Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Ontology-Mediated Querying on Databases of Bounded Cliquewidth

Carsten Lutz[1] , **Leif Sabellek**[2] , **Lukas Schulze**[1]
[1]Department of Computer Science, Leipzig University, Germany
[2]Department of Computer Science, University of Bremen, Germany
{clu, lschulze}@informatik.uni-leipzig.de, sabellek@uni-bremen.de

Abstract

We study the evaluation of ontology-mediated queries (OMQs) on databases of bounded cliquewidth from the viewpoint of parameterized complexity theory. As the ontology language, we consider the description logics $\mathcal{ALC}$ and $\mathcal{ALCI}$ as well as the guarded two-variable fragment GF_2 of first-order logic. Queries are atomic queries (AQs), conjunctive queries (CQs), and unions of CQs. All studied OMQ problems are fixed-parameter linear (FPL) when the parameter is the size of the OMQ plus the cliquewidth. Our main contribution is a detailed analysis of the dependence of the running time on the parameter, exhibiting several interesting effects.

1 Introduction

Ontology-mediated querying is an established subfield of knowledge representation. The general aim is to enrich a database with an ontology to add domain knowledge and to extend the vocabulary available for query formulation (Bienvenu et al. 2014; Bienvenu and Ortiz 2015; Calvanese et al. 2009). Expressive description logics (DLs) such as $\mathcal{ALC}$ and $\mathcal{ALCI}$ are a popular choice for the ontology language as they underpin the widely used OWL DL profile of the OWL 2 ontology language. However, ontology-mediated querying with these languages is coNP-complete in data complexity even for simple atomic queries (AQs) of the form $A(x)$ and thus scales poorly to larger amounts of data. One reaction to this problem is to resort to approximate query answers, as done for example in (Zhou et al. 2015; Haga et al. 2021). A potential alternative is to only admit databases from a class that is sufficiently restricted so that non-approximate answers can be computed in PTIME, or to decompose the input database into component databases from such a class.

The aim of this paper is to study ontology-mediated querying on classes of databases of bounded cliquewidth. This is relevant because such classes are maximal known ones on which ontology-mediated querying with DLs such as $\mathcal{ALC}$ and $\mathcal{ALCI}$ is in PTIME in data complexity, even when unions of conjunctive queries (UCQs) are admitted as queries. This follows from a folklore translation of ontology-mediated querying into MSO_1 and the known result that MSO_1 has PTIME data complexity on structures of bounded cliquewidth (Courcelle, Makowsky, and Rotics 2000). In fact, it even has linear time data complexity (when a k-expression for constructing the database is provided). We recall that bounded treewidth implies bounded cliquewidth since every database of treewidth k has cliquewidth at most $3 \cdot 2^{k-1}$ (Corneil and Rotics 2005). The converse is false because databases of bounded treewidth are sparse in the sense of graph theory, whereas databases of bounded cliquewidth may be dense.

We do not claim that databases encountered in practical applications typically have small cliquewidth. In fact, an empirical study of real-world databases has found that their treewidth tends to be high (Maniu, Senellart, and Jog 2019), and one may expect similar results for cliquewidth. The same study, however, also points out the opportunities that lie in the decomposition of a database into a part of high treewidth and parts of low treewidth, and this has in fact been used successfully to implement efficient querying (Wei 2010; Akiba, Sommer, and Kawarabayashi 2012; Maniu, Cheng, and Senellart 2017). While we are not aware that similar approaches based on cliquewidth have yet been studied, it seems entirely reasonable to pursue them. The work presented here may provide a foundation for such an endeavour.

In this paper, we focus on the framework of parameterized complexity theory. Linear time data complexity with a uniform algorithm trivially implies *fixed-parameter linearity (FPL)* when the parameter is the size of the ontology-mediated query (OMQ). Recall that FPL is defined like the more familiar *fixed-parameter tractability (FPT)* except that the running time may depend only linearly on the size of the database, that is, it must be $f(|Q|) \cdot |\mathcal{D}|$ where f is a computable function, Q the OMQ, and $\mathcal{D}$ the database. Our main goal is to determine the optimal running time of FPL algorithms for ontology-mediated querying on databases of bounded cliquewidth and to provide algorithms that achieve this running time.

