).

Indeed, for each linear TGD \( σ ∈ Σ \), we need to scan the instance chase_{i−1}^0(D, Σ), and check whether the atom body(σ) matches with an atom of chase_{i−1}^0(D, Σ), and, if this is the case, add to the instance under construction H_Σ new atoms. By Lemma A.2, for each i ≥ 0,

\[ \text{chase}_i^0(D, Σ) ≤ \text{chase}_s^{g(|Σ|+|q|)}(D, Σ) \]

\[ ≤ |D| · (|Σ| · H_Σ + 1)^{g(|Σ|+|q|)} · |Σ| · H_Σ · (\text{ar}(T) + 1)^2. \]

This completes the proof of Lemma A.1.

The second technical lemma we need to complete the proof of item (3) of Proposition 3.3 follows:

**Lemma A.3.** Let D be an S-database, and \( Q = (S, Σ, q) \in (G, UCQ) \). There exists a database \( D^* \), and a set \( Σ^* ∈ L \) such that \( Q(D) = q(\text{chase}(D^*, Σ^*)) \). Furthermore, \( D^* \) can be computed in time \(|D|^{O(1)} · f(|Q|)\), and \( Σ^* \) in time \( g(|Q|)\) for some computable functions \( f, g : N → N \).

Before giving the proof of the above result, let us explain how Lemma A.1 and Lemma A.3 allow us to obtain item (3) of Proposition 3.3. Given an OMQ \( Q = (S, Σ, q(\bar{x})) ∈ (G, UCQ) \), for \( k ≥ 1 \), an S-database D, and \( \bar{ε} ∈ \text{dom}(D)[|\bar{x}|] \), by Lemmas A.1 and A.3, we simply need to construct a finite instance I in time \(|D|^{O(1)} · f(|Q|)\) for some computable function f : N → N, and then check if \( \bar{ε} ∈ q(I) \).

Since \( q ∈ UCQ_k \), by Proposition 2.1, the overall procedure takes time \(|D|^{O(1)} · g(|Q|)\) for some computable function g : N → N.

A.1 Proof of Lemma A.3

The rest of this section is devoted to the proof of Lemma A.3. Consider an S-database D, and an OMQ \( Q = (S, Σ, q) ∈ (G, UCQ) \). We proceed to construct a database \( D^* \), and a set \( Σ^* ∈ L \), by adapting a construction from [23], such that \( Q(D) = q(\text{chase}(D^*, Σ^*)) \). Let us first explain the high-level idea underlying \( D^* \) and \( Σ^* \).

The High-level Idea. A key notion when reasoning with guarded TGDs is the type of an atom \( η \) (w.r.t. D and Σ), denoted type_T D (η), defined as the set \( \{β ∈ \text{chase}(D, Σ) | \text{dom}(β) ⊆ \text{dom}(η)\} \). Roughly speaking, the key property of the type is that the atoms that can be derived during the chase from the atom \( η \) used as a guard is determined by its type; see [15] for further details. The main idea underlying the construction of \( D^* \) is to encode an atom \( σ ∈ D \) and its type as a single atom of the form \( [\tau](), \) where \( r \) is essentially a representation of the “shape” of \( σ \) and its type. For example, given an atom \( R(\bar{a}, b) ∈ D \), and assuming that its type is \( \{R(\bar{a}, b), S(\bar{b}, a), T(\bar{a}), T(\bar{b})\} \), we can encode \( R(\bar{a}, b) \) and its type as the atom \( \{R(1, 2), S(2, 1), T(1), T(2)\}(\bar{a}, \bar{b}) \). When it comes to \( Σ^* \), the intention is, for a TGD \( σ ∈ Σ \), to encode the shape of the type \( r \)
of the guard of $\sigma$ in a predicate $[\tau]$, and then replace $\sigma$ with a linear TGD that uses in its body an atom of the form $[\tau](\cdot)$. However, we need an effective way to compute the type of an atom $\alpha$ by completing its known part, which is inherited from the type of the guard atom that generates $\alpha$, with atoms that mention the new nulls invented in $\alpha$. This relies on the main property of the type mentioned above. In particular, for each $\alpha \in \text{chase}(D, \Sigma)$ obtained due to the application of the TGD $\sigma = \psi(\hat{x}, \hat{y}) \Rightarrow \exists \bar{z} \psi(\hat{x}, \hat{z}) \in \Sigma$ with the witness $(\hat{i}, \hat{u})$, we can construct $\text{type}_{D, \Sigma}(\alpha)$ from $\psi(\hat{i}, \hat{v})$, where $\hat{v}$ is a tuple of new null values, together with the restriction of $\text{type}_{D, \Sigma}(\alpha)$ to the terms $\bar{t}$. This is precisely how we are going to generate new types from existing ones. Before giving the formal construction, we need to introduce some auxiliary terminology.

**Auxiliary Terminology.** Let $\text{sch}(\Sigma)$ be the set of predicates occurring in $\Sigma$. For an atom $\alpha$, let $\text{base}(\alpha, \Sigma)$ be the set of all atoms that can be formed using terms from $\text{dom}(\alpha)$ and predicates from $\text{sch}(\Sigma)$. A $\Sigma$-type $\tau$ is a pair $(\alpha, \Gamma)$, where $\alpha = R(t_1, \ldots, t_n) \in \text{sch}(\Sigma)$, with $t_i = 1$ and $t_i \in \{t_1, \ldots, t_{i-1}, t_{i+1}, t_n\}$ for each $i \in \{2, \ldots, n\}$, and $\Gamma \subseteq \text{base}(R(t_1, \ldots, t_n), \Sigma) \setminus \{\alpha\}$. We write $\text{guard}(\tau)$ for the atom $\alpha$, and $\text{atoms}(\tau)$ for $\{\{\alpha \cup \Gamma\} \cup T\}$. The arity of $\tau$, denoted $\text{ar}(\tau)$, is the maximum integer occurring in $\text{guard}(\tau)$. Intuitively, $\tau$ encodes the shape of a guard atom $\alpha$ and a set of side atoms that are “covered” by $\alpha$. Consider a tuple $\hat{u} = (u_1, \ldots, u_n)$ that is isomorphic to $\bar{t} = (t_1, \ldots, t_n)$, written $\hat{u} \equiv \bar{t}$, which means that $u_i = u_j$ iff $t_i = t_j$. The instantiation of $\tau$ with $\hat{u}$, denoted $\tau(\hat{u})$, is the set of atoms obtained from atoms($\tau$) after replacing each $t_i$ with $u_i$.

The formal construction of $D^*$ is defined as follows:

$$D^* = \{[\tau](\cdot) \mid \tau = (R(\bar{t}), \cdot) \text{ with } \bar{t} \equiv \bar{t} \wedge \text{complete}(I, \Sigma) \wedge (R(\bar{t}) \rightarrow \text{chase}(I, \Sigma))\}.$$  

The new set $\Sigma^*$ of linear TGDs is the union of $\Sigma^*_t$ and $\Sigma^*_e$, where

1. $\Sigma^*_t$ is the type generator, i.e., is responsible for generating new $\Sigma$-types from existing ones.
2. $\Sigma^*_e$ is the expected, it expands a derived $\Sigma$-type $\tau$, i.e., it explicitly constructs the atoms over $\text{sch}(\Sigma)$ encoded by $\tau$.

The type generator is defined as follows. For each $\sigma \in \Sigma$

$$\psi(\hat{x}, \hat{y}) \rightarrow \exists z_1 \cdots \exists z_m R_1(u_1), \ldots, R_n(u_n)$$

with $\text{guard}(\sigma) = G(\bar{u})$, and $(\hat{x} \cup \{z_i\}_{i \in [m]})$ the variables occurring in $\text{head}(\sigma)$, and for every $\Sigma$-type $\tau$ such that there is a homomorphism $h$ from $\psi(\hat{x}, \hat{y})$ to atoms($\tau$) and $h(G(\bar{u})) = \text{guard}(\tau)$, we add to $\Sigma^*_t$

$$[\tau](\hat{u}) \rightarrow \exists z_1 \cdots \exists z_m [\tau_i](\hat{u}_i), \ldots, [\tau_n](\hat{u}_n),$$

where, having the function $f$ with

$$f(\tilde{t}) = \begin{cases} h(t) & \text{if } t \in \hat{x} \\ \text{ar}(\Sigma) + i & \text{if } t = z_i, \end{cases}$$

as well as the function $\rho$ that renames the integers in atoms in order to appear in increasing order starting from 1 (e.g., $\rho(R(2, 2, 4, 1)) = R(1, 1, 2, 3))$, for each $i \in [n]$, with $a_i = R_i(\rho(\bar{u})))$, the $\Sigma$-type $\tau_i = (\rho(a_i), \{\beta \in \text{complete}(I, \Sigma) \mid \text{dom}(\beta) \subseteq \text{dom}(\rho(a_i))\} \setminus \{\rho(a_i)\})$ with

$$I = \rho([\{a_i\}_{i \in \{1, \ldots, n\}} \cup \prod_{i \in [k]} \{x_{i,x} \mid x \in x\}](\tau)).$$

The expander it actually constructs, for each $\Sigma$-type $\tau$, the guard atom of $\tau$. More precisely, for each $\Sigma$-type $\tau$, we add to $\Sigma^*_e$

$$[\tau](x_1, \ldots, x_k) \rightarrow R(x_1, \ldots, x_k),$$

where $R$ is the $k$-ary predicate of guard($\tau$).

This completes the construction of $\Sigma^*$. It is clear that $\Sigma^* \in \Sigma$. It also does not restrict to show, by exploiting the properties of the type, that $\text{Q}(D) = g(\text{chase}(D, \Sigma^*), \text{as needed})$. It remains to show that $D^*$ and $\Sigma^*$ can be constructed in the claimed times.

**APPT Construction.** By construction, $\Sigma^*$ depends only on $\Sigma$, and thus, it is clear that it can be computed in time $g(||\Sigma||)$ for some computable function $g : \mathbb{N} \rightarrow \mathbb{N}$. On the other hand, $D^*$ depends both on $D$ and $\Sigma$. It remains to show that that $D^*$ can be computed in time $||D||^{O(1)} : f(||\Sigma||)$ for some computable function $f : \mathbb{N} \rightarrow \mathbb{N}$.

To this end, we first show the following auxiliary result:

**Lemma A.4.** For a database $D'$ and set $\Sigma' \subseteq \text{GnFULL}$(the finite) instance chase($D', \Sigma'$) can be constructed in time $||D'||^{O(1)} : g(||\Sigma'||)$ for some computable function $g : \mathbb{N} \rightarrow \mathbb{N}$.

**Proof.** We provide similar analysis as in the proof of Lemma A.1. Let us first establish an upper bound on the size of chase($D', \Sigma'$).

We assume that $\Sigma'$ is over the schema $T$. We can show that:

$$\text{size}(D', \Sigma') \leq |D'| \cdot |T| \cdot \text{ar(T)}^{\text{ar}(T)}.$$

Observe that, due to guardedness, two constants $c, d \in \text{dom}(D')$ can occur together in an atom of chase($D', \Sigma'$) only if they already occur together in an atom of $D'$. Thus, for each n-tuple $\bar{c}$ occurring in $D'$ (i.e., there is an atom $R(\bar{c})$ in $D'$), the chase can produce at most $n^{\text{ar}(T)}$ new tuples of constants from $\bar{c}$, while each new such tuple can be stored in a predicate of $T$. Since $n \leq \text{ar}(T)$, we get that chase($D', \Sigma'$) can have at most $|D'| : |T| : \text{ar(T)}^{\text{ar}(T)}$ atoms.

We can now show that indeed chase($D', \Sigma'$) can be constructed in time $||D||^{O(1)} : g(||\Sigma||)$. Let $B_{\Sigma'}$ (resp., $B_{\Sigma}$) be the maximum number of atoms in the body (resp., head) of a TGD of $\Sigma'$.

Observe that, for $i = 0$, chase$_i(D', \Sigma')$, where $s_i$ is a level-wise chase sequence for $D'$ under $\Sigma'$, can be computed from chase$_{i-1}(D', \Sigma')$ in time

$$|\Sigma'| \cdot \left(\text{chase}_{i-1}(D', \Sigma') \cdot (\text{ar(T)} + 1) + B_{\Sigma'} \cdot (\text{ar(T)} + 1)\right) \cdot B_{\Sigma} \cdot (\text{ar(T)} + 1).$$

Indeed, for each guarded TGD $\sigma \in \Sigma'$, we need to scan the instance chase$_{i-1}(D', \Sigma')$, and check whether the atom guard($\sigma$) matches with an atom $\alpha \in \text{chase}_{i-1}(D', \Sigma')$. Note that the matching of guard($\sigma$) with $\alpha$ uniquely determines the atoms $\alpha_1, \ldots, \alpha_m$ with which the atoms of body($\sigma$) apart from guard($\sigma$) should match. Then, for each such atom $\alpha$, we need to scan chase$_{i-1}(D', \Sigma')$, and check that indeed it is present. If all the atoms $\alpha_1, \ldots, \alpha_m$ occur in chase$_{i-1}(D', \Sigma')$, then we add $B_{\Sigma'}$ new atoms.

As shown above, for each $i \geq 0$, chase$_i(D', \Sigma') \leq |\text{chase}(D', \Sigma')| \leq |D'| : |T| : \text{ar(T)}^{\text{ar}(T)}$.
Since chase(D', Σ') can have at most \( n = |D'| \cdot |\Sigma| \cdot \ar(T)^{\ar(T)} \) s-levels, it can be computed in time
\[
\sum n \cdot |\Sigma'| \cdot (n \cdot (\ar(T) + 1) + B_{2^\gamma} \cdot (\ar(T) + 1) + n \cdot B_{2^\gamma} \cdot (\ar(T) + 1)).
\]
Summing up, chase(D', Σ') can be computed in time
\[
|D'|^2 \cdot g(||\Sigma'||)
\]
for some computable function \( g: \mathbb{N} \to \mathbb{N} \), and the claim follows.

In view of Lemma A.4, to show that \( D' \) can be computed in time \( |D|^{O(1)} \cdot f(||Q||) \), it suffices to show that we can construct in time \( g(||\Sigma||) \) a set \( \hat{\Sigma} \subseteq \Sigma \cap \text{FULL} \) such that \( D' = \text{chase}(D, \hat{\Sigma}) \), where \( T \) consists of the values of the form \( \tau r \), where \( r \) is a \( \Sigma \)-type, occurring in chase(D, \hat{\Sigma}). The set \( \hat{\Sigma} \) will simply:

1. complete the database \( D \) with all the atoms of chase(D, \Sigma) that mention only constants from \( \text{dom}(D) \), i.e., add to \( D \) the set of atoms chase(D, \Sigma) \cap \text{base}(D, \Sigma)
2. generate all the atoms of \( D' \).

For achieving task (1) above, we exploit a result from [24], which states the following: given a set \( \Sigma' \subseteq \Sigma \), we can construct a set \( \xi(\Sigma') \in \Sigma \cap \text{FULL} \) over \( \text{sch}(\Sigma') \) such that, for every database \( D' \), chase\( (D', \xi(\Sigma')) = \text{chase}(D', \Sigma') \cap \text{base}(D', \Sigma') \). Thus, task (1) can be done via the set \( \xi(\Sigma) \in \Sigma \cap \text{sch}(\Sigma) \).

Task (2) can be achieved via the set of TGDs \( \Sigma_{\text{types}} \subseteq \Sigma \cap \text{FULL} \) defined as follows. For each \( \Sigma \)-type \( r \), we have a TGD
\[
\tau(x_1, \ldots, x_k) \rightarrow [\tau](x_{f_1}(i), \ldots, x_{f_l}(i)),
\]
where \( k = \ar(r) \), \( f \) is the arity of the predicate \( \text{guard}(\tau) \), and, assuming that \( (i_1, \ldots, i_l) \) is the tuple of \( \text{guard}(\tau) \), \( f(i) = i_j \), for each \( j \in [l] \). By abuse of notation, we use \( r(x_1, \ldots, x_k) \) to denote the conjunction of atoms in the instantiation of \( r \) with \( (x_1, \ldots, x_k) \).

The set \( \hat{\Sigma} \) is defined as \( \xi(\Sigma) \cup \Sigma_{\text{types}} \), which can be clearly constructed in time \( g(||\Sigma||) \) since it depends only on \( \Sigma \).

**B. PROOF OF THEOREM 5.1**

Given a CQ \( q \), we write \( \text{var}(q) \) for the set of variables in \( q \). We may also treat, as usual, a CQ \( q \) as the set of atoms in \( q \). The proofs in this section rely on notions and results introduced in the proof of Proposition 5.2, which we advise the reader to read first.

The 2EXPTime-hardness is inherited from [7], where the same problem for OMQs based on DLs has been studied. It remains to establish the 2EXPTime upper bound. Let \( Q = (S, \Sigma, q) \). Proposition 5.2 provides a procedure for deciding (uniform) \( \text{UCQ}_k \)-equivalence:

1. Compute the \( \text{UCQ}_k \)-approximation \( Q_k^k = (S, \Sigma, q_k^k) \) of \( Q \);
2. Accept if \( Q \equiv Q_k^k \); otherwise, reject.

This yields decidability. It also shows that if \( Q \) is \( \text{UCQ}_k \)-equivalent, then an \( \text{OMQ} Q' \) from \( (S, \Sigma, Q_k) \) such that \( Q \equiv Q' \) can be constructed in double exponential time. Actually, to find \( Q' \) we can simply construct the \( \text{OMQ} Q_k^k \).

**lemma B.1.** \( Q_k^k \) can be constructed in double exponential time.

Proof. For each disjunct \( p \) of \( q \), we first need to compute its specializations \( (p', V) \). It is easy to see that there are single exponentially many specialization that can be found in single exponential time. For each specialization \( s = (p', V) \), we then have to find all \( \Sigma \)-groundings \( g_s(\bar{x}) \) of \( s \), and keep the ones of treewidth at most \( k \). Note that we can form double exponentially many guarded CQs, each of exponential size. Thus, there are double exponentially many candidates for being a \( \Sigma \)-grounding of \( s \), each for exponential size, that can be found in double exponential time. Now, in the check whether such a candidate \( q \) is indeed a \( \Sigma \)-grounding of \( s \), the only non-trivial part is that we have to check for each maximally \( |V| \)-connected component \( p_i \) of \( p' \) whether \( p_i \rightarrow \text{chase}(q_i, \Sigma) \), where \( q_i \) is a guarded subquery of \( g \) of exponential size, via a homomorphism that is the identity on \( \varphi(p_i) \cap V = \{y_1, \ldots, y_m\} \). This boils down to the problem whether \( (y_1, \ldots, y_m) \in p_f'(\text{chase}(q_i, \Sigma)) \), where \( p_f'(y_1, \ldots, y_m) \) consists of the atoms of \( p_i \). By item (2) of Proposition 3.2, this can be checked in 2EXPTime. Actually, this is not immediate from Proposition 3.2 since \( g_i \) might be of exponential size. We additionally need to say that the double exponential bound obtained from Proposition 3.2 is only polynomial in the size of the database; implicit in [14], Finally, checking whether a \( \Sigma \)-grounding \( g_k \) has treewidth at most \( k \) is feasible in double exponential time since \( g_k \) is, in general, of exponential size.

Containment between OMQs from \( (G, \text{UCQ}) \) can be decided in 2EXPTime [6]. Since, however, \( q_k^k \) may consists of double exponentially many CQs, it is not clear how to implement step (2) above in 2EXPTime and how to obtain the 2EXPTime upper bound in Theorem 5.1 by a direct implementation of the above procedure. We solve this problem in two steps. First, we replace \( Q_k^k \) with an \( \text{OMQ} Q_k' = (S, \Sigma', q_k') \) that is equivalent to \( Q_k^k \), and such that \( q_k' \) contains only single exponentially many CQs, each of polynomial size. And second, we modify the containment check from [6] in a mild way.

**B.1 Replacing \( Q_k^a \) with \( Q_k' \)**

The \( \text{OMQ} Q_k' = (S, \Sigma', q_k') \) is defined as follows. We introduce a fresh unary relation symbol \( A \) (that is not in \( S \)) and obtain \( \Sigma' \subseteq \Sigma \) by extending every TGD head with the atom \( A(x) \) whenever \( x \) is an existentially quantified variable in that TGD head. Then \( q_k' \) is the UCQ that includes \( p_k \) for all specializations \( s = (p, V) \) of a CQ \( p \) in \( q \) such that there exists a \( \Sigma \)-grounding of \( s \) that is of treewidth at most \( k \), extended by adding \( A(x) \) for every variable \( x \in \varphi(p_k) \cap V \).

It is clear that \( q_k' \) has only single exponentially many disjunctions, each of polynomial size. Reusing the arguments from the proof of Lemma B.1, it is also clear that \( Q_k' \) can be constructed in double exponential time. We further make the following important observation, which rests on the assumption that \( k \geq \ar(T) - 1 \).

**Lemma B.2.** Let \( s = (p, V) \) be a specialization of a CQ in \( q \). If there is a \( \Sigma \)-grounding of \( s \) that has treewidth at most \( k \), then all \( \Sigma \)-groundings of \( s \) have treewidth at most \( k \).

Proof. Let \( p_1, \ldots, p_n \) be the maximally \( |V| \)-connected components of \( p \), let \( g_k(\bar{x}) = \exists \bar{y} (g_k \land g_1 \land \cdots \land g_n) \) be a \( \Sigma \)-grounding of \( s \), and let \( \delta = (T_\delta, \chi) \), where \( T_\delta = (V_\delta, E_\delta) \), be a tree decomposition of \( C_{\text{CH}}^k \) of width \( k \), \( \bar{y} \) the existentially quantified variables of \( g_k(\bar{x}) \). For \( i \in [n] \), there must be a \( u_i \in V_\delta \) with \( \varphi(p_i) \cap V_\delta \subseteq \chi(u_i) \). Now, let \( g_k^i(\bar{x}) = \exists \bar{y} (g_k \land g_1^i \land \cdots \land g_n^i) \) be another \( \Sigma \)-grounding of \( s \). Let \( \delta' = (T_\delta', \chi') \) be obtained from \( \delta \) by dropping all variables that occur in \( g_k \), but not in \( g_k^i \) from every bag and adding, for \( i \in [n] \), a fresh successor \( u_i \) to \( u_i \) and setting \( \chi'(u_i) = \varphi(g_k^i) \). It can be
verified that $\delta'$ is a tree decomposition of $G'_{\delta}$, $g'$ the existentially quantified variables of $G'_{\delta}$. In particular, Condition (2) of tree decompositions is satisfied for every edge induced by an atom in some $g'_i$ due to the presence of $u_i$. Since $g'_i$ is guarded, the number of variables in $\text{var}(g'_i)$ is bounded by $\alpha(T)$. Since $k \geq \alpha(T) - 1$, the width of $\delta'$ is bounded by $k$, and the claim follows.

We now proceed to show that $Q'_k$ and $Q'_k$ are indeed equivalent.

**Lemma B.3.** $Q'_k \equiv Q'_k$.

**Proof.** First assume that $\hat{c} \in Q'_k(D)$ with $D$ an S-database. Then, there is a homomorphism $h$ from a CQ $p(x)$ in $Q'_k$ to chase($D, \Sigma'$) with $h(\hat{x}) = \hat{c}$. Assume that $p$ was constructed for the specialization $s = (p, V)$, that is, $p$ is the extension of $p_1$ with $A(x)$ for all $x \in \text{var}(p_1) \setminus V$. By construction of $\Sigma'$, $h$ thus maps all variables in $\text{var}(p) \setminus V$ to constants that the chase has introduced to satisfy existential quantifiers in TGD heads. Let $p_1, \ldots, p_n$ be the maximally $[V]$-connected components of $p_1[V]$. It follows from the facts that $p_1$ is $[V]$-connected, and that $h$ maps all variables in $\text{var}(p) \setminus V$ to existential constants that we find a fact $s_i \in D$ such that the restriction of $h$ to the variables in $p_1$ is a homomorphism to chase(type$_{p_i, V}((a_i), \Sigma'))$, for $i \in [n]$. Let the $\Sigma'$-grounding $g_i$ of $s_i$ be obtained as $g_i$ the guarded full CQ that is obtained by viewing type$_{p_i, V}((a_i), \Sigma')$ as a full CQ. Clearly, the restriction of $h$ to the variables in $V$ can be extended to a homomorphism $h'$ from $g_i$ to chase($D, \Sigma'$) by mapping the variables from the CQs $g_i$ that are not in $V$ to constants that $s_i$ in the expected way. Since $g_i$ does not contain the relation $A$, $h'$ maps $g_i$ to chase($D, \Sigma'$). It remains to argue that $g_i$ is a CQ in $Q'_k$. Since $p$ is in $G'_k$, there is a $\Sigma'$-grounding of $s$ that is of treewidth at most $k$. By Lemma B.2, this implies that $g_i$ is of treewidth at most $k$ and thus $g_i$ is in $Q'_k$.

Conversely, assume that $\hat{c} \in Q'_k(D)$ with $D$ an S-database. Then, there is a homomorphism $h$ from a CQ $g_i(\hat{x})$ in $Q'_k$ to chase($D, \Sigma$) with $h(\hat{x}) = \hat{c}$. Let $g_i$ be a $\Sigma'$-grounding of the specialization $s = (p_1, V)$ of a CQ $p$ in $q$. Then $g_i$ is of treewidth at most $k$. Consequently, $s$ gives rise to a corresponding CQ $p'$ in $q'$, that is, $p'$ is the extension of $p_1$ with $A(x)$ for all $x \in \text{var}(p_1) \setminus V$. Clearly, the restriction of $h_i$ to the variables in $V$ is a homomorphism from $p'_{\mid V}$ to chase($D, \Sigma$). Let $p_1, \ldots, p_n$ be the maximally $[V]$-connected components of $p_1[V]$. By definition of $\Sigma'$-groundings, $p_1 \rightarrow chase(g_i, \Sigma)$ via a homomorphism that is the identity on $\text{var}(p_1) \setminus V$, for $i \in [n]$. We can combine all these homomorphisms with $h_i_{\mid V}$ to obtain a homomorphism $h'$ from $p_1$ to chase($D, \Sigma$) that maps all variables outside of $V$ to constants that have been introduced by the chase to satisfy existential quantifiers in TGD heads. By construction of $p'$ and $\Sigma'$, $h'$ is also a homomorphism from $p'$ to chase($D, \Sigma'$).

### B.2 Checking Equivalence

We now proceed to analyze the complexity of checking whether $Q \equiv Q'_k$. By item (1) of Lemma C.7 and Lemma B.3, $Q'_k \subseteq Q$. The non-trivial task is to check whether $Q \subseteq Q'_k$. We know from [6] that checking containment among OMQs from $\mathcal{G}(\mathcal{U}, \mathcal{C})$ is in 2ExpTime. In fact, [6] first provides a polynomial time reduction from containment among OMQs from $\mathcal{G}(\mathcal{U}, \mathcal{C})$, denoted $\mathsf{Cont}(\mathcal{G}, \mathcal{U}, \mathcal{C})$, to containment among OMQs from $\mathcal{G}(\mathcal{C})$, denoted $\mathsf{Cont}(\mathcal{G}, \mathcal{C})$, and then devises an automata-based procedure for $\mathsf{Cont}(\mathcal{G}, \mathcal{C})$ that runs in double exponential time in the combined size of the OMQs. However, $Q'_k$ might be of exponential size, which implies that if we first apply the reduction from $\mathsf{Cont}(\mathcal{G}, \mathcal{U}, \mathcal{C})$ to $\mathsf{Cont}(\mathcal{G}, \mathcal{C})$, we get an OMQ that has a CQ of exponential size, and then, by using the decision procedure for $\mathsf{Cont}(\mathcal{G}, \mathcal{C})$ from [6] as a black box, we only get a 3ExpTime upper bound. Nevertheless, it turns out that we can reuse the automata constructions from [6] for $\mathsf{Cont}(\mathcal{G}, \mathcal{C})$ in order to devise a procedure that directly operates on OMQs from $\mathcal{G}(\mathcal{U}, \mathcal{C})$ and decides containment in time double exponential in the size of the ontology, the schema, and the maximum size of the CQs in the UMQs in the OMQs, but only single exponential in the number of disjuncts in those UMQs. Having such a procedure in place, it is then clear that checking whether $Q \subseteq Q'_k$ is in 2ExpTime since, due to Lemma B.1, and since although the UMQ in $Q'_k$ has exponentially many disjuncts, each one is of polynomial size. For the sake of completeness, we recall the key ingredients underlying the automata-based procedure from [6], and then explain how we get the automata-based procedure that directly operates on OMQs from $\mathcal{G}(\mathcal{U}, \mathcal{C})$ that leads to the desired upper bound.

**The Automata-Based Procedure for $\mathsf{Cont}(\mathcal{G}, \mathcal{U}, \mathcal{C})$**

Let us first recall that we can focus, w.l.o.g., on Boolean OMQs since there is an easy reduction, originally proposed in [10], of containment for non-Boolean OMQs to containment for Boolean OMQs. In what follows, we write $\mathcal{U}, \mathcal{C}$ and $\mathcal{C}$ meaning the classes of Boolean UCQs and Boolean CQs, respectively.

We know from [6] that, given two OMQs $Q_1$ and $Q_2$ from $\mathcal{G}(\mathcal{C})$, if $Q_1 \nsubseteq Q_2$, then this is witnessed by a “nearly acyclic” database. To formalize this we need the notion of guarded tree decomposition. A tree decomposition $\delta = (T_8, \chi)$, where $T_8 = (V_8, E_8)$, of $G^D$ for some database $D$ is called $[V]$-guarded, where $V \subseteq V_8$, if for every node $v \in V_8$ in the subset of $D$ induced by $\chi(v)$. An S-database is called a C-tree, where $C \subseteq D$, if there exists a tree decomposition $\delta$ of $G^D$ such that: (i) $D_8(\text{root}(\delta)) = C$, and (ii) $\delta$ is $([\text{root}(\delta)])$-guarded. Roughly, if for a database $D$ is a C-tree, can be viewed as the cyclic part of $D$, while the rest of $D$ is acyclic. In [6], it has been shown that for containment purposes we can focus on databases that are C-trees with the size of $C$ bounded and depending only on the left-hand side OMQ. Note, however, that the result of [6] talks only about OMQs from $\mathcal{G}(\mathcal{C})$. A careful inspection of the proof given in [6] reveals that the following statement for OMQs from $\mathcal{G}(\mathcal{U}, \mathcal{C})$ holds:

**Lemma B.4.** Let $Q_1 = (S, \Sigma_1, q_1)$ and $Q_2 = (S, \Sigma_2, q_2)$ be OMQs from $\mathcal{G}(\mathcal{U}, \mathcal{C})$. The following are equivalent:

1. $Q_1 \nsubseteq Q_2$.
2. There exists a C-tree $S$-database $D$ with $||\text{dom}(C)|| \leq \alpha(S \cup \text{sch}(S_1)) \cdot \max_{p \in q_1} ||p||$ such that $Q_1(D) \nsubseteq Q_2(D)$.

The goal is to devise a two-way alternating parity tree automaton (2ATA) that checks for the second item of Lemma B.4. To this end, we need a convenient encoding of a C-tree database as a tree over a finite alphabet that can be accepted by a 2ATA. As shown in [6], this can be done by exploiting standard encodings of bounded treewidth.
databases. In a nutshell, a C-tree S-database D can be encoded as a
finite Γₖₙₜ –labeled tree, with ||dom(C)|| ≤ ℓ and Γₖₙₜ being an
alphabet of double exponential size in ar(S), and exponential in
|S| and ℓ; the actual encoding is not important for our discussion.
Of course, it is possible that a Γₖₙₜ –labeled tree does not encode a
C-tree database. Thus, some additional consistency properties are
needed, which are also not important for our discussion. A Γₖₙₜ –labeled
tree is consistent if it satisfies those syntactic properties, and
it can be shown that a consistent Γₖₙₜ –labeled tree L can be decoded
into an S-database, denoted [L], that is a C-tree with ||dom(C)|| ≤ ℓ.
Therefore, by Lemma B.4, we get the following:

Lemma B.5. Let Q₁ = (S, Σ₁, q₁) and Q₂ = (S, Σ₂, q₂) be OMQs
from (G, UCQ). The following are equivalent:

(1) Q₁ ⫋ Q₂.
(2) There exists a consistent Γₖₙₜ –labeled tree L, where ℓ = ar(S ∪
sch(Σ₁)) · maxₚ∈q₁(||p||), such that Q₁((L)) ⫋ Q₂((L)).

It should be clear now that the goal is to devise a 2ATA A over
finite trees such that its language, denoted L(A), is the set of
Γₖₙₜ –labeled tree L, where ℓ ≤ ar(S ∪ sch(Σ₁)) · maxₚ∈q₁(||p||), such
Q₁((L)) ⫋ Q₂((L)). Having such an automaton in place, checking
whether Q₁ ⫋ Q₂ boils down to checking whether L(A) is empty,
which can be done in exponential time in the number of states of
A, and in polynomial time in the size of the input alphabet. Thus,
to obtain the desired 2ExpTime upper bound, we need to construct
A in polynomial time, while the number of its states should
be at most exponential. To construct the desired 2ATA we are going
to exploit the automata constructions from [6].

First, we need a way to check consistency of labeled trees, which,
as shown in [6], can be done via an automaton

Lemma B.6. Consider a schema S, and an integer ℓ > 0. There is a
2ATA Cₖₙₜ that accepts a Γₖₙₜ –labeled tree L iff L is consistent. The
number of states of Cₖₙₜ is logarithmic in the size of Γₖₙₜ. Furthermore,
Cₖₙₜ can be constructed in polynomial time in the size of Γₖₙₜ.

We also know from [6] that, given an OMQ Q from (G, CQ), we
can devise an automaton that accepts labeled trees which corre-
spond to databases that make Q true.

Lemma B.7. Let Q = (S, Σ, q) ∈ (G, CQ). There is a 2ATA A_Q,ℓ,
where ℓ > 0, that accepts a consistent Γₖₙₜ –labeled tree L iff [L] |= Q.
A_Q,ℓ has exponentially many states in ||Q|| and ℓ, and it can be
constructed in double exponential time in ||Q|| and ℓ.

We are now ready to devise the desired 2ATA. To this end, we
are going to exploit Lemmas B.5 to B.7, and some well-known
facts about 2ATAs. In particular, languages accepted by 2ATAs are
closed under union and complementation. Given two 2ATAs A₁ and
A₂, we write A₁ ∪ A₂ or a 2ATA, which can be constructed in
polynomial time, that accepts the language L(A₁) ∪ L(A₂). More-
over, we write A for the 2ATA, which can also be constructed in
polynomial time, that accepts the complement of L(A). Import-
antly, the number of states of A₁ ∪ A₂ is the sum of the num-
bers of states of A₁ and A₂ plus one. Given Q₁ = (S, Σ₁, q₁) and
Q₂ = (S, Σ₂, q₂) from (G, UCQ), we write Q₁ ※₂ for the largest OMQ
among ∪ₚ∈q₁(S, Σ₁, p) ∪ ∪ₚ∈q₂(S, Σ₂, p).

Lemma B.8. Let Q₁ = (S, Σ₁, q₁) and Q₂ = (S, Σ₂, q₂) be OMQs
from (G, UCQ). There is a 2ATA A such that
Q₁ ⊆ Q₂ ⇐⇒ L(A) = ∅.

A has exponentially many states in ||Q₁ ※₂|| and ℓ, and polynomially
many states in ||q₁|| + ||q₂||. It can be constructed in double exponential
time in ||Q₁ ※₂|| and ℓ, and in polynomial time in ||q₁|| + ||q₂||.

Proof. Let ℓ = ar(S ∪ sch(Σ₁)) · maxₚ∈q₁(||p||). The 2ATA A is
\[(Cₖₙₜ \cap \bigcupₚ∈q₁ A(S, Σ₁, p, ℓ)) \cap \bigcupₚ∈q₂ A(S, Σ₂, p, ℓ)\].

Using Lemmas B.5, B.6, and B.7, the fact that union and com-
plementation of 2ATAs are feasible in polynomial time, and the stated
bound on the number of states in automata for union, it is easy to
verify that indeed A is the desired 2ATA.

It is now easy to see, having Lemma B.8 in place, that indeed
checking whether Q ⊆ Q’ is feasible in double exponential time.
This actually follows from the fact that the UCQ of Q’ has exponen-
tially many disjuncts, each of polynomial size, and the fact that
emptiness for 2ATA can be decided in exponential time in the
number of states of the automaton.

C PROOF OF PROPOSITION 5.2

UCQₖₙₜ–approximations for OMQs from (G, UCQ) rely on the notion
of specialization of a CQ. We first introduce CQ specializations, and
give a lemma that illustrates their usefulness. We then define the
notion of grounding of a CQ specialization, and establish a technical
lemma that will allow us to define UCQₖₙₜ–approximations. We then
establish our main technical result about UCQₖₙₜ–approximations,
which in turn allows us to complete the proof of Proposition 5.2. In
what follows, given a CQ q, we write var(q) for the set of variables
in q. We may also treat, as usual, a CQ q as the set of atoms of q.

C.1 CQ Specializations

Recall that a contraction of a CQ q(⟨x⟩) is a CQ p(⟨x⟩) that can be
obtained from q by identifying 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
formal definition of CQ specialization follows:

Definition C.1. Consider a CQ q(⟨x⟩). A specialization of q is a pair
s = (p, V), where p is a contraction of q, and x ⊆ V ⊆ var(p).

Intuitively speaking, a specialization of a CQ q describes a way
how q can be homomorphically mapped to the chase of a database
D under a set Σ of TGDs. First, instead of mapping q to chase(D, Σ),
we injectively map p to chase(D, Σ), while the set of variables
V collects all the variables of p that are mapped to constants of
dom(D). The next result, which relies on the properties of the type
of an atom (see the proof of Lemma A.1 in Appendix ??) illustrates
the usefulness of CQ specializations. A CQ q is |V|–connected, for
V ⊆ var(q), if cₚ[|var(q)| \ V], that is, the subgraph of Gₚ induced by
var(q) \ V, is connected. We also write q[V] for the subquery of q
obtained after dropping the atoms of q[V], i.e., the atoms that contain
only variables of V. Given a database D, and a set Σ of TGDs, we
write chase(⟨D, Σ⟩) for the ground part of chase(D, Σ), that is, the
set consisting of all atoms in chase(D, Σ) with only constants from
dom(D).
We now define the notion of grounding of a specialization \( p, \nu \) of \( q \). The intention is to replace each maximally \([V]-\)connected component \( p' \) of \( p[V] \) with a guarded set of atoms, which is ought to be mapped to the ground part of the chase, that entails \( p' \) under the given set of TGDs. A CQ \( q \) is guarded if it has an atom, denoted \( \text{guard}(q) \), that contains all the variables of \( \text{var}(q) \), and full if all the variables of \( \text{var}(q) \) are answer variables.