As the ontology language, we consider the DLs $\mathcal{ALC}$ and $\mathcal{ALCI}$ and the more expressive guarded two-variable fragment GF_2 of first-order logic. As queries, we admit AQs, conjunctive queries (CQs), and UCQs. An *ontology-mediated query (OMQ)* combines an ontology and a query, and an *OMQ language* is determined by the choice of an ontology language and a query language. For instance, $(\mathcal{ALC}, \mathrm{AQ})$ is the OMQ language that combines $\mathcal{ALC}$ ontologies with AQs. We find that the running time may be single or double exponential in $|Q|$, depending on the choice of the OMQ language. There are several surprising effects, which we highlight in the following. An overview is provided in Table 1.

We show that in $(\mathcal{ALCI}, \text{AQ})$, evaluating an OMQ Q on a database $\mathcal{D}$ is possible in time $2^{O(|Q|\cdot k)} \cdot |\mathcal{D}|$ where k is the cliquewidth of $\mathcal{D}$.[1] We assume here that a k-expression for constructing $\mathcal{D}$ is given (otherwise, the running time is cubic in $|\mathcal{D}|$ and thus FPT, but not FPL). Note that this implies FPL even when the parameter is the size of the OMQ plus the cliquewidth of the database (and so the cliquewidth of the database needs not be bounded by a constant). In (GF_2, AQ), in contrast, we achieve a running time of $2^{2^{O(|Q|)}\cdot k^2} \cdot |\mathcal{D}|$ and show that attaining $2^{2^{o(|Q|)}} \cdot \mathtt{poly}(|\mathcal{D}|)$ is impossible even on databases of cliquewidth 2 unless the exponential time hypothesis (ETH) fails. This is interesting for several reasons. First, it is folklore that OMQ evaluation in (GF_2, AQ) is ExpTime-complete in combined complexity on unrestricted databases, and thus one has to 'pay' for the running time to be polynomial in $|\mathcal{D}|$ on databases of bounded cliquewidth by an exponential increase in the running time in $|Q|$. Second, the higher running time for (GF_2, AQ) compared to $(\mathcal{ALCI}, \text{AQ})$ is *not* due to the fact that GF_2 is more expressive than $\mathcal{ALCI}$. In fact, the GF_2 ontology used in the lower bound proof can be expressed in $\mathcal{ALC}$ and the increase in complexity is due to the higher succinctness of GF_2. And third, we also show that in (GF_2, AQ), OMQs can be evaluated in time $2^{O(|Q|\cdot k^2)} \cdot |\mathcal{D}|$ where k is the *treewidth* of $\mathcal{D}$. Thus, the exponential difference in complexity between $(\mathcal{ALCI}, \text{AQ})$ and (GF_2, AQ) that we observe for cliquewidth does not exist for treewidth.

The above results concern AQs, and transitioning to (U)CQs brings about some interesting differences. We show that in $(\mathcal{ALCI}, \text{UCQ})$, evaluation of an OMQ Q is possible in time $2^{|\mathcal{O}|\cdot k^{O(|q|\log(|q|))}} \cdot |\mathcal{D}|$ where $\mathcal{O}$ is the ontology in Q and q is the query in Q. This is complemented by the result that already in $(\mathcal{ALC}, \text{CQ})$, a running time of $2^{2^{o(|Q|)}} \cdot \mathtt{poly}(|\mathcal{D}|)$ cannot be achieved even on databases of cliquewidth 3 unless ETH fails. This should be contrasted with the fact that OMQ evaluation in $(\mathcal{ALC}, \text{CQ})$ and $(\mathcal{ALC}, \text{UCQ})$ is ExpTime-complete in combined complexity on unrestricted databases (Lutz 2008) and thus also in these OMQ languages one has to pay for polynomial running time in $|\mathcal{D}|$ by an exponential increase in overall running time. Note however, that the increase is only in $|q|$, but not in $|\mathcal{O}|$. This is in contrast to the case of AQs where the size of queries is actually constant, and it is good news as queries tend to be much smaller than ontologies.