**Definition C.3.** Consider a set \( \Sigma \) of TG\( s \) over \( T \), and a CQ \( q(\bar{x}) \) over \( T \). Let \( s = (p, \nu) \) be a specialization of \( q \) with \( p_1, \ldots, p_n \) being the maximally \([V]-\)connected components of \( p[V] \). A \( \Sigma \)-grounding of \( s \) is a CQ \( g_q(\bar{x}) \) over \( T \) of the form

\[
\exists \bar{z} \left( g_0 \land g_1 \land \cdots \land g_n \right)
\]

where

- \( g_0 \) is the full CQ consisting of the atoms of \( p[V] \),
- for each \( i \in [n] \), \( g_i \) is a guarded full CQ such that:
  - \( \text{var}(g_i) \subseteq (\text{var}(p_i) \land V) \cup \{ y_i^1, \ldots, y_i^d | \nu(T) = m \} \subseteq V \),
  - \( \text{var}(p_i) \land V \subseteq \text{var}\left(\text{guard}(g_i)\right) \), and
  - \( p_i \rightarrow \text{chase}(g_i, \Sigma) \) via a homomorphism that is the identity on \( \text{var}(p_i) \land V \), and
- \( \bar{z} = \left( \bigcup_{0 \leq i \leq n} \text{var}(g_i) \right) \setminus \bar{x} \).

Before giving the main lemma concerning groundings of CQ specializations, let us establish a useful auxiliary claim.

**Claim C.4.** Let \( \Sigma \) be a set of TG\( s \) over \( T \), and \( q(\bar{x}) \) a CQ over \( T \). Let \( g_q(\bar{x}) \) be a \( \Sigma \)-grounding of a specialization \( s = (p, \nu) \) of \( q \), and assume that \( g_q \rightarrow I \), for an instance \( I \), via a homomorphism \( \mu \). Then, \( p \rightarrow \text{chase}(I, \Sigma) \) via a homomorphism \( \xi \) with \( \mu(\bar{x}) = \xi(\bar{z}) \).

**Proof.** Let \( p_1, \ldots, p_n \) be the maximally \([V]-\)connected components of \( p[V] \). Assume that \( g_q(\bar{x}) \) is of the form

\[
\exists \bar{z} \left( g_0 \land g_1 \land \cdots \land g_n \right),
\]

where \( g_0 \) is the full CQ consisting of the atoms of \( p[V] \), and for each \( i \in [n] \), \( g_i \) is a guarded full CQ with the properties given in Definition C.3. Clearly, for each \( i \in [n] \), there exists a mapping \( \lambda_i \) that is the identity on \( \text{var}(p_i) \land V \) that maps \( p_i \) to \( \text{chase}(g_i, \Sigma) \). Since the sets of atoms \( p_1[V], \ldots, p_n[V] \) share only variables of \( V \), we conclude that \( \lambda = \bigcup_{i \in [n]} \lambda_i \) is a well-defined mapping that maps \( p \) to \( \text{chase}(g, \Sigma) \) with \( \lambda(\bar{x}) = \bar{z} \). Since, by hypothesis, \( g_q \rightarrow I \) via some \( \mu \), it is not difficult to show that \( \text{chase}(g, \Sigma) \) maps to \( \text{chase}(I, \Sigma) \) via \( \mu' \) that extends \( \mu \) and thus, \( \mu(\bar{x}) = \mu'(\bar{z}) \). Therefore, \( \xi = \mu' \circ \lambda \) maps \( p \) to \( \text{chase}(I, \Sigma) \) with \( \mu(\bar{x}) = \xi(\bar{z}) \), as needed.

We are now ready to show the main lemma about groundings of CQ specializations.

**Lemma C.5.** Consider a database \( D \), a set \( \Sigma \in \mathcal{G} \) over a schema \( T \), a CQ \( q(\bar{x}) \) over \( T \), and \( \bar{c} \in \text{dom}(D)^{|\bar{x}|} \). The following are equivalent:

1. \( \bar{c} \in q(\text{chase}(D, \Sigma)) \).
2. There exists a specialization \( s = (p, \nu) \) of \( q(\bar{x}) \) such that \( p \rightarrow \text{chase}(D, \Sigma) \) via an injective mapping \( \lambda \) with \( \mu(\bar{x}) = \bar{c} \).

**Proof.** (1) \( \Rightarrow \) (2). By Lemma C.2, there is a specialization \( s = (p, \nu) \) of \( q(\bar{x}) \), with \( p_1, \ldots, p_n \) being the maximally \([V]-\)connected components of \( p[V] \), such that \( p \rightarrow \text{chase}(D, \Sigma) \) via an injective mapping \( \lambda \) with \( \mu(\bar{x}) = \bar{c} \), and \( \lambda(\bar{z}) \subseteq \text{chase}(\Sigma, D, \Sigma) \). Consider the CQ \( g_q(\bar{x}) \) of the form

\[
\exists \bar{z} \left( g_0 \land g_1 \land \cdots \land g_n \right),
\]

where \( g_0 \) is the full CQ consisting of the atoms of \( p[V] \), and, for each \( i \in [n] \), \( g_i \) is the guarded full CQ obtained from \( \text{type}_{D,\Sigma}(a_i) \) by replacing each disjunct \( p \) of \( \lambda(y) \), where \( y \in \text{var}(p_i) \land V \), with the variable \( \mu(y) \). Finally, let \( \bar{z} = \left( \bigcup_{0 \leq i \leq n} \text{var}(g_i) \right) \setminus \bar{x} \). Clearly, \( g_q(\bar{x}) \) is a \( \Sigma \)-grounding of \( s \). Moreover, the mapping \( \mu = \lambda \cup \{ y_i^1 \mapsto c_{i}^j \}_{1 \leq j \leq |\nu(T)|, 1 \leq i \leq n} \) injectively maps each \( g_i \) to \( \text{chase}(D, \Sigma) \), since \( \text{type}_{D,\Sigma}(a_i) \subseteq \text{chase}(D, \Sigma) \), with \( \mu(\bar{x}) = \bar{c} \).

(2) \( \Rightarrow \) (1). Consider a specialization \( s = (p, \nu) \) of \( q \). Assume that there exists a \( \Sigma \)-grounding \( g_q(\bar{x}) \) of \( s \) that maps to \( \text{chase}(D, \Sigma) \) via an injective mapping \( \mu \) with \( \mu(\bar{x}) = \bar{c} \), actually, the fact that \( \mu \) is injective is not crucial here. By Claim C.4, we get that \( p \) maps to \( \text{chase}(D, \Sigma) \) via a homomorphism \( \xi \) with \( \xi(\bar{x}) = \bar{c} \). Clearly, by the definition of contractions, there exists a homomorphism \( h \), which is the identity on \( \bar{x} \), that maps \( q \) to \( p \). Therefore, \( \lambda = h \circ \mu \) maps \( q \) to \( \text{chase}(D, \Sigma) \) with \( \lambda(\bar{x}) = \bar{c} \), which implies that \( \bar{c} \in q(\text{chase}(D, \Sigma)) \).
(1) The subgraph of \(G \cup D_k\) induced by the elements of \(\text{dom}(D_k) \setminus \bar{c}\) has treewidth \(k\).

(2) \(D_k \rightarrow D\) via a homomorphism that is the identity on \(\bar{c}\).

(3) For every OMQ \(Q\) from \((G, \text{UCQ}_k)\) with data schema \(S\), \(\bar{c} \in Q(D)\) implies \(\bar{c} \in Q(D_k)\).

Let us stress that item (3) relies on the assumption that \(k \geq \text{ar}(S) - 1\) because, otherwise, there can be single atoms in \(D\) that cannot be part of any database of treewidth \(k\). In what follows, we say that \(D\) has treewidth \(k\) up to \(\bar{c}\) if the subgraph of \(G\) induced by the elements of \(\text{dom}(D) \setminus \bar{c}\) has treewidth \(k\).

**Lemma C.7.** Let \(Q\) be an OMQ from \((G, \text{UCQ})\) over \(T\) with data schema \(S \subseteq T\), and \(Q_k^k\) its \(\text{UCQ}_k\)-approximation for \(k \geq \text{ar}(T) - 1\).

1. \(Q_k^k \subseteq Q\).
2. For every \(S\)-database \(D\) and tuple \(\bar{c}\) over \(\text{dom}(D)\) such that \(D\) has treewidth at most \(k\) up to \(\bar{c}\), \(\bar{c} \in Q(D)\) implies \(\bar{c} \in Q_k^k(D)\).
3. For every OMQ \(Q'\) from \((G, \text{UCQ}_k)\) with data schema \(S\) such that \(Q' \subseteq Q_k^k\), it holds that \(Q' \subseteq Q_k^k\).

**Proof.** Let \(Q = (S, \Sigma, q(\bar{x}))\), and \(Q_k^k = (S, \Sigma, q_k^k(\bar{x}))\).

**Item (1).** Consider an \(S\)-database \(D\), and a tuple \(\bar{c}\) over \(\text{dom}(D)\) such that \(\bar{c} \in Q_k^k(D)\). Assume that \(\bar{c} \in Q_k^k(D)\), or, equivalently, there is a disjunct \(q_k^k\) of \(Q_k^k\) that maps to \(\text{chase}(D, \Sigma)\) via a homomorphism \(\mu'\) with \(\mu'(\bar{x}) = \bar{c}\). By definition, \(p_k^k\) is a \(\Sigma\)-grounding of a specialization \(s = (p_c, V)\) of some disjunct \(p\) of \(q\). By Claim C.4, \(p_k^k\) maps to \(\text{chase}(D, \Sigma)\) via a homomorphism \(\mu'\) with \(\mu'(\bar{x}) = \bar{c}\). Since, by definition of contractions, there exists a homomorphism \(h\), which is the identity on \(x\), that maps \(p\) to \(p_c\), we get that \(\lambda = \mu' \circ h\) maps \(p\) to \(\text{chase}(D, \Sigma)\) with \(\lambda(\bar{x}) = \bar{c}\). Since \(p\) is a disjunct of \(q\), we get that \(\bar{c} \in Q(D)\).

**Item (2).** Consider an \(S\)-database \(D\), and a tuple \(\bar{c}\) over \(\text{dom}(D)\) such that \(\bar{c} \in Q_k^k(D)\). Since \(\text{chase}(D, \Sigma)\) maps to \(\text{chase}(D_k, \Sigma)\), we get that \(\bar{c} \in Q(D_k)\), and thus, \(\bar{c} \in Q(D_k)\). Recall that \(D_k\) is the \(k\)-unraveling of \(D\) up to \(\bar{c}\). From item (2), we get that \(\bar{c} \in Q_k^k(D_k)\), or, equivalently, there is a disjunct \(p_k^k\) of \(Q_k^k\) that maps to \(\text{chase}(D, \Sigma)\) via a homomorphism \(\mu\) with \(\mu(\bar{x}) = \bar{c}\). We also know that \(D_k\) maps to \(D\) via a homomorphism \(\lambda\) with \(\lambda(\bar{c}) = \bar{c}\), which allows us to show that \(\text{chase}(D_k, \Sigma)\) maps to \(\text{chase}(D, \Sigma)\) via an extension \(\lambda'\) of \(\lambda\). Therefore, \(\lambda' = \lambda' \circ \mu\) maps \(p\) to \(\text{chase}(D, \Sigma)\) with \(\lambda'(\bar{x}) = \bar{c}\), and thus, \(\bar{c} \in Q_k^k(D)\), as needed.

**C.4 Finalizing the Proof of Proposition 5.2.**

Having Lemma C.7 in place, it is now easy to establish Proposition 5.2. Observe that (3) \(\Rightarrow\) (2) and (2) \(\Rightarrow\) (1) hold trivially. It remains to show that (1) \(\Rightarrow\) (3). By hypothesis, there exists an OMQ \(Q'\) from \((G, \text{UCQ}_k)\) such that \(Q' \equiv Q'\). By item (3) of Lemma C.7, \(Q' \subseteq Q_k^k\), which implies that \(Q \subseteq Q_k^k\). By item (1) of Lemma C.7, we also get that \(Q_k^k \subseteq Q\), and the claim follows.

**C.5 The Case \(k < \text{ar}(T) - 1\)**

Notice that Theorem 5.1 and Proposition 5.2 only cover the case where \(k \geq \text{ar}(T) - 1\), and also Lemma C.7, which gives the main technical properties of \(\text{UCQ}_k\)-approximations, requires this condition. We view the case \(k < \text{ar}(T) - 1\) as a somewhat esoteric corner case. In the context of Theorem 5.1, for example, we ask for an equivalent OMQ from \((G, \text{UCQ})\) when the arity of some relations \(R\) is so high that the actual UCQ in such an OMQ cannot contain a fact \(R(x_1, \ldots, x_n)\) unless some of the \(x_1, x_2\) are identical. We show that this case is also technically very different. In particular, our \(\text{UCQ}_k\)-approximations do not serve their purpose.

We consider the case where \(k < 1\) and \(\text{ar}(S) = 3\) and \(\text{ar}(T) = 6\) with \(S\) being the data schema and \(T\) being the extended schema. Let \(S = \{t_1, t_2\}\) with both \(t_1\) and \(t_2\) ternary. We use the Boolean CQ (for brevity, the existential quantifiers are omitted)

\[q(x) = R(x_1, x_2) \land R(x_2, x_3) \land R(x_2, x_4) \land R(x_3, x_4) \land A(x_2) \land A(x_3)\]

which is a core of treewidth \(2 > k\). Our ontology takes the form

\[\Sigma = \{T_1 \rightarrow q(x)\} \cup \{t_1 \cup t_1 \}

where \(\Sigma\) makes sure that there is an \(S\)-path of length \(2^n\) whenever there is a \(T_1\)-atom and \(\Sigma_2\) make sure that there is an \(S\)-path of length \(2^n - 1\) whenever there is a \(T_2\)-atom, where \(S\) is a binary relation symbol. We need auxiliary relation symbols to establish these paths, but we will make sure that they are of arity at least three so that they cannot be ‘seen’ by \(\text{UCQ}_k\)-approximations. No that while the presence of a \(T_1\)-atom implies that \(q(x)\) is true, the presence of a \(T_2\)-atom does not. In detail, \(\Sigma_1\) contains the following TGDs:

1. \(T_1(x_1, x_2, x_3) \rightarrow B^1_i(x_1, x_2, x_3)\) for \(1 \leq i \leq n\);
2. for \(1 \leq i \leq n\),
   \[B^1_i(x_1, x_2, x_3) \rightarrow \exists y_1 \exists y_2 \exists y_3 \ G(x_1, x_2, x_3, y_1, y_2, y_3)\]
   \[\land S(x_1, y_1)\]
3. for all \(i < n\), where \(G\) abbreviates \(G(x_1, x_2, x_3, y_1, y_2, y_3)\):
   \[G \land \bigwedge_{j=1}^{i-1} B^1_j(x_1, x_2, x_3) \land B^1_i(x_1, x_2, x_3) \rightarrow B^1_i(y_1, y_2, y_3)\]
4. for all \(i, j\) with \(j < i < n\), using the same abbreviation \(G\):
   \[G \land B^1_j(x_1, x_2, x_3) \land B^1_i(x_1, x_2, x_3) \rightarrow B^1_j(y_1, y_2, y_3)\]
   \[G \land B^1_j(x_1, x_2, x_3) \land B^1_i(x_1, x_2, x_3) \rightarrow B^1_j(y_1, y_2, y_3)\]

We leave the definition of \(\Sigma_2\) to the reader.
Proof. Take the S-database $D_1 = \{T_1(c_1, c_2, c_3)\}$. Then $Q$ evaluates to true on $D_1$ and thus so does $Q''$. Consequently, there is a CQ $p$ in $Q''$ and a homomorphism $h$ from $p$ to chase($D_1, \Sigma$). Since $p$ is of treewidth 1, it cannot contain any atoms $P(x)$ with $P$ of arity at least three and all variables in $x$ different, and thus $h$ is a homomorphism from $p$ to chase($D_1, \Sigma$), which is obtained from chase($D_1, \Sigma$) by removing all facts that use predicates of arity at least three (by choice of $D_1$ and definition of $\Sigma$, none of these facts contains repeated constants) and which takes the form of an S-path of length $2^n$. We aim to show that $p$ has at least $2^n$ atoms. To this end, it suffices to show that for every edge $S(a_1, a_2)$ in the path chase($D_1, \Sigma \setminus \{T_1(c_1, c_2, c_3)\}$), there is an atom $S(x_1, x_2) \in p$ such that $h(x_i) = a_i$ for $i \in \{1, 2\}$. Assume that this is not the case. Then clearly we also find a homomorphism from $p$ into an S-path of length $2^n - 1$, thus into chase($D_2, \Sigma$) where $D_2 = \{T_2(c_1, c_2, c_3)\}$. But $Q$ evaluates to false on $D_2$, which is a contradiction.

Lemma C.8 allows us to show that the UCQ$_k$-approximations $Q_k^a$ do not behave as intended when $k < ar(T) - 1$. In particular, although the OMQ $Q = (S, \Sigma, q)$ defined above is UCQ$_{k}$-equivalent, we can show that $Q''$ is not equivalent to $Q_k^a$. By Lemma C.8, we get that for any $Q'' = (G, \text{UCQ})$ that is equivalent to $Q$, $Q''$ necessarily contains a CQ of exponential size. On the other hand, the UCQ$_k$-approximation $Q_k^a$ of $Q$ contains only CQs of polynomial size since $ar(T)$ is a constant. Therefore, $Q$ is not equivalent to $Q_k^a$.

For $k < ar(T) - 1$, we conjecture that every UCQ$_k$-approximation (that is based on the same ontology as the original OMQ $Q$ and has the fundamental property of being equivalent to $Q$ when the later is uniformly UCQ$_k$-equivalent, must contain a CQ of double exponential size. This is not true our notion of UCQ$_k$-approximation in Definition C.6, where each CQ is, in general, of exponential size.

D PROOF OF THEOREM 5.4

D.1 Preliminaries

We start by showing that existential quantifiers can be eliminated:

**Theorem D.1.** Consider an OMQ $Q = (S, \Sigma, q) \in (G, \text{UCQ})$, where $\Sigma$ and $q$ are over the schema $T \supseteq S$. We can construct an OMQ $Q' = (S, \Sigma' q') \in (G \cap \text{FULL}, \text{UCQ})$, where $\Sigma'$ and $q'$ are over $T$, such that, for every S-database $D$, $Q(D) = Q'(D)$.

Before giving the proof of the above result, let us recall a key property of linear TGDs, that is, UCQ-rewriteability, shown in [15].

**Proposition D.2.** Given a set $\Sigma \subseteq L$ and a UCQ $q$, both over a schema $T$, we can construct a UCQ $q'$ over $T$ such that, for every T-database $D$, $q'(\text{chase}(D, \Sigma)) = q'(D)$.

We are now ready to give the proof.

Proof. (Theorem D.1) The desired OMQ $Q'$ can be actually extracted from the proof of Lemma A.3. From that proof, we know that we can construct, for some schema $T'$ disjoint from $T$,

- a set $\Sigma_1 \in G \cap \text{FULL}$, over $T'$,
- a set $\Sigma_2 \in G \cap \text{FULL}$ over $T \cup T'$, where the predicates of $T$ (resp., $T'$) occur only in the body (resp., head) of a TGD, and
- a set $\Sigma_3 \subseteq L$ over $T \cup T'$,

such that, for every S-database $D$,

$$Q(D) = q'(\text{chase}(D, \Sigma_1 \cup \Sigma_2, \Sigma_3)).$$

Notice that we can further assume that $\Sigma_1 \cup \Sigma_2$ consists of TGDs with only one atom in the head; simply split a multi-head full TGD into several single-head TGDs, one for each head atom, that have the same body. Since $\Sigma_3$ is set of linear TGDs, by Proposition D.2, we can construct a UCQ $\tilde{q}$ over $T \cup T'$ such that

$$q'(\text{chase}(D, \Sigma_1 \cup \Sigma_2)) = \tilde{q}(\text{chase}(D, \Sigma_1 \cup \Sigma_2)).$$

To obtain the desired set $\Sigma'$ and UCQ $q'$, it remains to eliminate the predicates of $T'$. This can be achieved by simply unfolding the atoms in $\tilde{q}$ with a predicate from $T'$ using the single-head TGDs of $\Sigma_2$, and get a new UCQ $\tilde{q}$ over $T$. It is clear that

$$q'(\text{chase}(D, \Sigma_1 \cup \Sigma_2)) = \tilde{q}(\text{chase}(D, \Sigma_1)).$$

Therefore, the claim follows with $Q' = (S, \Sigma_1, \tilde{q})$.

Let $S$ be a schema, $D$ an S-database, and $\hat{a} \subseteq \text{dom}(D)$ a guarded set in $D$. We aim to define a potentially infinite database $D^\omega$, the guarded unraveling of $D$ at $\hat{a}$. In parallel with $D^\omega$, we define a tree decomposition $(T, \chi)$, $T = (V, E)$, of the Gaifman graph of $D^\omega$ whose width is the maximum arity of relation names in $S$ minus one.

Let $V$ be the set of sequences $v = a_0 \cdots a_n$ of guarded sets in $D$ such that $a_0 = \hat{a}$ and $a_i \cap a_{i+1} = \emptyset$ for $0 \leq i < n$. Further, let

$$E = \{(v, v') \in V \times V | v' = a \hat{b}$$

for some $\hat{b}

We associate with each $v \in V$ a database $D(v)$ and then define $\chi(v)$ to be dom($D(v)$). This completes the definition of $(T, \chi)$ and allows us to define $D^\omega$ as $\bigcup_{v \in V} D(v)$.

Take an infinite supply of copies of every $a \in \text{dom}(D)$. Set $b^V = a$ if $b$ is a copy of $a$, and $a^\chi = a$. For $v \in V$, define $D(v)$ and its domain dom($D(v)$) by induction on the length of the sequence $v$. For $v = \hat{a}$, $D(v) = D^\omega$, the restriction of $D$ to those atoms with only constants in $\hat{a}$. To define $D(v)$ when $v = a_0 \cdots a_n$ with $n > 0$, take for any $a \in a_n \setminus a_{n-1}$ a fresh copy $a'$ of $a$ and define $D(v)$ with domain

$$\text{dom}(D(v)) = \{b \in \text{dom}(D(a_0 \cdots a_{n-1}) | b^V \in a_n \cap a_{n-1}) \cup a'| a \in a_n \setminus a_{n-1})\}$$

such that $b \mapsto b^V$ is an isomorphism from $D(v)$ to $D(a_n)$.

Injective homomorphisms play a crucial role in our proof of Theorem 5.4. Here, we make some basic observations pertaining to such homomorphisms. For a database $D$, a CQ $q(\hat{x})$, and a tuple of distinct constants $\hat{a}$, we write $D \models q(\hat{a})$ if $D \models q(\hat{a})$ and all homomorphisms $h$ from $q$ to $D$ with $h(\hat{x}) = \hat{a}$ are injective. Note that since the constants in $\hat{a}$ are distinct, the condition $h(\hat{x}) = \hat{a}$ does not conflict with injectivity. Here, ‘isomorphic’ stands for ‘injectively only’. The following is a simple observation, see [7] for a proof.

**Lemma D.3.** If $D \models q(\hat{a})$, for $D$ a potentially infinite database, $q$ a CQ, and $\hat{a}$ a tuple of distinct constants, then $D \models^{\text{iso}} q_{\text{c}}(\hat{a})$ for some contraction $q_{\text{c}}$ of $q$.

The next lemma is an important ingredient to the proof of Theorem 5.4. We write $D, \hat{a} \rightarrow E, \hat{b}$ to indicate that there is a homomorphism $h$ from database $D$ to database $E$ with $h(\hat{a}) = \hat{b}$.

**Lemma D.4.** Let $Q = (S, \Sigma, q) \in (G \cap \text{FULL}, \text{UCQ})$ and $\hat{a}$ a tuple of distinct constants. Then for every S-database $D$ with $D \models Q(\hat{a})$, there is an S-database $\tilde{D}$ such that the following are satisfied:
(1) $\overline{D} \models Q(\overline{a})$ and $\overline{D}, \overline{a} \rightarrow D, \overline{a}$;

(2) if chase$(\overline{D}, \Sigma) \models q_r(\overline{a})$, for $q_r$ a contraction of $q$, then there is no $S$-database $D'$ and contraction $q'_r$ of $q$ where $D', \overline{a} \rightarrow \overline{D}, \overline{a}$, chase$(D', \Sigma) \models q'_r(\overline{a})$, and $q_r \neq q'_r$ is a contraction of $q'_r$.

**Proof.** We start with $\overline{D} = D$. Then, of course, Condition 2 is not guaranteed to be satisfied. We thus iteratively replace $\overline{D}$ with more suitable databases, as follows. As long as there are an $S$-database $D'$ and contractions $q_r, q'_s$ of $q$ such that chase$(\overline{D}, \Sigma) \models q_r(\overline{a})$, $D', \overline{a} \rightarrow D', \overline{a}$, chase$(D', \Sigma) \models q'_s(\overline{a})$, and $q_r \neq q'_s$ is a contraction of $q'_s$, replace $\overline{D}$ by $D'$. Informally, this iterative process terminates since we lose at least one contraction $q_r$ of $q$ with chase$(\overline{D}, \Sigma) \models q_r(\overline{a})$ in every step. We refer to [7] for details.

It is clear that the resulting $\overline{D}$ satisfies Condition (2). Condition (1) is satisfied as well: we have chase$(\overline{D}, \Sigma) \models q_r(\overline{a})$ for some contraction $q_r$ of $q$, thus $\overline{D} \models Q(\overline{a})$.

**Lemma D.5.** Given an OMQ $Q = (S, \Sigma, q)$ from $(G \land FULL, UCQ)$ and an $S$-database $\overline{D}$, it is decidable whether Conditions 1 and 2 from Lemma D.4 hold.

**Proof.** As Condition 1 is clearly decidable, we concentrate on Condition 2. We start with a central observation. Let $q_r$ be a contraction of $q$ and let $q_r'$ be the UCQ that consists of all contractions of $q_r$ that are distinct from $q_r$. Then checking whether chase$(D', \Sigma) \models q_r'(\overline{a})$, for some $S$-database $D'$ with $D', \overline{a} \rightarrow D', \overline{a}$, is equivalent to checking that the OMQs $(S, \Sigma, q_r')$ and $(S, \Sigma, q_r'')$ are not equivalent on $S$-databases $D'$ with $D', \overline{a} \rightarrow D', \overline{a}$. As a consequence, we can decide Condition 2 by checking that there are no contractions $q_r, q_r'$ of $q$ such that $q_r$ is a contraction of $q_r'$, $q_r \neq q_r'$, chase$(\overline{D}, \Sigma) \models q_r(\overline{a})$ and $(S, \Sigma, q_r'')$ are not equivalent on $S$-databases $D'$ with $D', \overline{a} \rightarrow D', \overline{a}$. That is, in the statement ‘chase$(\overline{D}, \Sigma) \models q_r(\overline{a})$, we have dropped the $\overline{D}$ that is present in Condition 2. This is not harmful since if chase$(\overline{D}, \Sigma) \models q_r(\overline{a})$, but chase$(\overline{D}, \Sigma) \not\models q_r'(\overline{a})$, we can simply replace $q_r$ by a contraction of itself that satisfies this stronger condition.

The width of an OMQ $Q = (S, \Sigma, q)$ from $(G \land FULL, UCQ)$, denoted $\omega(Q)$, is the maximum number of variables in the body of a rule in $\Sigma$ or in a CQ in $q$.

To establish decidability of Condition 2, it suffices to show that given OMQs $Q_1$ and $Q_2$ from $(G \land FULL, UCQ)$ over schema $S$, an $S$-database $D$, and a tuple of constants $\overline{a}$, it is decidable whether there exists an $S$-database $D'$ such that $D \models Q_1(\overline{a})$, $D \not\models Q_2(\overline{a})$, and $D, \overline{a} \rightarrow D', \overline{a}$. We note that this is the case iff there exists a database with these properties that is of treewidth $\omega(Q_1)$.

For an OMQ $Q = (S, \Sigma, q)$ from $(G \land FULL, UCQ)$ we construct a Guarded Second Order (GSO) [25] formula $\phi_S$ such that for every $S$-database $D$ and tuple of constants $\overline{a}$, $D \models Q(\overline{a})$ iff $D \models \phi_S(\overline{a})$. Let $J_1, \ldots, J_n$ be the relation symbols in $\Sigma$ and reserve fresh relation symbols $J'_1, \ldots, J'_m$ and $J''_1, \ldots, J''_n$ of the same arity. Further let:

- $\phi'_S$ (resp. $\phi''_S$) be the conjunction of all TGDs in $\Sigma$, viewed as universally quantified FO formulas, with each relation symbol $J_i$ replaced by $J'_i$ (resp. $J''_i$).

- $\phi_S$ be a formula which says that if $J_i$ is in $S$, then the extension of $J_i$ is included in that of $J'_i$;

- $\phi$ be a formula which says that the extension of each $J'_i$ is included in the extension of $J''_i$, with at least one inclusion being strict.

Then $\phi_S$ is the following formula:

$$\exists J'_1, \ldots, J'_n (\phi_S \land \phi_Z \land q \land \forall J''_1, \ldots, J''_n (\phi_i \rightarrow \neg \phi_Z)).$$

Note that the extension of $J'_1, \ldots, J'_n$ represents the chase of the input database with $\Sigma$.

We also take a formula $\phi_S$ that evaluates to true exactly on those $S$-databases $D$ such that $D, \overline{a} \rightarrow D, \overline{a}$. Then, we have to decide whether there exists an $S$-database $D$ of treewidth at most $\omega(Q_1)$ such that $D \models \phi_S, \neg \phi_Z \land \phi_D$. Note that on sparse structures, GSO has the same expressive power as MSO [13]. As databases of bounded treewidth are sparse [19], it follows that GSO over such databases is decidable.

For $k, \ell \geq 1$, the $k \times \ell$-grid is the (undirected) graph with vertex set $\{(i, j) \mid 1 \leq i \leq k \land 1 \leq j \leq \ell\}$ and an edge between $(i, j)$ and $(i', j')$ iff $|i - i'| + |j - j'| = 1$. A graph $H$ is a minor of a graph $G$ if $H$ is isomorphic to a graph that can be obtained from a subgraph of $G$ by contracting edges. Equivalently, $H$ is a minor of $G$ if there is a minor map from $H = (V_H, E_H)$ to $G = (V_G, E_G)$, that is, a mapping $\mu : V_H \rightarrow 2^{\mathbb{G}}$ with the following properties for all $v, w \in V_H$:

- $\mu(v)$ is non-empty and connected in $G$;
- $\mu(v)$ and $\mu(w)$ are disjoint whenever $v \neq w$;
- if $\{v, w\} \in E_H$, then there are $v' \in \mu(v)$ and $w' \in \mu(w)$ such that $\{v', w'\} \in E_G$.

We say that $\mu$ is onto if $\bigcup_{v \in V_H} \mu(v) = V_G$. For a database $D$, we say that $a \in dom(D)$ is isolated in $D$ if there is only a single atom $R(\overline{a}) \in D$ with $a \in \overline{a}$. When $k$ is understood from the context, we use $K$ to denote $[k]$. Note that Point (3) of the following theorem can be viewed as a form of ontology requirement for the homomorphism $h_0$ from Point (1): for the fact $R(a_1, \ldots, a_n) \in D$ mentioned in that condition, it is possible to find a fact $R(c_1, \ldots, c_n) \in D_G$ that maps to it.

**Theorem 6.1 (Grohe).** Given an undirected graph $G$, a $k \geq 1$, a connected $S$-database $D$, and a set $A \subseteq dom(D)$ such that the restriction $G^D_A$ of the Gaifman graph of $D$ to vertices $A$ contains the $k \times k$-grid as a minor, one can construct in time $f(k)$-poly$(||G||, ||D||)$ an $S$-database $D_G$ with $dom(D) \setminus A \subseteq dom(D_G)$ such that:

1. there is a surjective homomorphism $h_0$ from $D_G$ to $D$ that is the identity on $dom(D) \setminus A$;
2. $G$ contains a $k$-clique iff there is a homomorphism $h$ from $D$ to $D_G$ such that $h$ is the identity on $dom(D) \setminus A$ and $h_0(\cdot) = h(\cdot)$ is the identity, and
3. if $R(a_1, \ldots, a_n) \in D$, $h_0(b_i) = a_i$ for each $i \in [n]$, and $\{b_1 \mid a_i$ non-isolated in $D \}$ is a clique in the Gaifman graph of $D_G$, then $D_G$ contains an atom $R(c_1, \ldots, c_n)$ where $c_i = b_i$ if $a_i$ is non-isolated in $D$ and $h_0(c_i) = a_i$ for each $i \in [n]$.

**Proof.** The construction of $D_G$ is a slight extension of the original construction from [26]. The extension is necessary because we have to treat the constants from $dom(D) \setminus A$ in a special way whereas in [26] there is no set $A$ or, in other words, $A = dom(D)$.

---

We use the variant of GSO whose the syntax is identical to that of Second Order Logic, but the second-order quantifiers range only over guarded relations, see [25].
Let $\mu$ be a minor map from the $k \times k$-grid to $G^D_A$. Since $D$ is connected, we can assume w.l.o.g. that $\mu$ is onto. To attain Point (3) of Theorem 6.1, we actually have to choose $\mu$ a bit more carefully. We first proceed with the construction and argue that Points (1) and (2) are satisfied, and then give details on Point (3). Fix a bijection $\rho$ between $[K]$ and the set of unordered pairs over $[K]$. For $p \in [K]$, let $i \in p$ be shorthand for $i \in \rho(p)$. Let $G^D_A = (V, E)$.

The domain of $D_G$ is

$$(\text{dom}(D) \setminus A) \cup \{(v, e, i, p, a) \in V \times E \times [k] \times [k] \times A \mid (v, e) \in i \in p, a \in \mu(i, p).$$

Before we say what the atoms of $D_G$ are, let us define the homomorphism $h_0 : \text{dom}(D_G) \rightarrow \text{dom}(D)$ from Point (1) by taking the identity on $\text{dom}(D) \setminus A$ and the projection to the last component for all other constants from $\text{dom}(D_G)$. We extend $h_0$ to tuples over $\text{dom}(D_G)$ in the expected way.

We now define $D_G$ to contain every atom $R(b, b)$ a tuple over $\text{dom}(D_G)$, such that $R(b_0(b), b)$ and for all constants $b, b'$ in $\bar{a}$ that are of the form $b = (v, e, i, p, a)$ and $b' = (v', e', i', p', a')$,

(C1) $i = i'$ implies $v = v'$ and (C2) $p = p'$ implies $e = e'$.

This finishes the construction of $D_G$.

Point (1) follows directly from the construction. The proof of Point (2) is a slight variation of the proofs of Lemma 4.3 (for the `$\Rightarrow$' direction) and Lemma 4.4 (for the `$\Leftarrow$' direction) from [26]. In fact, the homomorphism $h$ constructed in the proof of Lemma 4.3 simply needs to be extended to be the identity on $\text{dom}(D) \setminus A$. In Lemma 4.4, the assumption that $D$ is a core has been replaced here with the condition that $h_0(R(\cdot, \cdot))$ is the identity. This, however, is exactly what is derived from the assumption that $D$ is a core in the original proof. No further changes to the proof are required despite the presence of the elements in $\text{dom}(D) \setminus A$.

For Point (3), assume that $R(a_1, \ldots, a_n) \in D$, $h_0(b_j) = a_j$ for $1 \leq j \leq n$, and $S = \{b_j \mid a_j$ non-isolated in $D\}$ is a clique in the Gaifman graph of $D_G$. As $D$ is connected, $S$ cannot be empty. We can assume w.l.o.g. that the minor map $\mu$ from the $k \times k$-grid onto $G^D_A$ is such that one of the following holds:

1. There is an $a_0$ that is non-isolated in $D$ such that $a_0 \in A$ and if $i, p$ are such that $a_0 \in \mu(i, p)$, then all $a_j \in \mu(A)$ are also in $\mu(i, p)$; in this case, $b_j$ must take the form $(v, e, i, p, a_0)$;

2. There is no $a_0$ that is non-isolated in $D$ such that $a_0 \in A$ and there is $i, p$ such that all $a_j \in A$ that are isolated in $D$ are also in $\mu(i, p)$; in this case, choose $v, e$ arbitrarily such that $v \in e$ if $i \in p$.

Define $c_1, \ldots, c_n$ by setting

$$c_j := \begin{cases} (v, e, i, p, a) & \text{if } a_j \in A \text{ is isolated in } D \\ b_j & \text{otherwise.} \end{cases}$$

It can be verified that $(v, e, i, p, a)$ is a constant in $D_G$ if $a_j \in A$ is isolated in $D$, due to our assumptions (1) and (2). Moreover, $h_0(c_j) = h_0(b_j)$ for $1 \leq j \leq n$ and Conditions (C1) and (C2) are satisfied for the fact $R(c_1, \ldots, c_n)$, which is thus in $D_G$. Finally, $h_0(c_i) = a_i$ for $1 \leq i \leq n$, by definition of $h_0$.

D.2 The Reduction

Recall that our goal is to establish the following lower bound:

**THEOREM 6.6.** Fix $r \geq 1$. Let $\mathcal{O}$ be a recursively enumerable class of OMQs from $(\mathcal{G}, \mathcal{UCQ})$ over a schema of arity $r$, and, for each $k \geq 1$, $\mathcal{O} \not\subseteq (\mathcal{G}, \mathcal{UCQ})^r_k$. Then, p-OMQ-Evaluation$(\mathcal{O})$ is $W[1]$-hard.

The proof is by fpt-reduction from p-Clique, a W[1]-hard problem. Assume that $G$ is an undirected graph and $k \geq 1$ a clique size, given as an input to the reduction. By Robertson and Seymour’s Excluded Grid Theorem, there is an $f$ such that every graph of treewidth exceeding $f$ contains the $k \times k$-grid as a minor [32]. We may assume, w.l.o.g., that $f$ exceeds the fixed arity $r$. By our assumption on $\mathcal{O}$, we find an OMQ $Q = (S, \Sigma, q)$ from $\mathcal{O}$ that is not in $(\mathcal{G}, \mathcal{UCQ})^r$. Since the choice of $Q$ is independent of $G$ and since it is decidable whether an OMQ from $(\mathcal{G}, \mathcal{UCQ})$ is in $(\mathcal{G}, \mathcal{UCQ})^r$ by Theorem 5.1, we can enumerate the OMQs from $\mathcal{O}$ until we find $Q$.