Finally we observe that, in $(\mathcal{ALCF}, \text{AQ})$, where $\mathcal{ALCF}$ is the extension of $\mathcal{ALC}$ with functional roles, OMQ evaluation on databases of treewidth 2 is coNP-hard in data complexity when the unique name assumption is not made.

Proof details are delegated to the appendix, see (Lutz, Sabellek, and Schulze 2022).

Related Work. Monadic second-order logic (MSO) comes in two versions: MSO_1 admits quantification only over sets of nodes while MSO_2 can additionally quantify over sets of edges. Courcelle's theorem states that MSO_2 model checking is FPL on graphs of bounded treewidth and (Courcelle, Makowsky, and Rotics 2000) shows the same for MSO_1 and graphs of bounded cliquewidth (when a k-expression is provided). The problem studied in the current paper can be translated into MSO_1, but this does not yield the tight complexities presented here. It was shown in (Kreutzer and Tazari 2010) that MSO_2 is not FPT for graph classes that have unbounded treewidth and are closed under substructures (under certain assumptions), see also (Ganian et al. 2014); results in this style do not appear to be known for MSO_1 and cliquewidth.

[1] We explain our use of the O-notation in the appendix.

	Bnd. CW	Bnd. TW	Unrestr.
$(\mathcal{ALC}(\mathcal{I}), \text{AQ})$	sngl exp	sngl exp	ExpTime
(GF_2, AQ)	dbl exp	sngl exp	ExpTime
$(\mathcal{ALC}, \text{(U)CQ})$	dbl exp	sngl exp	ExpTime
$(\mathcal{ALCI}, \text{(U)CQ})$	dbl exp	dbl exp	2ExpTime
$(\text{GF}_2, \text{(U)CQ})$	open	dbl exp	2ExpTime

Table 1: Overview of results

2 Preliminaries

Description Logics. Let $\mathsf{N_C}$, $\mathsf{N_R}$, and C be countably infinite sets of *concept names*, *role names*, and *constants*. A *role* is a role name r or an *inverse role* r^-, where r is a role name. Set $(r^-)^- = r$. *$\mathcal{ALCI}$-concepts* are generated by the rule

$$C, D ::= A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C$$

where A ranges over concept names and r over roles. An *$\mathcal{ALC}$-concept* is an $\mathcal{ALCI}$-concept that does not use inverse roles. We use $\top$ to denote $A \sqcup \neg A$ for some fixed concept name A and $\bot$ for $\neg\top$. For $\mathcal{L} \in \{\mathcal{ALC}, \mathcal{ALCI}\}$, an *$\mathcal{L}$-ontology* is a finite set of *concept inclusions (CIs)* $C \sqsubseteq D$ with C and D $\mathcal{L}$-concepts. A *database* is a finite set of *facts* of the form $A(c)$ or $r(c, c')$ where $A \in \mathsf{N_C} \cup \{\top\}$, $r \in \mathsf{N_R}$, and $c, c' \in \mathsf{C}$. We use $\mathtt{adom}(\mathcal{D})$ to denote the set of constants used in a database $\mathcal{D}$, also called its *active domain*. The *size* of a syntactic object X, denoted $|X|$, is the number of symbols needed to write X as a word over a finite alphabet using a suitable encoding.

The semantics is given in terms of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, we refer to (Baader et al. 2017) for details. An interpretation $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, a fact $A(c)$ if $c \in A^{\mathcal{I}}$, and a fact $r(c, c')$ if $(c, c') \in r^{\mathcal{I}}$. We thus make the standard names assumption, that is, we interpret constants as themselves. This implies the unique name assumption. For $S \subseteq \Delta^{\mathcal{I}}$, we use $\mathcal{I}|_S$ to denote the restriction of $\mathcal{I}$ to domain S. An interpretation $\mathcal{I}$ is a *model* of an ontology or database if it satisfies all inclusions or facts in it. A database $\mathcal{D}$ is *satisfiable* w.r.t. an ontology $\mathcal{O}$ if there is a model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{D}$.