By Theorem D.1, we can assume, w.l.o.g., that $Q = (S, \Sigma, q)$ is from $(\mathcal{G} \land \text{FULL}, \mathcal{UCQ})$. It might of course be that the OMQ from $(\mathcal{G} \land \text{FULL}, \mathcal{UCQ})$ is not in $\mathcal{O}$, but we will see that this is not a problem. Let $q = q_1 \lor \cdots \lor q_m$ and let $Q_i = (S, \Sigma, q_i)$ for $1 \leq i \leq m$. We can assume w.l.o.g. that $Q_i \not\subseteq Q_j$ for all $i \neq j$ because if this is not the case, then we can drop $q_j$ from $q$ in $Q$ and the resulting OMQ is equivalent to $Q$. Also, we exhaustively replace CQs in $Q$ with $(q_i)^g_o$ as defined in the context of $\mathcal{UCQ}_r$-approximations (see Definition C.6), whenever the resulting OMQ is equivalent to $Q^g$. Since $Q$ is not in $(\mathcal{G}, \mathcal{UCQ})^r$, the final OMQ must contain a $q_o$ that is not replaced by $(q_o)^g_o$ and, in particular, we must have $Q \not\subseteq Q'$ where $Q'$ is obtained from $Q$ by replacing $q_o$ with $(q_o)^g_o$. We thus find an S-database $D_0$ and tuple of constants $\bar{a}_0$ such that $D_0 \models Q_\omega(\bar{a}_0), D_0 \not\models (Q_\omega)^g_o(\bar{a}_0)$, and $D_0 \not\models Q_i(\bar{a}_0)$ for all $i \neq \ell$. By duplicating constants, we can achieve that all constants in $\bar{a}_0$ are distinct. We also assume that Condition 2 of Lemma D.4 is satisfied for $Q = Q_\omega, q = q_\omega, D = D_0$, and $\bar{a} = \bar{a}_0$. If it is not, we can apply that lemma and replace $D_0$ with the resulting $D_0$. By Condition (1), $D_0 \models Q_\omega(\bar{a}_0), D_0 \not\models (Q_\omega)^g_o(\bar{a}_0)$, and $D_0 \not\models Q_i(\bar{a}_0)$ for all $i \neq \ell$.

Since the properties of $D_0$ are independent of $G$ and due to Lemma D.5, we can find $D_0$ by enumeration. However, $D_0$ is still not as required and needs to be manipulated further to make it suitable for the reduction. Before we can carry out the actual manipulation, we need some preliminaries.

Recall that for a guarded set $\bar{a} \subseteq \text{dom}(D_0)$, $D_0^\bar{a}$ denotes the guarded unraveling of $D_0$ starting at $\bar{a}$. An atomic query $(\mathcal{A}Q)$ takes the form $R(\cdot, \cdot)$, i.e., it has a single atom and no quantified variables.

**LEMMA 7.5.** $D_0 \models (S, \Sigma, p)(\bar{a}^*)$ iff $D_0^\bar{a} \models (S, \Sigma, p)(\bar{a}^*)$ if $p$ is an AQ and all constants in $\bar{a}'$ are also in $\bar{a}$.

Of course, $D_0^\bar{a}$ can be infinite. By compactness, however, there is a finite $D_0 \subseteq D_0^\bar{a}$ such that Lemma D.7 is satisfied for all (finitely many) AQs $p$ that use only symbols from $\Sigma$. For brevity, we say that $D_0$ satisfies Lemma D.7 for all relevant AQs. We can find $D_0$ by constructing $D_0^\bar{a}$ level by level and deciding after each such extension whether we have found the desired database, by checking the condition in Lemma D.7 for all relevant AQs.

A **diversification** of $D_0$ is a database $D$ obtained from $D_0$ by replacing every atom $R(\bar{a}) \in D_0$ with a (potentially empty) finite

---

We rely on the key property of $\mathcal{UCQ}_r$-approximations given by item (2) of Lemma C.7.
set of atoms \(R(a_1', \ldots, R(a_n'))\) such that each \(a_i'\) is obtained from \(a\) by replacing some constants that do not occur in \(a_0\) with fresh constants. A constant \(a \in \text{dom}(D)\) is old if it already occurs in \(D_0\) and fresh otherwise. For a fresh constant \(b\), we use \(b\) to denote the constant in \(D_0\) that it was introduced for. For old constants \(a, a'\), \(a = a'\). This extends to tuples of constants from \(D\) in a component-wise fashion. Note that all fresh constants in \(D\) are isolated in \(D\) and that all constants from \(a_0\) still occur in \(D\) as old constants. If \(D_1\) and \(D_2\) are diversifications of \(D_0\), we write \(D_1 \preceq D_2\) if there is a homomorphism from \(D_1\) to \(D_2\) that is the identity on all old constants.

**Example D.8.** Assume that \(D = \{R(a, b, c), S(a, b, d)\}\) and \(a_0 = ()\). Then \(D_1 = \{R(a, b', c'), S(a, b, d')\}\) and \(D_2 = \{R(a, b, c'), S(a, b, d)\}\) are diversifications of \(D\) and \(D_1 \preceq D_2\).

If \(D\) is a diversification of \(D_0\), we use \(D^\bullet\) to denote the database obtained from \(D\) by adding each database \(D_i\) such that for some atom \(R(\bar{a})\) in \(D\), we obtain \(D_0\) from \(D_0^\bullet\) by renaming the constants in \(a_i\) to those in \(a\). A constant \(a \in \text{dom}(D^\bullet)\) is old if it is an old constant from \(\text{dom}(D)\), and fresh otherwise.

For what follows, we choose a \(\preceq\)-minimal diversification \(D_0\) of \(D\) such that \(D_0^\bullet \models Q_w(\bar{a}_0)\).

**Example D.9.** Assume that \(Q_w\) is a Boolean CQ that takes the form of an \(n \times m\)-grid using the binary relations \(X\) and \(Y\). Let \(S = (X', Y')\) with both relations ternary and let \(\Sigma = (X'(x, y, z) \rightarrow X(x, y), Y'(x, y, z) \rightarrow Y(x, y))\). Further let \(D_0\) be the database
\[
\{X'(a_i, j_i, k_i, l_i) \mid 1 \leq i \leq m \land 1 \leq j < n\} \cup \{Y'(a_i, j_i, k_i, l_i) \mid 1 \leq i < m \land 1 \leq j \leq n\}
\]
Then the following database is a \(\preceq\)-minimal diversification of \(D_0\) such that \(D_0^\bullet \models Q_w(\bar{a}_0)\):
\[
\{X'(a_i, j_i, k_i, l_i) \mid 1 \leq i \leq m \land 1 \leq j < n\} \cup \{Y'(a_i, j_i, k_i, l_i) \mid 1 \leq i < m \land 1 \leq j \leq n\}.
\]

We next observe some basic properties of \(D_1\).

**Lemma D.10.**
1. \(D_1^\bullet \models Q_w(\bar{a}_0)\) and \(D_1^\bullet \models Q(\bar{a}_0)\) for all \(w \neq w'\).
2. \(D_1\) has treewidth exceeding \(t\) up to \(a_0\).
3. \(D_1^\bullet\) satisfies Condition 2 of Lemma D.4.

**Proof.** For Point 1, we have \(D_1^\bullet \models Q_w(\bar{a}_0)\) by choice of \(D_1^\bullet\). By construction of \(D_1^\bullet\), we have \(D_1^\bullet, a_0 \rightarrow D_0, a_0\) and thus \(D_1^\bullet \not\models Q_i(\bar{a}_0)\) for all \(i \neq w\). We proceed with Point 2, noting that we can argue in the same way to show that \(D_0^\bullet \not\models Q_w(\bar{a}_0)\). Thus Point 2 of Lemma C.7 implies that the treewidth of \(D_1^\bullet\) up to \(a_0\) exceeds \(t\). Note that the treewidth of \(D_1^\bullet \setminus D_0\) is bounded by the fixed arity \(r\). As we assume \(t > r\), \(D_1\) has treewidth exceeding \(t\) up to \(a_0\).

Now for Point 3, that is, \(D_1^\bullet\) satisfies Condition 2 of Lemma D.4. Assume to the contrary that there is an S-database \(D'\) and contractions \(q_e, q_{e'}\) of \(Q_w\) such that \(\text{chase}(D_1^\bullet, \Sigma) \models Q_e(\bar{a}_0), D', a_0 \rightarrow D_1^\bullet, a_0\), \(\text{chase}(D_1^\bullet, \Sigma) \models Q_{e'}(\bar{a}_0)\), and \(q_e \neq q_{e'}\) is a contraction of \(q_e\).

We use a case distinction: If \(q_e = q_{e'}\), then \(\text{chase}(D_0, \Sigma) \models Q_e(\bar{a}_0)\).

**Proof.** Assume to the contrary that there is a homomorphism \(h\) from \(Q_w(\bar{a}_0)\) to \(\text{chase}(D_1^\bullet, \Sigma)\) with \(h(x) = \bar{a}_0, R(\bar{a}) \in D_1\), and \(a_1, a_2\) are old constants in \(\bar{a}\) such that there is no path in the Gaifman graph \(G^{D^\bullet}(Q_w)\) as required by Lemma D.11. We argue that we can then find an S-database \(D_2\) with \(D_2 \prec D_1\) and \(D_2^\bullet \models Q_w(\bar{a}_0)\), in contradiction to the choice of \(D_1\).

Let \(D_2\) be obtained from \(D_1\) by replacing \(R(\bar{a})\) with the atoms \(R(\bar{a}_1), R(\bar{a}_2)\) where \(\bar{a}_1\) is obtained from \(\bar{a}\) by replacing \(a_i\) with a fresh constant \(b_i\) and each fresh constant with a new fresh constant. Clearly, \(D_2\) is a diversification of \(D_0\) and \(D_2 \prec D_1\). We convert \(h\) into a homomorphism \(h'\) from \(Q_w\) to \(\text{chase}(D_1^\bullet, \Sigma)\) such that \(h'(\bar{x}) = \bar{a}_0\), which contradicts the choice of \(D_1\). As a preparation, let us discuss the construction of \(D_1^\bullet\) and \(D_1^\bullet\). During the construction of \(D_1^\bullet\), we have attached a copy \(D_0\) of \(D_0^\bullet\) at \(\bar{a}\). In the construction of \(D_1^\bullet\), we have attached copies \(D_0\) and \(D_0\) of the same \(D_0\) at \(a_1\) and \(a_2\). The constants in \(\text{dom}(D_0) \setminus \text{dom}(D_1)\) are the only constants in \(D_1^\bullet\) that may not be in \(D_1^\bullet\). For \(i \in \{1, 2\}\), fix an isomorphism \(\iota_i\) between \(D_0\) and \(D_0\). Note that \(\iota_i\) is the identity on \(a_{2-i}\) and in fact on all old constants in \(\bar{a}\) except \(a_i\).

Now define \(h'(x)\) as follows, for every variable \(x\) in \(Q\):
1. If \(h(x) \in \text{dom}(D_1^\bullet) \setminus \text{dom}(D_0), h'(x) = h(x)\)
2. If \(h(x) \notin \text{dom}(D_0)\):
   a. If \(G^{D_0}(Q_w)\) contains a path \(x_i, \ldots, x_p\) such that \(h(x_1) = a_i\) for \(i \in \{1, 2\}\), \(y_p = x, h(x_2), \ldots, h(x_{p-1})\) are in \(\text{dom}(D_0), h(x_2) = \iota_2(i_2(h(x)))\); 
   b. If \(h(x)\) is still undefined, then \(h'(x) = \iota_1(h(x))\).

Note that \(h'\) is defined properly: due to our assumption that there is no path as in Lemma D.11, the precondition of 2b cannot be satisfied for both \(i = 1\) and \(i = 2\). Also note that if \(S(\bar{x}) \in Q_w\) with \(\bar{x} \subseteq \text{dom}(D_0), h'(x) = \iota_1(h(x))\), for some \(i \in \{1, 2\}\). It remains to show that \(h'\) is a homomorphism from \(Q_w\) to \(\text{chase}(D_1^\bullet, \Sigma)\). Let us first observe three helpful claims. Recall that \(\iota\) maps each constant in a diversification of \(D_0\), such as \(D_1\) and \(D_2\), to
to the original constant that it was introduced for. The first claim is a consequence of Lemma D.7 and the fact that since $D_1$ and $D_2$ are diversifications of $D_0$, $\bar{D}$ defines a homomorphism from $D_1$ to $D_0$ for each $j \in \{1, 2\}$ that can be extended to a homomorphism from $D_j^*$ to $D_0$ and further from chase($D_j^*, \Sigma$) to chase($D_0, \Sigma$).

**Claim 1.** Let $\bar{b}$ be a guarded set in $D_j$, $j \in \{1, 2\}$. Then $R(\bar{b}) \in$ chase($D_j^*, \Sigma$) iif $R(\bar{b}) \in$ chase($D_j^*, \Sigma$) for all facts $R(\bar{b})$.

Here is the second claim.

**Claim 2.** Let $\bar{b}$ be a guarded set in $D_0$. Then $R(\bar{b}) \in$ chase($D_j^*, \Sigma$) implies $R(\bar{b}) \in$ chase($D_j^*, \Sigma$) for all facts $R(\bar{b})$ and $i \in \{1, 2\}$.

Assume to the contrary that $R(\bar{b}) \notin$ chase($D_j^*, \Sigma$). Set $D = \text{chase}(D_j^*, \Sigma) \cup \{R'(\bar{e}) \mid R(\bar{e}) \in \text{chase}(D_j^*, \Sigma)\}$. Using the fact that $\Sigma$ is guarded, it can be shown that $D$ is a model of $\Sigma$.

The proof of the third claim is identical to that of Claim 2.

**Claim 3.** Let $\bar{e}$ be a guarded set in some tree $D_j^*$, $j \in \{1, 2\}$, $\bar{e} \neq \bar{a}$. Then $R(\bar{e}) \in$ chase($D_j^*, \Sigma$) implies $R(\bar{e}) \in$ chase($D_j^*, \Sigma$) for all atoms $R(\bar{e})$.

Now for the proof that $h^*$ is a homomorphism from $q_{\Sigma}$ to $\text{chase}(D_j^*, \Sigma)$. Let $S(\bar{x}) \in q_{\Sigma}$.

First assume that $h(\bar{x})$ contains only constants from dom($D_1$). It follows from (two applications of) Claim 1 that $S(h(\bar{x})) \in$ chase($D_j^*, \Sigma$) implies $S(h^*(\bar{x})) \in$ chase($D_j^*, \Sigma$).

Next assume that $h(\bar{x})$ contains a constant from a tree dom($D_i$), $\bar{e} \neq \bar{a}$, that is not in dom($D_1$). By construction of $D_j^*$ and by definition of the chase, this implies that $h(\bar{x}) \subseteq \text{dom}(D_j^*)$. We can thus apply Claim 3 to infer that $S(h^*(\bar{x})) \in$ chase($D_j^*, \Sigma$) implies $S(h^*(\bar{x})) \in$ chase($D_j^*, \Sigma$).

Finally assume that $h(\bar{x})$ contains a constant from dom($D_0$) \ dom($D_1$). Then $h(x) \subseteq \text{dom}(D_0)$. Thus $h^*(\bar{x}) = h(\bar{x})$ for some $i \in \{1, 2\}$. We can thus apply Claim 2 to infer that $S(h^*(\bar{x})) \in$ chase($D_j^*, \Sigma$) implies $S(h^*(\bar{x})) \in$ chase($D_j^*, \Sigma$).

Clearly, neither $D_1$ nor $D_j^*$ are guaranteed to be connected. In particular, $D_0$ might be disconnected and even if it is connected the diversification might replace all constants in an atom with fresh constants and in this way make $D_1$ disconnected. By Point 2 of Lemma D.10, however, the restriction of $G^{D_1}$ to dom($D_1$) \ $\bar{a}_0$ has a maximal connected component that is of treewidth exceeding $t$. Let $A$ be the set of constants in that component. Moreover, let $D$ be the maximal connected component of $D_1$ that contains all constants from $A$ (it might additionally contain constants from $\bar{a}_0$) and let $D_j^*$ denote the part of $D_j^*$ that is disconnected from $D \subseteq D_j^*$. By choice of $D$, $A$, and $t$, the restriction of $G^D$ to $A$ contains the $k \times K$-grid as a minor. We can thus apply Theorem 6.1 to $G$, $K$, and $D$, and $A$, obtaining an S-database $D_0$ and a homomorphism $h_0$ from $D_0$ to $D$ such that Points 1 to 3 of that theorem are satisfied. Let $D_j^*$ be obtained by starting with $D_0$ and then disjointly adding, for each guarded set $\bar{a}$ in $D_0$, such that the restriction of $h_0$ to $\bar{a}$ is injective, a copy of the database $D_0$ that was attached to $R(h_0(\bar{a})) \in D_1$ during the construction of $D_j^*$, identifying the root of this copy with $\bar{a}$.

The fpt-reduction of p-Clique then consists in computing $Q$ and $D_j^*$ from $G$ and $k \geq 1$. We mean here the original $Q$ that is guaranteed to be in $\mathcal{O}$ and where $\Sigma$ is from $G$ but not necessarily from $\mathcal{O} \cap \mathcal{FULL}$.

**D.3 Correctness of the Reduction**

We show in the subsequent lemma that $D_j^* \models Q(\bar{a}_0)$ if and only if $G$ has a $k$- clique. Clearly, we can work with the version of $Q$ here in which $\Sigma$ is from $\mathcal{O} \cap \mathcal{FULL}$ since it is equivalent to the original $Q$.

**Lemma D.12.** $G$ has a $k$-clique iiff $D_j^* \models Q(\bar{a}_0)$.

**Proof.** The `only if` direction is easy. If $G$ has a $k$-clique, then by Point 2 of Theorem 6.1 and since $A$ contains none of the constants from $\bar{a}_0$, there is a homomorphism $h$ from $D$ to $D_j^*$ such that $h(\bar{a}_0) = \bar{a}_0$ and $h(\bar{h}(\cdot))$ is the identity. It is straightforward to extend $h$ to a homomorphism from $D_j^*$ to $D_j^* = D_j^* \sqcup D_j^*$; in particular, if $R(\bar{a})$ is an atom in $D$, then the restriction of $h_0$ to $\bar{a}$ must be injective since $h_0(\bar{h}(\cdot))$ is the identity and thus a copy of the subtree that was attached to $\bar{h}(\bar{a})$ in the construction of $D_j^*$ from $D_1$ was attached to $\bar{a}$ in the construction of $D_j^*$ from $D_0$. Consequently, $D_j^* \models Q(\bar{a}_0)$ implies $D_j^* \models Q(\bar{a}_0)$.

For the `if` direction, assume that $D_j^* \models Q(\bar{a}_0)$. The surjective homomorphism $h_0$ from $D_0$ to $D$ given by Theorem 6.1 can be extended to a homomorphism from $D_j^* = D_j^* \sqcup D_j^*$ to $D_j^*$ and from chase($D_j^*$, $\Sigma$) to chase($D_j^*$, $\Sigma$).