We also consider the *guarded two-variable fragment (GF_2)* of first-order logic. In GF_2, only two fixed variables x and y are available and quantification is restricted to the pattern

$$\forall \bar{y}(\alpha(\bar{x}, \bar{y}) \rightarrow \varphi(\bar{x}, \bar{y})) \quad \exists \bar{y}(\alpha(\bar{x}, \bar{y}) \wedge \varphi(\bar{x}, \bar{y}))$$

where $\varphi(\bar{x}, \bar{y})$ is a GF_2 formula with free variables among $\bar{x} \cup \bar{y}$ and $\alpha(\bar{x}, \bar{y})$ is an atomic formula (possibly an equality) called the *guard* that uses all variables in $\bar{x} \cup \bar{y}$. Function symbols and constants are not admitted, and neither is equal-

ity in non-guard positions. We only admit relation symbols of arity one and two, identifying the former with concept names and the latter with role names. We may thus interpret GF_2 formulas in DL interpretations. A *GF_2-ontology* is a finite set of GF_2-sentences.

Queries. A *conjunctive query (CQ)* is of the form $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$, where $\bar{x}$ and $\bar{y}$ are tuples of variables and $\varphi(\bar{x}, \bar{y})$ is a conjunction of *concept atoms* $A(x)$ and *role atoms* $r(x, y)$, $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and x, y variables from $\bar{x} \cup \bar{y}$. We call the variables in $\bar{x}$ the *answer variables* of q, and use $\mathtt{var}(q)$ to denote $\bar{x} \cup \bar{y}$. We may write $\alpha \in q$ to indicate that α is an atom in q. For $V \subseteq \mathtt{var}(q)$, we use $q|_V$ to denote the restriction of q to the atoms that use only variables in V. A tuple $\bar{d} \in (\Delta^{\mathcal{I}})^{|\bar{x}|}$ is an *answer* to q on an interpretation $\mathcal{I}$ if there is a homomorphism h from q to $\mathcal{I}$ with $h(\bar{x}) = \bar{d}$. More details are in the appendix.

A *union of conjunctive queries (UCQ)* $q(\bar{x})$ is a disjunction of CQs with the same answer variables $\bar{x}$. A tuple $\bar{d} \in (\Delta^{\mathcal{I}})^{|\bar{x}|}$ is an *answer* to q on interpretation $\mathcal{I}$, written $\mathcal{I} \models q(\bar{d})$, if $\bar{d}$ is an answer to some CQ in q on $\mathcal{I}$. We use $q(\mathcal{I})$ to denote set of all answers to q on $\mathcal{I}$. The *arity* of q is the length of $\bar{x}$ and q is *Boolean* if it is of arity zero. An *atomic query (AQ)* is a CQ of the form $A(x)$ with A a concept name.

Ontology-Mediated Querying. An *ontology-mediated query (OMQ)* is a pair $Q = (\mathcal{O}, q)$ with $\mathcal{O}$ an ontology and q a query such as a UCQ. While OMQs are often defined to include an additional third component, the data signature, the problem of query evaluation studied in this paper is insensitive to that component, and so we omit it. We write $Q(\bar{x})$ to indicate that the answer variables of q are $\bar{x}$. A tuple $\bar{a} \in \mathtt{adom}(\mathcal{D})^{|\bar{x}|}$ is an *answer* to $Q(\bar{x})$ on a database $\mathcal{D}$, written $\mathcal{D} \models Q(\bar{a})$, if $\mathcal{I} \models q(\bar{a})$ for all models $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{D}$. When more convenient, we might alternatively write $\mathcal{D}, \mathcal{O} \models q(\bar{a})$. We write $Q(\mathcal{D})$ to denote the set of all answers to Q on $\mathcal{D}$. We use $(\mathcal{L}, \mathcal{Q})$ to denote the *OMQ language* that contains all OMQs Q in which $\mathcal{O}$ is formulated in the DL $\mathcal{L}$ and q in the query language $\mathcal{Q}$, such as in $(\mathcal{ALCI}, \mathrm{UCQ})$ and $(\mathrm{GF}_2, \mathrm{AQ})$.