For brevity, we denote this extension also with $h_0$. For later use, we observe that

$(P1)$ $b \in \text{dom}(D_j^*)$ iff $h_0(b) \in \text{dom}(D)$;
$(P2)$ $b_1, b_2$ are in the same tree of $D_j^*$ iff $h_0(b_1), h_0(b_2)$ are in the same tree of $D_j^*$.

So $h_0$ witnesses $D_j^*, \bar{a}_0 \rightarrow D_j^*, \bar{a}_0$. Thus $D_j^* \models Q(\bar{a}_0)$ for all $\bar{a}_0$ implies $D_j^* \models Q(\bar{a}_0)$ for all $\bar{a}_0 \notin Q$. It follows that $D_j^* \models Q(\bar{a}_0)$ and thus $\text{chase}(D_j^*, \Sigma) \models Q(\bar{a}_0)$.

By Lemma D.3, we find a contraction $q_c$ of $q_{\Sigma}$ such that $\text{chase}(D_j^*, \Sigma) \models q_c(\bar{a}_0)$. As witnessed by $h_0$, chase($D_j^*$, $\Sigma$), $\bar{a}_0 \rightarrow$ chase($D_j^*$, $\Sigma$), $\bar{a}_0$. Consequently chase($D_j^*$, $\Sigma$) $\models q_c(\bar{a}_0)$. Thus we find a contraction $q_c$ of $q_c$ such that chase($D_j^*$, $\Sigma$) $\models q_c(\bar{a}_0)$. We must have $q_c = q_c$ since $D_j^*$ satisfies Condition 2 of Lemma D.4, via Lemma D.10.

Let $h_0$ be a homomorphism from $q_c$ to chase($D_j^*$, $\Sigma$). The composition $g = h_0(h(\cdot))$ is a homomorphism from $q_c$ to chase($D_j^*$, $\Sigma$). Since chase($D_j^*$, $\Sigma$) $\models q_c$, this homomorphism must be injective. Consequently, also $h_0$ is injective.

Since $h_0$ is injective, we can construct a function $h_0^{-1}$ as follows:

1. start with the restriction of $h_0$ to the range of $h$;
2. take the inverse;
3. restrict to old constants in $D$.

*The injectivity requirement implies that we can add these copies as they are, without duplicating any constants in them.*

*The extension needs not be surjective.*
It follows from Lemma D.11 that the range of $h_0$ includes all old constants in $D$, and thus so does the domain of $h_0'$. In particular, it thus also contains all constants in $\tilde{a}_0$. Trivially, $h_0(h'_0(\cdot))$ is the identity and $h_0''$ is the identity on all constants from $\text{dom}(D) \setminus A \subseteq \tilde{a}_0$ (since $h_0$ is the identity on such constants).

We are going to show that a certain extension of $h_0''$ is a homomorphism from $D$ to $D_G$ that is the identity on $\text{dom}(D) \setminus A$, and thus Point 2 of Theorem 6.1 yields that $G$ contains a $k$-clique.

Let $R(\bar{a}) \in D$. By definition of $g$ and since the range of $g$ contains all old constants in $D$, we find for each old $a \in a$ a variable $x_a \in \bar{x}$ and a $b_a \in \text{dom}(D_G')$ such that $h(x_a) = b_a$ and $h_0(b_a) = a$. Let $\Gamma$ be the set of all $b_a$. By Lemma D.11, for any two old constants $a_1, a_2$ in $\bar{a}$, there is a path $x_1, \ldots, x_n$ in the Gaifman graph of $D[\bar{x}]$, such that $g(x_1) = a_1, g(x_n) = a_2$, and $g(x_2), \ldots, g(x_{m-1})$ are in $\text{dom}(D_G)$. By Properties P1 and P2, it follows that for any two $b_1, b_2$ in $\Gamma$, there is a path $x_1, \ldots, x_n$ in the Gaifman graph of $D[\bar{q}_w]$ such that $h(x_1) = b_1, h(x_m) = b_2$, and $h(x_2), \ldots, h(x_{m-1})$ are in the same tree of $D_G'$. The fact that $h_0$ maps all elements of $\Gamma$ to constants in $D$ and Property P1 further imply $b_1, b_2 \in \text{dom}(D_G)$. By construction of $D_G'$, there must be thus a fact in $D_G$ that contains both $b_1$ and $b_2$ (the one for which the tree was added that $h(x_2), \ldots, h(x_{m-1})$ are in). It follows that $\Gamma$ forms a clique in the Gaifman graph of $D_G$. Since (the non-extended) $h_0''$ is surjective, for each fresh $a \in \bar{a}$ we find some $b_a \in \text{dom}(D_G)$ such that $h_0(b_a) = a$. We can now apply Point 3 of Theorem 6.1 to $R(\bar{a})$ and the $b_a$ to obtain an extension of $h_0''$ to the fresh constants in $\bar{a}$ such that $R(h'_0(\bar{a})) \in D_G$. Note that the set $\{b_a | a \text{ a non-isolated in } D\}$ can only contain old constants since all fresh constants are isolated in $D$, it thus is a subset of $\Gamma$ and forms a clique in the Gaifman graph of $D_G$. The extension is such that if $a$ is fresh and covered by the extension, then $h_0''(a) = c$ for some $c$ with $h_0(c) = a$. Thus, $h_0(h_0''(\cdot))$ is still the identity and $h_0''$ is the identity on $\text{dom}(D) \setminus A$. Also note that this extension can be done independently for all $R(\bar{a}) \in D$ because all fresh constants in $D$ are isolated in $D$.

Finally, Point 2 of Theorem 6.1 yields that $G$ has a $k$-clique.

**E PROOF OF PROPOSITION 5.5**

Recall that, given two CQs $S_0 = (\Sigma, q)$ and $S' = (\Sigma, q')$ over $S$, we write $S \subseteq S'$ if $q(D) \subseteq q'(D)$ for every $S$-database $D$ that satisfies $\Sigma$. We also write $S \equiv S'$ if $S \subseteq S'$ and $S' \subseteq S$. Analogously, we define the notions that consider unrestricted (not necessarily finite) instances. We write $S \equiv_{\text{unf}} S'$ if $q(I) \subseteq q'(I)$ for every (possibly infinite) $S$-instance $I$ that satisfies $\Sigma$, and $S \equiv_{\text{unf}} S'$ if $S \equiv_{\text{unf}} S'$ and $S' \equiv_{\text{unf}} S$. The next simple result states that for guarded TGDs the notions $\equiv$ and $\equiv_{\text{unf}}$ coincide, which, unsurprisingly, relies on the fact that guarded TGDs are finitely controllable.

**Lemma E.1.** Consider two CQs $S = (\Sigma, q(\bar{x}))$ and $S' = (\Sigma, q'(\bar{x}))$ from $(\mathbb{G}, \mathbb{UCQ})$. Then, $S \equiv_{\text{unf}} S'$ if and only if $S \equiv S'$.

**Proof.** The ($\equiv$) direction holds trivially. For the ($\equiv_{\text{unf}}$) direction, assume that $S \equiv_{\text{unf}} S'$. There are two cases: (i) $S \equiv S'$, or (ii) $S' \equiv_{\text{unf}} S$. We consider only the first case; the second case is shown analogously. Since $S \equiv_{\text{unf}} S'$, we get that $\bar{x} \notin q'(\text{chase}(\Sigma, \bar{x}))$. With $Q'$ being the OMG $\text{omq}(S')$, we get that $\bar{x} \notin Q'(D[q])$. Since, by Lemma 6.6 and Theorem 6.7, guarded TGDs are finitely controllable, we conclude that there exists a finite model $M$ of $D[q]$ and $\Sigma$ such that $\bar{x} \notin q'(M)$. However, it holds trivially that $\bar{x} \in q(M)$, and thus, $S \not\equiv S'$, which in turn implies that $S \not\equiv_{\text{unf}} S'$, as needed.

Having the above lemma in place, we can now give the proof of Proposition 5.5. Assume that $S = (\Sigma, q)$ is over a schema $\mathbb{S}$. Since, by Proposition 5.2, an OMG from $(\mathbb{G}, \mathbb{UCQ})$ is UCQ$_k$-equivalent iff it is uniformly UCQ$_k$-equivalent, it suffices to show that, for each $k \geq 1$, the following are equivalent:

1. There is $q' \in \text{UCQ}_k$ over $S$ such that $S \equiv (\Sigma, q')$.  
2. There is $q' \in \text{UCQ}_k$ over $S$ such that $\text{omq}(S) \equiv (\Sigma, q')$.

1. $\Rightarrow$ (2). By hypothesis, there exists $q' \in \text{UCQ}_k$ over $S$ such that $S \equiv (\Sigma, q')$. It is easy to show that $\text{omq}(S) \equiv (\Sigma, q')$; for brevity, let $Q = \text{omq}(S)$ and $Q' = (\Sigma, q')$. Consider an arbitrary $S$-database $D$. Since, by construction, chase($D$) is a model of $\Sigma$, and, by Lemma E.1, $S \equiv_{\text{unf}} (\Sigma, q')$, we get that $q(\text{chase}(D, \Sigma)) = q'(\text{chase}(D, \Sigma'))$. Therefore, $Q(D) = Q'(D)$, as needed.

2. $\Rightarrow$ (1). By hypothesis, there exists $q' \in \text{UCQ}_k$ over $S$ such that $\text{omq}(S) \equiv (\Sigma, q')$; again, let $Q = \text{omq}(S)$ and $Q' = (\Sigma, q')$. Consider an arbitrary $S$-database $D$ such that $D$ is a model of $\Sigma$. Observe that $q(D) = Q(D)$ and $q'(D) = Q'(D)$. Since $Q \equiv Q'$, we get that $q(D) = q'(D)$, and the claim follows.

**F PROOF OF PROPOSITION 5.8**

Consider an OMG $Q = (\Sigma, q(\bar{x}))$ from $\text{omq}(\mathbb{G})$, an $S$-database $D$, and a tuple $\bar{a} \in \text{dom}(D)$. We are going to construct an $S$-database $D^*$ such that $D^* \equiv \Sigma$, and $\bar{a} \in \text{dom}(D^*)$. We can now apply Construction 6.7.2, $\bar{a} \in \text{dom}(D^*)$, or, equivalently (due to Proposition 2.2), $\bar{a} \in \text{chase}(D, \Sigma)$, as claimed.

**F.1 Construction of the Database $D^*$**

Note that the definition of $D^*$ has been already given in the main body of the paper (Section 6.2). However, we repeat it here for the sake of readability. We first define $D^*$ as the database

$$D \cup \{R(\bar{a}) \in \text{chase}(D, \Sigma) | \bar{a} \subseteq \text{dom}(D)\}.$$

Let $A$ be the family of all maximal tuples $\bar{a}$ over $\text{dom}(D)$ that are guarded in $D^*$, i.e., there is an atom $R(\bar{b}) \in D^*$ such that $\bar{a} \subseteq \bar{b}$. Fix an arbitrary tuple $\bar{a} \in A$. Since, by Theorem 6.7, the class $\mathbb{G}$ is strongly finitely controllable, and also finite witnesses are realizable, we can compute an instance $M(D^*_a, \Sigma, n) \in \text{fmods}(D^*_a, \Sigma)$, where $n$ is the number of variables in $q$, such that for each CQ $q'$ of arity $|\bar{a}|$ with at most $n$ variables, it holds that

$$\bar{a} \subseteq q'(M(D^*_a, \Sigma, n)) \Rightarrow \bar{a} \subseteq q'(\text{chase}(D^*_a, \Sigma)).$$

W.l.o.g., we assume that $\text{dom}(M(D^*_a, \Sigma, n)) \subseteq \text{dom}(D)$, for every two distinct tuples $\bar{a}, \bar{b} \in A$. The database $D^*$ is

$$D^* \cup \bigcup_{\bar{a} \in A} M(D^*_a, \Sigma, n).$$

**F.2 Correctness of the Reduction**

The next lemma, which is actually Lemma 6.8 in the main body of the paper, shows that $D^*$ is the desired database.

**Lemma F.1.** It holds that:

1. $D^* \models \Sigma$.
2. $\bar{a} \in \text{chase}(D, \Sigma) \iff \bar{a} \in q(D^*)$. 

(3) There exists a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $D^*$ can be constructed in time $|D|^O(1) \cdot f(|Q|)$.

Proof. Item (1). Consider a TGD $\sigma \in \Sigma$ of the form $\phi(x, \bar{y}) \rightarrow \exists x \phi(x, \bar{z})$, with guard $\sigma = R(u)$. Let $q_\phi = \exists \bar{y} \phi(x, \bar{y})$ and $q_\phi = \exists \bar{y} \phi(x, \bar{z})$. Fix an arbitrary tuple $\bar{c} \in \text{dom}(D^*)^{||\bar{x}||}$, and assume that $\bar{c} \in q_\phi(D^*)$. We need to show that $\bar{c} \in q_\phi(D^*)$. Since $\bar{c} \in q_\phi(D^*)$, we get that $\phi(\bar{x}, \bar{y}) \rightarrow D^*$ via a homomorphism $h$ such that $h(\bar{x}) = \bar{c}$. Clearly, $R(h(\bar{u})) \in M(D(\bar{x}, \Sigma, n))$ for some tuple $\bar{a} \in A$ that contains all the constants $h(\bar{u}) \cap \text{dom}(D)$. It is then easy to see that, due to guardedness, $h(\phi(\bar{x}, \bar{y})) \subseteq M(D(\bar{a}, \Sigma, n))$. Since $M(D(\bar{a}, \Sigma, n))$ is a model of $\Sigma$, we get that $\bar{c} \in q_\phi(M(D(\bar{a}, \Sigma, n)))$. Therefore, due to the monotonicity of CQs, $\bar{c} \in q_\phi(D^*)$, as needed.

Item (2). Let us first concentrate on the (⇒) direction. Since $D^* \models \Sigma$, we get that chase(D, $\Sigma$) → $D^*$ via a homomorphism $h$ that is the identity on dom(D); the latter holds due to Proposition 2.2. Moreover, by hypothesis, $q \rightarrow$ chase(D, $\Sigma$) via a homomorphism $\mu$ that maps the answer variables $\bar{x}$ of $q$ to $\bar{c}$. Therefore, $\mu \circ h$ maps $q$ to $D^*$ and $\bar{c}$ to $\bar{c}$, which in turn implies that $\bar{c} \in q(D^*)$. We now show the (⇐) direction. By hypothesis, $q \rightarrow D^*$ via a homomorphism $h$ such that $h(\bar{x}) = \bar{c}$. For each tuple $\bar{a} \in A$, let $q_a(\bar{x}_a)$ be the maximal subquery of $q$ such that $h(\bar{q}_a) \subseteq M(D(\bar{a}, \Sigma, n))$ and $h(\bar{x}_a) = \bar{c}$, i.e., $\bar{x}_a$ are the variables of $q_a$ that are mapped to constants of dom(D). By (⇒), we get that $q_a \rightarrow \text{chase}(D(\bar{a}, \Sigma))$ via a homomorphism $h_a$ that maps $\bar{x}_a$ to $\bar{a}$. It should be clear that

\[
\mu \equiv \bigcup_{\bar{a} \in A} h_a
\]

is well-defined mapping that maps $q$ to chase(D, $\Sigma$) with $\mu(\bar{x}) = \bar{c}$.

This implies that $\bar{c} \in q(\text{chase}(D, \Sigma))$, and the claim follows.

Item (3). This is shown by exploiting the fact that the database $D^*$ can be constructed in time $|D|^O(1) \cdot g(|Q|)$ for some computable function $g : \mathbb{N} \rightarrow \mathbb{N}$, and the fact that the cardinality of $A$, as well as the size of $D^*_a$ for some $\bar{a} \in A$, do not depend on $D$. For showing that $D^*$ can be constructed in the claimed time, we exploit Lemma A.4, and the fact that we can construct a set $(\Sigma) \in G \cap FUL\Sigma$ such that $D^* = \text{chase}(D(\Sigma))$; the latter is inherited from [24].

G PROOF OF PROPOSITION 5.11

Let $S = (\Sigma, q(\bar{x}))$ be a CQS from $(\text{PG}_{\text{sm}}, \text{UCQ})$ over a schema S of arity r. Via a proof similar to that of Proposition 5.5, we get that:

- $S$ is uniformly $\text{UCQ}_{\text{eq}}$-equivalent iff omq(S) is.
- $S \cong S_{\text{eq}}$ iff omq(S) $\equiv$ omq($S_{\text{eq}}$).

Let $Q_S = \text{omq}(S)$ and $Q_{S_{\text{eq}}} = \text{omq}(S_{\text{eq}})$. Thus, for showing Proposition 5.11, it suffices to show that the following are equivalent:

(1) $Q_S$ is uniformly $\text{UCQ}_{\text{eq}}$-equivalent.

(2) $Q_S \equiv Q_{S_{\text{eq}}}^*$.

The proof of the next lemma is along the lines of the proof of Lemma C.7. The only difference is that we rely on the following fact: given a database D of treewidth at most k up to $\bar{c}$, chase(D, $\Sigma$) has treewidth at most k up to $\bar{c}$. The latter heavily exploits the bound $m$ on the number of head atoms in the TGDs of $\Sigma$.

Lemma G.1. For every OMQ $Q' = (\Sigma, q')$ from $(\text{PG}_{\text{sm}}, \text{UCQ})$ such that $Q' \subseteq Q_S$, it holds that $Q' \subseteq Q_{S_{\text{eq}}}^*$.

Having Lemma G.1 in place, it is now easy to establish the above equivalence, i.e., $Q_S$ is uniformly $\text{UCQ}_{\text{eq}}$-equivalent iff $Q_S \equiv Q_{S_{\text{eq}}}^*$. Observe that the (⇐) direction holds trivially. It remains to show the (⇒) direction. By hypothesis, there exists an OMQ $Q'' = (\Sigma, q'')$ from $(\text{PG}_{\text{sm}}, \text{UCQ})$ such that $Q_S \equiv Q''$. By Lemma G.1, $Q'' \subseteq Q_{S_{\text{eq}}}^*$, which in turn implies that $Q_S \subseteq Q_{S_{\text{eq}}}^*$. The fact that $Q_{S_{\text{eq}}}^* \subseteq Q_S$ holds by construction, and the claim follows.

H PROOF OF THEOREM 5.13

We provide a complete proof of Theorem 5.13 here. Our proof does not use Grohe’s database as defined in [26], but a variant of it which we define below. We start by introducing several important notions.

Let $G = (V, E)$ and $H = (V', E')$ be simple graphs. A minor map from H to G is a mapping $\mu : V' \rightarrow V^2$ such that:

(1) For every $v' \in V'$, $\mu(v')$ is nonempty and connected in $G$.

(2) For every $v', u' \in V'$, $v' \neq u'$, $\mu(v')$, $\mu(u')$ are disjoint.

(3) For every edge $\{v', u'\} \in E'$, there are nodes $v \in \mu(v')$ and $u \in \mu(u')$ such that $\{v, u\} \in E$.

Such a minor map is said to be onto if, in addition, $\bigcup_{v' \in V'} \mu(v') = V$.

Let $G$ and $H$ be simple graphs. It is easy to see that $H$ is a minor of $G$ iff there is a minor map $\mu$ from $H$ to $G$. If, in addition, $G$ is connected, then there exists an onto minor map $\mu$ from $H$ to $G$.