Let $(\mathcal{L}, \mathcal{Q})$ be an OMQ language. We write *OMQ evaluation in* $(\mathcal{L}, \mathcal{Q})$ to denote the problem to decide, given an OMQ $Q(\bar{x}) \in (\mathcal{L}, \mathcal{Q})$, a database $\mathcal{D}$, and a tuple $\bar{c} \in \mathtt{adom}(\mathcal{D})^{|\bar{x}|}$, whether $\bar{c} \in Q(\mathcal{D})$. A *parameterization* is a function κ that assigns to every input $Q, \mathcal{D}, \bar{c}$ a parameter $\kappa(Q, \mathcal{D}, \bar{c}) \in \mathbb{N}$. We will use $\kappa = |Q|$ in lower bounds and $\kappa = |Q| + k$ in upper bounds where k is the cliquewidth of $\mathcal{D}$ as defined below. OMQ evaluation is *fixed-parameter tractable (FPT) for* κ if it can be decided by an algorithm that runs in time $f(\kappa(Q, \mathcal{D}, \bar{c})) \cdot \mathtt{poly}(n)$ where f is a computable function, here and throughout the paper, $\mathtt{poly}$ denotes an unspecified polynomial, and n is the size of the input. *Fixed-parameter linearity (FPL) for* κ is defined analogously, with running time $f(\kappa(Q, \mathcal{D}, \bar{c})) \cdot O(n)$.

Treewidth and Cliquewidth. Treewidth describes how similar a graph (in our case: a database) is to a tree, and cliquewidth does the same for cliques (Courcelle and Engelfriet 2012). Due to space restrictions and since this article focuses on cliquewidth, we provide the definition of treewidth only in the appendix. Databases of cliquewidth k can be constructed by a k-expression as defined below, where k refers to the number of *labels* in the expression. For convenience, we use a reservoir of fresh concept names $L_1, L_2, \ldots$ as labels that may occur in databases, but not in ontologies and queries. We say that a constant $c \in \mathtt{adom}(\mathcal{D})$ is *labeled* i if $L_i(c) \in \mathcal{D}$. An *expression* s defines a database $\mathcal{D}_s$ built from four operations, as follows:

$\boldsymbol{s = \iota(\mathcal{D})}$ is the nullary operator for *constant introduction* where $i \geq 1$ and $\mathcal{D}$ is a database with $|\mathtt{adom}(\mathcal{D})| = 1$ that contains exactly one fact of the form $L_i(c)$, and $\mathcal{D}_s = \mathcal{D}$.

$\boldsymbol{s = s_1 \oplus s_2}$ is the binary operator for *disjoint union* where s_1 and s_2 are expressions such that $\mathtt{adom}(\mathcal{D}_{s_1}) \cap \mathtt{adom}(\mathcal{D}_{s_2}) = \emptyset$ and $\mathcal{D}_s = \mathcal{D}_{s_1} \cup \mathcal{D}_{s_2}$.

$\boldsymbol{s = \alpha^r_{i,j}(s')}$ is the unary operator for *role insertion* where r is a role name, $i, j \geq 1$ are distinct, and s' is an expression. It links all constants labeled i to all constants labeled j using role r, that is,

$$\mathcal{D}_s = \mathcal{D}_{s'} \cup \{r(a, b) \mid L_i(a) \in \mathcal{D}_{s'} \text{ and } L_j(b) \in \mathcal{D}_{s'}\}.$$

$\boldsymbol{s = \rho_{i \to j}(s')}$ is the unary operator for *relabeling* where $i, j \geq 1$ are distinct and s' is an expression. It changes all i-labels to j, that is,

$$\mathcal{D}_s = \mathcal{D}_{s'} \setminus \{L_i(c) \mid c \in \mathtt{adom}(\mathcal{D}_{s'})\} \cup \{L_j(c) \mid L_i(c) \in \mathcal{D}_{s'}\}.$$

An expression s is a *k-expression* if all labels in s are from $\{L_1, \ldots, L_k\}$. A database $\mathcal{D}$ has *cliquewidth* $k > 0$ if there is a k-expression s with $\mathcal{D}_s = \mathcal{D}$, but no k'-expression s' such that $k' < k$ and $\mathcal{D}_{s'} = \mathcal{D}$, up to facts of the form $L_i(c)$.