H.1 A Variation of Grohe’s Database

Let $G = (V, E)$ be an undirected graph, $k \geq 1$ be an integer, and $K = (\overline{\overline{\overline{\overline{k}}}})$. Suppose that $D$ and $D'$ are $S$-databases with $D \subseteq D'$ such that there exists a set $A \subseteq \text{dom}(D)$ for which there is a minor map $\mu$ from the $(k \times K)$-grid onto $G_D^{\text{omq}}$. We fix a injection $\chi$ that assigns to each 2-element subset of $[k]$ a value in $[K]$. We define a database $D^* := D'(G, D, D', A, \mu)$ as follows.

The domain. Every element of $D^*$ is either an element in dom($D'$) \ A or a tuple of the form $(v, e, i, p, z)$ where $v \in V$, $e \in E$, $i \in [k]$, $p$ is a two-element subset of $[k]$, and $z \in A$. We note here that dom($D'$) does not necessarily contain all elements of the previous form since some of them might not appear in any of its atoms.

The facts. A labelled clique in G is any partial mapping $\eta$ from $[k]$ to $V$ such that, for every different i, j in the domain of $\eta$, we have that $\eta(i)$ and $\eta(j)$ are adjacent in $G$. We say that an element $z \in \text{dom}(D')$ is covered by a labelled clique $\eta$ if $z \notin A$, or there exist i, j, t in domain of $\eta$ such that $z \in \mu(i, \chi((i, t)))$. Then, for every fact $R(z) \in D'$ and every labelled clique $\eta$ covering every element mentioned in z, it is the case that $D'$ contains the fact $R(\eta_z)$ where $\eta_z$ is obtained by replacing, in $\eta$, every element $z \in A$ with

\[
(\eta(i), (\eta(j), (\eta(t), i, (j, t), z)).
\]

where $\mu(i, \chi((i, t))) = z$.

Additionally, we define the projection

\[
h_0 : \text{dom}(D') \rightarrow \text{dom}(D')
\]

such that

\[
\begin{cases}
h_0(a, e, i, p, z) = z, & \text{if } z \in A, \text{ and} \\
h_0(y) = y, & \text{otherwise}
\end{cases}
\]
Although we avoid to introduce Grohe’s original database [26], it is useful to compare it with ours. In particular, consider the case when \( D = D’ \) and \( A = \text{dom}(D) \), and thus we have a database \( D^* = D(G, D, D, \text{dom}(D), \mu) \). Grohe’s database \( D_{\text{Grohe}} = D_{\text{Grohe}}(G, D, \mu) \) then satisfies the following:

- \( \text{dom}(D_{\text{Grohe}}) \) is contained in the set of tuples of the form \((v, e, i, p, z)\), for \( v \in V, e \in E, i \in [k], p \) is a two-element subset of \([k]\), and \( z \in \text{dom}(D) \). Therefore, \( h_0 \) is well-defined on \( \text{dom}(D_{\text{Grohe}}) \).
- Also, \( D^* \subseteq D_{\text{Grohe}} \).

In addition, the following crucial property holds for \( D_{\text{Grohe}} \):

**Lemma H.1.** ([26]) Let \( h \) be an homomorphism from \( D \) to \( D_{\text{Grohe}} \) such that \( h_0 \circ h \) is the identity. Then \( G \) contains a \( k \)-clique.

Our variant still enjoys many of the good properties of Grohe’s database, and some additional properties concerning the satisfaction of frontier-guarded TGDs, as shown in the following Theorem (which is essentially an equivalent reformulation of Theorem 7.1) for the case when CQs allow for free variables.

**Lemma H.2.** Let \( G, k, D, D’, A, D^* = D^*(G, D, D’, A, \mu) \), and \( h_0 \) be as defined above. The following statements hold:

1. There is a polynomial-time algorithm that, given \( G, D, D’, A, \) and \( \mu \), computes \( D^* \).
2. \( h_0 \) is a surjective homomorphism from \( D^* \) to \( D’ \).
3. \( G \) contains a \( k \)-clique iff there is a homomorphism \( h \) from \( D \) to \( D^* \) such that \( h_0(h(i)) \) is the identity on \( A \).
4. If \( D^* \models \Sigma \), and all other CQs of size at most \( 3 \cdot r \) in \( G \) is contained in a clique of size \( 3 \cdot r \cdot m \), then \( D’ \models \Sigma \).

Proof. Items (1) and (2) follow directly from the definition of \( D^* \) and \( h_0 \). However, for item (3) it is necessary to note that since the arity of the schema \( S \) is bounded by the fixed integer \( r \), in the construction of \( D \) it is enough to consider only all labelled cliques of size at most \( r \).

Let us prove (4). Although (\( \equiv \)) can be obtained easily by mimicking the proof in [26], we shall give a proof using Lemma H.1. Assume that \( h \) is a homomorphism from \( D \) to \( D^* \) such that \( h_0 \circ h \) is the identity on \( A \). Then the restriction \( h_{|A} \) of \( h \) to \( A \) defines a homomorphism from \( D_{|A} \) to \( D^* \) such that \( h_0 \circ h_{|A} \) is the identity. But then \( h_{|A} \) corresponds to a homomorphism from \( D_{|A} \) to \( D(G, D, D, \text{dom}(D_{|A}), \mu) \) with \( h_0 \circ h_{|A} \) being the identity. Since \( D(G, D, D, \text{dom}(D_{|A}), \mu) \) is contained in the original Grohe’s database, \( D_{\text{Grohe}}(G, D_{|A}, \mu) \), it follows from Lemma H.1 that \( G \) contains a \( k \)-clique. For the converse (\( \Rightarrow \)) assume that \( G \) has a \( k \)-clique, or alternatively, that there is a labelled clique \( \eta \) in \( G \) with domain \([k]\).

Then \( \eta \) covers all elements in \( \text{dom}(D^*) \) and, hence, the mapping \( h : \text{dom}(D) \rightarrow \text{dom}(D^*) \) with \( h(z) = \eta_z \), for every \( z \in \text{dom}(D) \), defines a homomorphism from \( D \) to \( D^* \) satisfying the conditions.

Let us finally show (4). Consider an arbitrary frontier-guarded TGD \( \phi(x, y) \rightarrow \exists y \psi(x, z) \) in \( S \), and let \( (a, b) \) be an arbitrary tuple in the evaluation of \( \phi(x, y) \) over \( D^* \). Since \( h_0 \) is a homomorphism from \( D^* \) to \( D’ \), we have that \( D^* \) contains all facts in \( \phi(h_0(a), b) \). Therefore, since \( D’ \models \Sigma \), there is a tuple \( \phi \) of elements in \( \text{dom}(D^*) \), for \( |\phi| = |\Sigma| \), such that \( D^* \) contains every atom in \( \psi(h_0(a), \phi) \). Since there is an atom guarding \( \phi \) in \( \phi \), it follows from the definition of \( D^* \) that \( a = h_0(\tilde{a}) \) for some labelled clique \( \eta \) covering \( h_0(\tilde{a}) \). In addition, \( \Sigma \in \text{PG}_m \) and hence \( (h_0(\tilde{a}), \phi) \) can have at most \( r \cdot m \) elements. It follows from the assumptions of item (4) that there is labelled clique \( \eta' \) in \( G \), extending \( \eta \), that covers all elements mentioned by \( \phi \) in addition to all elements in \( h_0(\tilde{a}) \) already covered by \( \eta \). Thus, \( D^* \) contains every atom in \( \psi(h_0(\tilde{a}), \phi, \tilde{\eta}) = \psi(\tilde{a}, \tilde{\eta}) \).

**H.2 A Crucial Lemma**

Our proof borrows several ideas and techniques from the proof of Grohe’s Theorem. However, the adaptation of such techniques is non-trivial for the reasons that we have already discussed in Section 5.2, and we briefly recall here again. First, we cannot construct an arbitrary database \( D \), but we need to construct one that satisfies the given set \( \Sigma \) of frontier-guarded TGDs. Second, the notion of core of a CQ, which is crucial for the proof of Grohe, cannot be directly used in our context. Instead, we need to develop a technical lemma, which, intuitively speaking, states that some subsets of our CQs behave like cores for the sake of our proof.

**Lemma H.3.** Fix \( t \geq r \cdot m \). There is a computable function that takes as input a CQ8 \( S = (\Sigma, q(\delta)) \) from \((\text{PG}_m, \text{CQ})\) that is not uniformly \( \text{CQ}_r \)-equivalent, and a positive integer \( t \geq 1 \), and outputs a CQ \( p(\delta) \), a subset \( X \) of the existentially quantified variables of \( p \), and a CQ \( p^*(\delta) \), such that:

1. \( q \equiv_X p \).
2. \( D[p^*] \models \Sigma \).
3. \( D[p] \subseteq D[p^*] \).
4. For every homomorphism \( h \) from \( p \) to \( p^* \) with \( h(\delta) = \delta \), we have that \( h(X) = X \).
5. \( \text{The treewidth of } G_{\Sigma}^p \text{ is larger than } t \).
6. For every CQ \( p^* (\delta) \) with at most \( t \) variables, if \( \tilde{x} \in \text{dom}(D[p^*]) \) then also \( \tilde{x} \in p^* (\text{domain}(D[p], \Sigma)) \).

Proof. Let us assume that \( S \) is defined over schema \( S \), and that \( q(\delta) \) has \( n > 0 \) variables, and that \( t \) is the maximum number of variables appearing in the body of some TGD in \( S \). We set \( s^* = \max(s, n, t) \).

First of all, we can compute by exhaustive search a CQ \( p(\delta) \) with a minimum number of variables that satisfies \( q \equiv_X p \). This is because the notion \( \equiv_X \) is decidable, for \( \Sigma \in \text{PG} \). In fact, to check whether \( p \equiv_X q \) one can construct the finite instance \( M = (M(D[p], \Sigma, n)) \) from Definition 6.5, which exists by the strong finite controllability of \( \text{PG} \) as stated in Theorem 6.7, and then check whether \( \tilde{x} \in q(M) \). We know that the latter holds iff \( \tilde{x} \in q(\text{domain}(D[p], \Sigma)) \). From Proposition 4.5 we get that \( p \equiv_X q \). Analogously, one can check \( q \equiv_X p \).

We have then that item (1) holds by definition. We now explain how to build \( p^* \) and \( X \). We need some preparation. First, we shall assume that for every TGD \( \sigma \) in \( S \) the new TGD \( \sigma^* \) obtained by identifying two variables in the body of \( \sigma \) belongs also to \( S \). We can assume this without loss of generality because \( \sigma^* \) is implied by \( \sigma \) and, furthermore, there is an easy procedure that iteratively adds to \( S \) all new TGDs that can be obtained by identifying two variables in the body. Also, we build a new relational schema \( S^* \) and set \( \Sigma^* \) of frontier-guarded TGDs in the following way. Initially, set \( S^* = S \) and \( \Sigma^* = \Sigma \). Then, for each \( \sigma = (\phi(\delta), \delta) \rightarrow \exists \psi(\delta, \phi) \) in \( S \), for every partition \( \chi \wedge \chi' = \phi \) of the atoms in \( \phi \), and for every CQ \( q'(\delta) = \chi(\delta, \phi) \) obtained by quantifying existentially some
variables in $\chi$, we include in $S'$ a new predicate $T_{q'}$ with the same arity than $\tilde{z}$. Furthermore, for every atom $R(\tilde{u})$ where $R \in S$ and $\tilde{u}$ contains variables occurring in $\phi$ and new fresh variables, we include in $\Sigma'$ two new TGDs $\sigma'$ and $\sigma''$ (provided they are in $\mathbb{FG}$). TGD $\sigma'$ is obtained by replacing, in $\phi$, $\chi(\tilde{u}, \tilde{z})$ by $T_q(\tilde{z}) \land R(\tilde{u})$, whereas $\sigma''$ is the TGD $\phi(\tilde{u}, \tilde{z}) \land R(\tilde{u}) \rightarrow T_{q'}(\tilde{z})$.

The following claim follows easily from the definition.

**Claim H.4.** For every S-instance $D$, and every atom $R(\tilde{c})$ where $R \in S$ and $\tilde{c}$ only mentions elements from $\text{dom}(D)$,

$$R(\tilde{c}) \in \text{chase}(D, \Sigma) \iff R(\tilde{c}) \in \text{chase}(D, \Sigma').$$

We use $\mathcal{A}$ to denote the collection of all subsets $A$ from $\text{dom}(p)$ that are guarded, i.e., there exists an atom $R(\tilde{a})$ of $p$ such that $\tilde{a}$ mentions all elements in $A$. For every $A \in \mathcal{A}$, let $E_A$ be the $S'$-database that contains all facts $R(\tilde{b}) \in \text{chase}(D[p], \Sigma')$ for which it is the case that all elements from $\tilde{b}$ appear in $A$. Notice that, by Theorem 6.7, each $E_A$ can be constructed from $D[p], \Sigma'$ and $A$. It follows also from Theorem 6.7 that we can construct a finite S'-instance $M_A = (E_A, \Sigma', s')$ with the property that, for every CQ $p''(\tilde{x}')$ with at most $s'$ variables,

$$\tilde{a} \in p''(M_A) \iff \tilde{a} \in p''(\text{chase}(E_A, \Sigma')),$$

for every tuple $\tilde{a}$ of elements in $\text{dom}(E_A)$ of the same arity as $\tilde{x}'$.

Let us define $D_A$ by removing from $M_A$ those atoms that are not over $S$. We can assume (renaming elements if necessary) that $\text{dom}(D_A) \cap \text{dom}(D) = A \cap A'$, for every $A, A' \in \mathcal{A}$. We then define $p'$ as $\cup_{A \in \mathcal{A}} D_A$. We have by construction that $D[p] \subseteq D[p']$, i.e., item (3) holds.

We now show item (6). We actually prove something stronger:

**Claim H.5.** Let $p''(\tilde{x})$ be a CQ with at most $s$ variables and $\tilde{c} \in \text{dom}(p)^{|k|}$ such that $\tilde{c} \in p''(D[p'])$. Then $\tilde{c} \in p''(\text{chase}(D[p], \Sigma)).$

**Proof.** Assume that $\tilde{c} \in p''(D[p'])$, i.e., there exists a homomorphism $h$ from $p''$ to $p'$ with $h(\tilde{x}) = \tilde{c}$. Let $p''(\tilde{x}) = \exists y \phi(x, y)$ and for every $A \in \mathcal{A}$, let $\phi_A$ be the conjunction of all atoms, $R(\tilde{z})$, in $\phi$ such that $h(\tilde{z})$ is entirely contained in $\text{dom}(D_A)$. Notice that if we define $p''(\tilde{x})$ as the CQ $\exists y \phi(x, y)$, then every $A \in \mathcal{A}$.

Since $h(\tilde{x}_A) \in p''(D_A)$, we have by definition that also $h(\tilde{x}_A) \in p''(\text{chase}(E_A, \Sigma'))$, and hence $h(\tilde{x}_A) \in p''(\text{chase}(D[p], \Sigma'))$. Since $A$ was arbitrarily chosen, we can then conclude that $h(\tilde{x}) \in p''(\text{chase}(D[p], \Sigma'))$. Consequently, since $p''$ only contains predicates in $S$ it follows directly from Claim H.4 that $h(\tilde{x}) \in p''(\text{chase}(D[p], \Sigma)).$ \hfill \QED

We then show that item (2) holds.

**Claim H.6.** It is the case that $D[p'] = \Sigma$.\hfill \QED

**Proof.** Let $\sigma = \phi'(x', y) \rightarrow \exists \psi(\tilde{x}', \tilde{w})$ any TGD in $\Sigma$ and let $h$ be any mapping of the variables in $\phi'$ to elements in $\text{dom}(p')$ such that $\phi'(h(\tilde{x}'), h(\tilde{y})) \subseteq p'$. Our goal is to show that $h(\tilde{x})$ belongs to the evaluation of $\exists \psi(\tilde{x}', \tilde{w})$ in $D[p']$. We can assume that $h$ is injective (no two variables from $\phi$ are mapped to the same element in $\text{dom}(p')$) since the non-injective case is reduced to the injective one. Indeed, if $h$ is non-injective we only need to replace $\sigma$ by the TGD obtained by identifying variables in the body that are mapped by $h$ to the same element (which by assumption also belongs to $\Sigma'$) and also modify $h$ accordingly.

Since $\Sigma \in \mathbb{FG}$, it follows that there exists some $A \in \mathcal{A}$ such that $h(\tilde{x})'$ is entirely contained in $\text{dom}(D_A)$. If $h(\tilde{y})$ is also contained in $\text{dom}(D_A)$ then there is nothing else to prove since, by construction, $D_A$ satisfies $\Sigma$.

Assume, then, that $h(\tilde{y})$ is not contained in $\text{dom}(D_A)$. Let $R(\tilde{a})$ be the atom in $p$ that guards $A$. Let $\tilde{u} = (u_1, \ldots, u_i)$ and let $u = (u_1, \ldots, u_i)$ be a tuple of variables where for every $1 \leq i \leq s$, $u_i$ is a new fresh variable if $a_i$ does not belong to the image of $h$, and $u_i = h^{-1}(a_i)$ otherwise (note that here we are using the fact that $h$ is injective).

Let $\tilde{y}'$ be a tuple containing all variables in $\tilde{y}$ that are not mapped by $h$ inside $\text{dom}(D_A)$, let $\chi(\tilde{x}', \tilde{y})$ be the conjunction of all atoms in $\phi$ containing some variable in $\tilde{y}'$ (where $\tilde{y}$ is a tuple that contains the rest of the variables in $\chi$) and let $q'(\tilde{z}) = \exists y \chi(\tilde{y}', \tilde{z})$. Note that $h(\tilde{z}) \subseteq \Sigma$ as, by construction, if a tuple in $p'$ contains elements in $\chi(\tilde{x}', h(\tilde{y})) \land R(\tilde{a}) \rightarrow q'(\tilde{z})$ is frontier-guarded by $R(\tilde{a})$ and, consequently, it belongs to $\Sigma'$. Let $h'$ be the extension of $h$ so that every fresh variable $u_i$ in $u$ is mapped by $h'$ to $a_i$. Note that the image of $h'$ is contained in $\text{dom}(p')$ and hence it follows, by applying TGD $\sigma$ with mapping $h'$, that $R_{q'}(h(\tilde{z})) \in \text{chase}(D[p], \Sigma')$. Consequently, $R_{q'}(h(\tilde{z}))$ belongs to $E_A$ and, hence, to $M_A$, as well.

Consider the TGD $\sigma''$ obtained by replacing $\chi(\tilde{y}', \tilde{z})$ by $R_{q'}(\tilde{y}) \land R(\tilde{a})$ in the body, $\phi(x', y')$, of $\sigma$. That is, $\sigma'' = \chi(\tilde{x}', \tilde{z}) \land R(\tilde{a}) \rightarrow \exists \psi(\tilde{x}', \tilde{w})$, where $\chi'$ is the set of all atoms in $\phi$ that are not in $\chi$. We shall prove that $\sigma''$ is frontier-guarded (and hence, it belongs to $\Sigma'$). Clearly $x'$ is guarded by some atom in $\chi$. In the latter case there is nothing else to prove so we can assume that $\chi$ contains some atom guarding $x'$.

Before continuing, we prove the following important properties of the homomorphisms from $p$ to $p'$.

**Claim H.7.** Let $h$ be any homomorphism from $p$ to $p'$ with $h(\tilde{x}) = \tilde{x}$. The following statements hold:

(a) It is the case that $p \equiv \Sigma p'_{(\text{dom}(h(p)))}$

(b) $h$ is injective.