Example 1. *Consider the class of databases about schools that use the concept names* $\mathtt{Pupil}$, $\mathtt{Teacher}$ *and* $\mathtt{School}$ *as well as role names* $\mathtt{teaches}$, $\mathtt{worksAt}$ *and* $\mathtt{isClassmateOf}$, *and where each teacher works at exactly one school and can teach any number of pupils. The set of all pupils is partitioned into groups of pupils that are classmates of one another, and each teacher teaches exactly one such group. A basic database from that class is*

$$\begin{aligned}\mathcal{D} = \{&\mathtt{Pupil}(a_1), \mathtt{Pupil}(a_2), \mathtt{Teacher}(b), \mathtt{School}(c),\\ &\mathtt{worksAt}(b, c), \mathtt{teaches}(b, a_1), \mathtt{teaches}(b, a_2),\\ &\mathtt{isClassmateOf}(a_1, a_2), \mathtt{isClassmateOf}(a_2, a_1)\}.\end{aligned}$$

This database has cliquewidth 3*, and in fact so does any database from the described class, independently of the number of pupils, teachers, and schools. When constructing k-expressions, it suffices to use up to two labels for pupils, one for teachers, and one for schools. With proper relabeling, no more than three labels are needed at the same time.*

3 Upper Bounds and Algorithms

We prove that OMQ evaluation on databases of bounded cliquewidth is FPL when the parameter is the size of the OMQ plus the cliquewidth of the database and establish upper bounds on the overall running time.

3.1 $\mathcal{ALCI}$ with AQs

Theorem 1. *In $(\mathcal{ALCI}, \mathrm{AQ})$, an OMQ $Q = (\mathcal{O}, q)$ can be evaluated on a database $\mathcal{D}$ of cliquewidth k in time $2^{O(|\mathcal{O}| \cdot k)} \cdot |\mathcal{D}|$.*

Theorem 1 implies that OMQ evaluation in $(\mathcal{ALCI}, \text{AQ})$ is in FPL, and also that it is in linear time in data complexity when the cliquewidth of databases is bounded by a constant. As announced before, we assume here and in all subsequent upper complexity bounds that a k-expression s_0 for $\mathcal{D}$ is given as part of the input. In fact, the number of subexpressions of s_0 must be $O(|\mathcal{D}|)$, a condition that is satisfied by any reasonable k-expression for $\mathcal{D}$. These assumptions could clearly be dropped if, given a database $\mathcal{D}$ of cliquewidth k, we could compute a k-expression that generates $\mathcal{D}$ (or a sufficient approximation thereof) in time $2^{O(k)} \cdot |\mathcal{D}|$. It is an open problem whether this is possible, see e.g. (Downey and Fellows 2013). There is, however, an algorithm that computes, given a database $\mathcal{D}$ of cliquewidth k, in time $|\mathtt{adom}(\mathcal{D})|^3$ a k'-expression with $k' \leq 2^{3k-1}$ (Oum 2008). As a consequence, we obtain an analogue of Theorem 1 that does not require a k-expression for $\mathcal{D}$ to be given and where the time bound is replaced with $2^{|Q| \cdot 2^{O(k)}} \cdot |\mathcal{D}|^3$. While this no longer yields FPL, it still yields FPT.

To prove Theorem 1, it suffices to give an algorithm for database satisfiability w.r.t. an $\mathcal{ALCI}$ ontology that runs within the stated time bounds. To decide whether $\mathcal{D}, \mathcal{O} \models A(c)$, we may then simply check whether $\mathcal{D} \cup \{\overline{A}(c)\}$ is unsatisfiable w.r.t. the ontology $\mathcal{O} \cup \{\overline{A} \equiv \neg A\}$.

Assume that we are given as input a database $\mathcal{D}_0$, an ontology $\mathcal{O}$, and a k-expression s_0 that generates $\mathcal{D}_0$. We may assume w.l.o.g. that $\mathcal{O}$ takes the form $\{\top \sqsubseteq C_{\mathcal{O}}\}$ where $C_{\mathcal{O}}$ is an $\mathcal{ALCI}$-concept in negation normal form (NNF), that is, negation is only applied to concept names, but not to compound concepts. Every $\mathcal{ALCI}$-ontology $\mathcal{O}$ can be converted into this form in linear time (Baader et al. 2017).

Our algorithm traverses the k-expression s_0 bottom-up, computing for each subexpression s a succinct representation of the models of the database $\mathcal{D}_s$ and ontology $\mathcal{O}$. Before giving details, we introduce some relevant notions. We use $\mathtt{sub}(\mathcal{O})$ to denote the set of all concepts in $\mathcal{O}$, closed under subconcepts and $\mathtt{cl}(\mathcal{O})$ for the extension of $\mathtt{sub}(\mathcal{O})$ with the NNF of all concepts $\neg C$, $C \in \mathtt{sub}(\mathcal{O})$. Moreover, $\mathtt{cl}^\forall(\mathcal{O})$ denotes the restriction of $\mathtt{cl}(\mathcal{O})$ to concepts of the form $\forall r.C$ and $\mathtt{cl}^*(\mathcal{O})$ is $\{C \mid \forall r.C \in \mathtt{cl}(\mathcal{O})\}$. A *type for* $\mathcal{O}$ is a set $t \subseteq \mathtt{cl}(\mathcal{O})$ such that for some model $\mathcal{I}$ of $\mathcal{O}$ and some $d \in \Delta^{\mathcal{I}}$, $t = \{C \in \mathtt{cl}(\mathcal{O}) \mid d \in C^{\mathcal{I}}\}$. We then also denote t with $\mathtt{tp}_{\mathcal{I}}(d)$.

An *input output assignment (IOA)* for $\mathcal{O}$ is a pair $\gamma = (\mathtt{in}, \mathtt{out})$ with $\mathtt{in} : \{1, \dots, k\} \to 2^{\mathtt{cl}^*(\mathcal{O})}$ and $\mathtt{out} : \{1, \dots, k\} \to 2^{\mathtt{cl}^\forall(\mathcal{O})}$ total functions. For easier reference, we use $\gamma^{\mathtt{in}}$ to denote $\mathtt{in}$ and $\gamma^{\mathtt{out}}$ to denote $\mathtt{out}$. Every database $\mathcal{D}$ and model $\mathcal{I}$ of $\mathcal{D}$ and $\mathcal{O}$ give rise to an IOA $\gamma_{\mathcal{I},\mathcal{D}}$ for $\mathcal{O}$ defined by setting, for $1 \leq i \leq k$,

$$\gamma^{\mathtt{in}}_{\mathcal{I},\mathcal{D}}(i) = \mathtt{cl}^*(\mathcal{O}) \cap \bigcap_{L_i(c) \in \mathcal{D}} \mathtt{tp}_{\mathcal{I}}(c) \text{ and}$$

$$\gamma^{\mathtt{out}}_{\mathcal{I},\mathcal{D}}(i) = \mathtt{cl}^\forall(\mathcal{O}) \cap \bigcup_{L_i(c) \in \mathcal{D}} \mathtt{tp}_{\mathcal{I}}(c).$$

Intuitively, $\gamma^{\mathtt{out}}(i)$ lists outputs generated by label i in the sense that every concept $\forall r.C \in \gamma^{\mathtt{out}}(i)$ 'outputs' C to all constants labeled j when we use the $\alpha^r_{i,j}$ operation to introduce new role edges. It is then important that $C \in \gamma^{\mathtt{in}}(j)$, and in this sense $\gamma^{\mathtt{in}}(j)$ lists inputs accepted by label j.