(c) For every $A \in \mathcal{A}$, it is the case that $\text{dom}(h(p)) \cap \text{dom}(D_A)$ has treewidth at most $r \cdot m$.

(d) For every $Y \subseteq \text{dom}(p) \times \text{chase}(D[p], \Sigma)$ and $\tilde{x}$, the treewidth of $\text{chase}(D[p], \Sigma)$ is at most the maximum between $r \cdot m$ and the treewidth of $\text{chase}(D[p], \Sigma)$.

**Proof.** Consider first item (a). Clearly $p'_{(\text{dom}(h(p)))} \subseteq p$ since $h$ defines a homomorphism from $p$ to $p'_{(\text{dom}(h(p)))}$. For the converse, we have $\exists \psi(\tilde{x}', \tilde{w})$ and, therefore, $\exists \psi(\tilde{x}', \tilde{w})$ in $D[p']$. Since $p$, and thus $p'_{(\text{dom}(h(p)))}$ has at most has at most $n \leq s'$
where every constant holds. To finish, we show that property (4) is satisfied.

Let us prove item (c). Let \( A \in \mathcal{A} \). Since \( h(p) \) is of size at most \( n \leq s' \), it follows directly from the construction of \( D_A \) that there exists a homomorphism \( g_A \) from \((D_A)_I\) to \( \sigma \) such that \( \sigma' \) acts as the identity on \( \bigwedge \sigma \cap \text{dom}(h(p)) \) (simply take the CQ induced in \( D_A \) by \( p_h \) with all variables corresponding to elements in \( \bigwedge \sigma \cap \text{dom}(h(p)) \) being free). Let \( \mathcal{F}_A \) be the instance of \( \bigwedge \sigma \cap \text{dom}(h(p)) \) that contains only those atoms \( R(\bar{a}) \), \( R \in \mathcal{R} \), such that all the elements mentioned in \( \bar{a} \) as the identity on \( A \).

Let \( \mathcal{F}_A(\bar{x}_A) \) be a CQ defined as the conjunction of atoms in \( \mathcal{F}_A \), where every constant \( c \) is replaced by a variable \( x_c \), and \( \bar{x}_A \) is the tuple of variables that represent the elements \( a \) in \( \text{dom}(\mathcal{F}_A) \cap A \). It is then the case that \( \bar{a} \in \mathcal{F}_A(\bigwedge \sigma(\bar{x}_A), \Sigma') \). By definition of \( \mathcal{E}_A \), this means that \( \bar{a} \in \mathcal{F}_A(\bigwedge \sigma(\bar{x}_A), \Sigma') \). But \( \mathcal{F}_A \) only mentions relation symbols in \( \Sigma \), and hence it is the case that \( \bar{a} \in \mathcal{F}_A(\bigwedge \sigma(\bar{x}_A)) \) from Claim H.4. This implies our claim for the existence of \( \mathcal{F}_A \) as described above.

Consequently, the mapping \( \mathcal{F} \) that sends every element \( a \) in \( \text{dom}(p) \) to \( f_A \circ g_A \circ h(a) \), where \( A \) is any set in \( \mathcal{A} \), is well-defined and corresponds to a homomorphism from \( p \) to \( (\bigwedge \sigma, \Sigma') \). Clearly, \( (\bigwedge \sigma, \Sigma') \cap \text{dom}(f(p)) \equiv p \) and, hence, the minimality of \( p \) if it follows that \( f \) is injective.

Let \( A \in \mathcal{A} \). Since \( \bigwedge \sigma \cap \text{dom}(h(p)) \leq s \) and \( \Sigma' \in \mathcal{F}_{G_m} \), it follows directly that \( \bigwedge \sigma(\bar{x}_A), \Sigma' \), and thus \( \mathcal{F}_A \), has a treewidth at most \( r \cdot m \). Since \( f \) is injective, then \( g_A \) is also injective. Therefore, \((D_A)_I\) has treewidth at most \( r \cdot m \). This establishes item (c).

We now prove item (d). Let \( (T, x) \) be a tree decomposition of \( G^p_{\bigwedge \sigma(\bar{x}_A)} \). We shall extend it into a tree decomposition of \( G^p_{\bigwedge \sigma(\bar{x}_A)} \).

Let \( A \in \mathcal{A} \). We know from item (c) that \((\bigwedge \sigma(\bar{x}_A)) \cap \text{dom}(h(p)) \) has a treewidth at most \( r \cdot m \). This implies \( h_A := G^p_{\bigwedge \sigma(\bar{x}_A)} \) has a tree decomposition \( (T_A, \chi_A) \) of width at most \( r \cdot m \) (as \( A \) is guarded, and thus every pair of elements in \( G^p \cap A \) is linked by an edge). Moreover, since \( A \) is a guarded clique, the elements in \( A \cap \bigwedge \sigma \) define a clique on \( \chi_A \). Hence, there is a node \( v \in T_A \) such that \( A \cap \bigwedge \sigma \subseteq \chi_A(v) \). Likewise, there is \( u \in T \) such that \( A \cap \bigwedge \sigma \subseteq \chi(u) \). Thus, we can construct a new tree decomposition by joining \( u \) and \( v \) with an edge. Iterating this process for all \( A \in \mathcal{A} \), we get a tree decomposition of \( G^p_{\bigwedge \sigma(\bar{x}_A)} \), whose width is the maximum between \( r \cdot m \) and the width of \( T \). Since \( h \) is injective by item (b), \( G^p_{\bigwedge \sigma(\bar{x}_A)} \) has a tree decomposition of the same width as \( G^p_{\bigwedge \sigma(\bar{x}_A)} \).

Henceforth, for \( k \geq 1 \) we let \( K = \left( \frac{n}{3} \right) \). From the Excluded Grid Theorem [32], there is a computable function \( F : \mathbb{N} \rightarrow \mathbb{N} \) such that, for each \( k \geq 1 \) and simple graph \( G \) of treewidth at least \( F(k) \), we have that at least some connected component of \( G \) contains a \((k \times k)\)-grid as a minor. By hypothesis on the class \( \mathcal{O} \), there exists a CQS \( S = (\Sigma, \mathcal{Q}(\bar{x})) \) from \( \mathcal{O} \) such that \( S \notin (\mathcal{F}_{G_m}, \mathcal{UCQ}) \cap F(k) \), where \( F \) is as defined above. W.l.o.g., we assume that \( F(k) \geq r \cdot m \). We build from \( (G, k) \) an instance \((D^*, \Sigma, \mathcal{Q}(\bar{x})) \) of \( p\)-CQ-Evaluation\(\mathcal{O} \), where \( D^* \) is a database defined as follows.

First, observe that there must be a CQ \( q(\bar{x}) \) in \( \mathcal{Q}(\bar{x}) \) such that \( (\Sigma, \mathcal{Q}(\bar{x})) \notin (\mathcal{F}_{G_m}, \mathcal{UCQ}) \cap F(k) \), while \( q \notin \mathcal{F}_{G_m} \) for every disjunct \( q^j \) other than \( q \). In fact, if we remove from \( q \) every disjunct that is not maximal with respect to \( \leq \Sigma \), then we obtain an equivalent UCQ \( q'' \) under \( \Sigma \). Therefore, \( (\Sigma, \mathcal{Q}(\bar{x})) \notin (\mathcal{F}_{G_m}, \mathcal{UCQ}) \cap F(k) \), and hence, there is at least one disjunct \( q \) in \( q'' \) such that \( (\Sigma, \mathcal{Q}(\bar{x})) \notin (\mathcal{F}_{G_m}, \mathcal{UCQ}) \cap F(k) \).

Since, by assumption, \( F(k) \geq r \cdot m \), it is possible to compute from \( q(\bar{x}) \) and \( s \), where \( s \) is the maximum number of variables over all CQs of \( q^j \), a CQ \( p(\bar{x}) \), a subset \( X \) of the existentially quantified variables of \( p \), and a CQ \( q^j(\bar{x}) \), that satisfy the properties stated in Lemma H.3 for \( t = F(k) \). In particular, the treewidth of \( G^p_{\bigwedge \sigma(\bar{x}_A)} \) is at least \( F(k) \), and hence, by the Excluded Grid Theorem, there exists a connected component \( H \) of \( G^p_{\bigwedge \sigma(\bar{x}_A)} \) and a minor map \( q^j \) from the \((k \times k)\)-grid onto \( H \). We then define \( D^* \) as \( D^*(G, D[p], D[p], X, \mu) \), where \( D^*(G, D[p], D[p], X, \mu) \) is our modified version of Grohe’s database that satisfies the properties stated in Lemma H.2.
H.4 Correctness of the Reduction

It remains to show that the above is an fpt-reduction from p-Clique to CQS-Evaluation(ό). To this end, we need to show the following.

**Lemma H.10.** The following statements hold:

1. D* ⊆ Σ.
2. G has a k-clique [iff] x ∈ q′(D*).
3. There are computable functions f, g : N → N such that (D*, Σ, q′(x)) can be constructed in time f(k) · ||G||O(1) and (||q'|| + ||Σ||) ≤ g(k).

**Proof.** The proof of (1) follows from the last item in Theorem H.2 since D[p] ⊆ Σ and every clique of size 3 · r in G is contained in a clique of size 3 · r · m. It is worth noticing though that this holds for our variant of Grohe’s database, but not necessarily for the one that Grohe originally defined in [26].

We now proceed to show item (2).

(⇒) Assume that G has a k-clique. Then, by Lemma H.2, there is a homomorphism h from p to D* such that h₀ ∪ h is the identity on X. It follows directly from the definition of D* that the mapping h′ defined as h′(x) = h(x), when x ∈ X, and h′(x) = x, otherwise, is also a homomorphism from p to D*. In particular, since X contains only quantified variables we have that h′(x) = x, and hence x ∈ p(D*). But q ≡₌ p, and thus, x ∈ q(D*) since, by item (1), D* ⊆ Σ. Therefore, x ∈ q′(D*) as q is a disjunct of q′.

(⇐) Conversely, assume that x ∈ q′(D*). First observe that x ∈ q(D*). By contradiction, assume otherwise. Then, there is a CQ q(φ) in q′(x), other than q, such that x ∈ q′(D*). Clearly, D* homomorphically maps to p via the mapping h₀, and hence, there is a homomorphism from q to p′ mapping x to x. By Lemma H.3, x ∈ q(chase(p, Σ)), which in turn implies that q ⊆₃ q; the latter holds due to Proposition 4.5. Hence, q ⊆₃ q as p ≡₃ q, which contradicts the way in which q has been chosen.

Now, since x ∈ q(D*), we have that x ∈ p(D*) because q ≡₃ p and D* ⊆ Σ. Then, there is a homomorphism h from p to D* with h(x) = x. It follows that h₀ ∪ h is a homomorphism from p to p′ that maps x to x. In consequence, from Lemma H.3 we obtain that h(X) = X. Therefore, there must exist some m ≥ 0 such that g = h ∘ (h₀ ∘ h)m is a homomorphism from p to D* that satisfies that h₀ ∪ h is the identity on X. It follows from Lemma H.2 that G has a k-clique.

As for item (3), first notice that the CQs S = (Σ, q′) can be computed by simply enumerating the CQs from Σ until we find S since, by Theorem 5.10, we can check whether S ∉ (PG, UCQ)₃. The same holds for q. From q we can construct the CQs p and p′, as well as the set of variables X, by applying Lemma H.3. We can then compute µ via an exhaustive search over G′. Notice that the construction of q, q, p, p′, X, and µ depends only on k. Lemma H.2 states, on the other hand, that it is possible to construct D* in time polynomial, given p′, p, X, µ, and G. Putting all these together, we obtain the existence of computable functions f, g : N → N as required in item (3), and the claim follows.

I PROOF OF LEMMA 6.6

For brevity, let finans(q, D, Σ) = ∩_{M ∈ fmods(D, Σ)} q(M).

(⇒) Consider an S-database D, a set Σ ∈ C of TGDs over S, and an integer n ≥ 0. Assuming that dom(D) = {d_1, …, d_r}, for t ≥ 1, let D* = D ∪ {Dom(d_1), …, Dom(d_t)} and d = (d_i, …, d_t).

**Lemma I.1.** There exists an instance M* ∈ fmods(D*, Σ) such that, for every UCQ q* over S ∪ {Dom} of arity ℓ with at most n + ℓ variables, d ∈ q*(chase(D*, Σ)) implies d ∈ q*(chase(D*, Σ)).

**Proof.** Let q be the UCQ of arity ℓ consisting of all the CQs q* with at most n + ℓ variables such that d ∈ q*(chase(D*, Σ)). Clearly, d ∉ q(chase(D*, Σ)). Thus, there exists an instance M* ∈ fmods(D*, Σ) such that d ∉ q(M*). Fix an arbitrary UCQ q* of arity ℓ with at most n + ℓ variables. Observe that d ∉ q*(chase(D*, Σ)) implies d ∉ q*(M*) since, by construction, each CQ in q* occurs also in q, and we know that d ∉ q(M*). The claim follows.

We claim that the desired finite model M(D, Σ, n) is the instance M*₈, where M* is the (S ∪ {Dom})-instance of fmods(D*, Σ) provided by Lemma I.1. To this end, we need to show that, for every UCQ q(φ) over S of arity r ≥ 0 with at most n variables, q(chase(D*, Σ)) = q(M*₈). Fix an arbitrary tuple c ∈ dom(D)°. Clearly, c ∈ q(chase(D, Σ)) implies c ∈ q(M*₈) since M*₈ is a model of D and Σ. Assume now that c ∈ q(M*₈). We proceed to show that c ∈ q(chase(D, Σ)). It is not difficult to see that there exists a UCQ ̃q of arity ℓ over S ∪ {Dom} with at most n + ℓ variables such that ̃q ∈ q(M*₈) implies ̃q ∈ q(chase(D, Σ)).

Since ̃q has at most n + ℓ variables, by Lemma I.1, we get that

(1) ̃q ∈ q(M*₈) ⇒ ̃d ∈ q(chase(D*, Σ))

(2) ̃d ∈ q(chase(D*, Σ)) ⇒ ̃d ∈ q(chase(D, Σ))

(3) Since, by hypothesis, ̃c ∈ q(M*₈), from (1) we get that ̃d ∈ q(M*₈). Thus, by (3), we get that ̃d ∈ q(chase(D*, Σ)), and hence, by (2), we conclude that ̃c ∈ q(chase(D, Σ)), as needed.

(⇐) Consider an S-database D, a set Σ ∈ C of TGDs over S, and a UCQ q(φ) over S. Fix an arbitrary tuple c ∈ dom(D)°. We need to show that c ∈ q(chase(D, Σ)) iff c finans(q, D, Σ).

Clearly, the (⇒) direction holds trivially. It remains to show that c ∉ q(chase(D, Σ)) implies c ∉ finans(q, D, Σ). Let n ≥ 0 be the number of variables occurring in q. Since C is strongly finitely controllable, there exists M(D, Σ, n) ∈ fmods(D, Σ) such that c ∉ q(M(D, Σ, n)). This immediately implies that c ∉ finans(q, D, Σ).

J PROOF OF THEOREM 6.7

To show that PG is strongly finitely controllable, by Lemma 6.6, it suffices to show that it is finitely controllable. This can be easily shown by exploiting the fact that the guarded negation fragment of first-order logic (GNFO) [5] enjoys the finite model property, i.e., if a GNFO sentence has a model, then it has a finite one. But let us first recall the guarded negation fragment.

GNFO restricts first-order logic by requiring that all occurrences of negation are of the form α ∧ ¬ϕ, where α is an atom containing


all the free variables of $\varphi$. Formally, the formulas of GNFO are generated by the recursive definition

$$ \varphi ::= R(t_1, \ldots, t_n) \mid t_1 = t_2 \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid \exists \alpha \mid \alpha \land \neg \varphi, $$

where each $t_i$ is a term (constant or variable), and in the last clause, $\alpha$ is an atomic formula containing all free variables of $\varphi$. We know that GNFO enjoys the finite model property, and we also have an upper bound on the size of finite models:

**Proposition J.1 ([5]).** Consider a GNFO sentence $\varphi$. If $\varphi$ has a model, then it has a finite one of size $2^{2^{\aleph_0(\varphi(\sigma))}}$.

We proceed to show that $\mathbb{F}G$ is finitely controllable. Consider a database $D$, a set $\Sigma \in \mathbb{F}G$, and a UCQ $q(\bar{x})$. We need to show that

$$ q(\text{chase}(D, \Sigma)) = \bigcap_{\Sigma \in \text{fmods}(D, \Sigma)} q(M). $$

It should be clear that the $(\subseteq)$ direction holds trivially. It remains to show the $(\supseteq)$ direction. Consider a tuple $\bar{c} \in \text{dom}(D)[\bar{x}]$. Our goal is to devise a GNFO sentence $\Phi$ such that

1. $\bar{c} \in \bigcap_{\Sigma \in \text{fmods}(D, \Sigma)} q(M) \implies \Phi$ does not have a finite model.
2. $\Phi$ does not have a model $\implies \bar{c} \in q(\text{chase}(D, \Sigma))$.

Having such a sentence in place we get the $(\supseteq)$ direction since, by Proposition J.1, we can conclude that if $\Phi$ does not have a finite model, then it does not have a model at all, which in turn implies that $\bar{c} \in q(\text{chase}(D, \Sigma))$, as needed. Let us now explain how $\Phi$ is constructed. We first observe that a frontier-guarded TGD $\sigma$ of the form $\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z} \psi(\bar{x}, \bar{z})$ can be equivalently rewritten as

$$ \psi_\sigma = \neg \exists\bar{x} \exists\bar{y} (\varphi(\bar{x}, \bar{y}) \land \neg \exists\bar{z} \psi(\bar{x}, \bar{z})), $$

which is a GNFO sentence since all the free variables of $\exists\bar{z} \psi(\bar{x}, \bar{z})$ occur in the guard of $\sigma$. We also observe that the sentence $\neg q(\bar{c})$ is trivially a GNFO sentence. Therefore,

$$ \Phi = D \land \bigwedge_{\sigma \in \Sigma} \psi_\sigma \land \neg q(\bar{c}), $$

which is clearly the desired GNFO sentence.

It remains to show that a finite witness is realizable, i.e., there is a computable function that takes as input an $S$-database $D$, a set $\Sigma \in \mathbb{F}G$ over $S$, and an integer $n \geq 0$, and outputs the finite model $M(D, \Sigma, n)$. In fact, this follows from the proof of Lemma 6.6 and Proposition J.1. In the proof of Lemma 6.6, we actually show that $M(D, \Sigma, n)$ is the model of a GNFO sentence $\varphi$, and thus, due to Proposition J.1, we can assume that is of size $2^{2^{\aleph_0(\varphi(\sigma))}}$. Therefore, we can compute the finite model $M(D, \Sigma, n)$ as follows:

1. we first compute the set $S_n$ of all the non-isomorphic UCQs over $S$ with at most $n$ variables, which is clearly finite; and
2. we enumerate all the non-isomorphic finite models $M$ of $\varphi$ of size $2^{2^{\aleph_0(\varphi(\sigma))}}$, which are finitely many, until we find one such that, for every $q \in S_n$, $q(\text{chase}(D, \Sigma)) = q(M)$. Note that the latter equality can be effectively checked since $\Sigma \in \mathbb{F}G$. 