The central idea of our algorithm is to compute, for each subexpression s of s_0, the set of IOAs

$$\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}_s} \mid \mathcal{I} \text{ model of } \mathcal{D}_s \text{ and } \mathcal{O}\}.$$

It then remains to check whether $\Theta(s_0)$ is non-empty. The sets $\Theta(s)$ are computed as follows:

$\boldsymbol{s = \iota(\mathcal{D})}$: Set $\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}} \mid \mathcal{I} \text{ model of } \mathcal{D} \text{ and } \mathcal{O}\}$.

$\boldsymbol{s = s_1 \oplus s_2}$: $\Theta(s)$ contains an IOA γ for each pair of IOAs $(\gamma_1, \gamma_2) \in \Theta(s_1) \times \Theta(s_2)$ defined by setting, for $1 \leq i \leq k$:

$$\gamma^{\mathtt{in}}(i) = \gamma_1^{\mathtt{in}}(i) \cap \gamma_2^{\mathtt{in}}(i) \quad \gamma^{\mathtt{out}}(i) = \gamma_1^{\mathtt{out}}(i) \cup \gamma_2^{\mathtt{out}}(i).$$

$\boldsymbol{s = \alpha^r_{i,j}(s')}$: $\Theta(s)$ is obtained from $\Theta(s')$ by removing IOAs that are ruled out by the additional role links. Formally, we keep in $\Theta(s)$ only those IOAs γ such that

a) if $\forall r.C \in \gamma^{\mathtt{out}}(i)$ then $C \in \gamma^{\mathtt{in}}(j)$ and
b) if $\forall r^-.C \in \gamma^{\mathtt{out}}(j)$ then $C \in \gamma^{\mathtt{in}}(i)$.

$\boldsymbol{s = \rho_{i \to j}(s')}$: for each $\widehat{\gamma} \in \Theta(s')$, $\Theta(s)$ contains the IOA γ defined as follows:

c) $\gamma^{\mathtt{in}}(i) = \mathtt{cl}^*(\mathcal{O})$ and $\gamma^{\mathtt{out}}(i) = \emptyset$
d) $\gamma^{\mathtt{in}}(j) = \widehat{\gamma}^{\mathtt{in}}(i) \cap \widehat{\gamma}^{\mathtt{in}}(j)$ and $\gamma^{\mathtt{out}}(j) = \widehat{\gamma}^{\mathtt{out}}(i) \cup \widehat{\gamma}^{\mathtt{out}}(j)$
e) $\gamma^{\mathtt{in}}(\ell) = \widehat{\gamma}^{\mathtt{in}}(\ell)$ and $\gamma^{\mathtt{out}}(\ell) = \widehat{\gamma}^{\mathtt{out}}(\ell)$ for all $\ell \in \{1, \dots, k\} \setminus \{i, j\}$.

We prove in the appendix that the algorithm is correct.

Lemma 1. *For all subexpressions s of s_0, $\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}_s} \mid \mathcal{I}$ model of $\mathcal{D}_s$ and $\mathcal{O}\}$.*

The presented algorithm achieves the intended running time. We do a single bottom-up pass over s_0, considering each subexpression s, of which there are at most $O(|\mathcal{D}|)$ many. The size of each IOA in $\Theta(s)$ is $O(|\mathcal{O}| \cdot k)$ and there are at most $2^{O(|\mathcal{O}| \cdot k)}$ IOAs. It can be verified that each set $\Theta(s)$ can be constructed in time $2^{O(|\mathcal{O}| \cdot k)}$. For the case $s = \iota(\mathcal{D})$, this essentially amounts to enumerating all types for $\mathcal{O}$, more details are in the appendix.

3.2 $\mathcal{ALCI}$ with UCQs

Theorem 2. *In $(\mathcal{ALCI}, \text{UCQ})$, an OMQ $Q = (\mathcal{O}, q)$ can be evaluated on a database $\mathcal{D}$ of cliquewidth k in time $2^{|\mathcal{O}| \cdot k^{O(|q| \log(|q|))}} \cdot |\mathcal{D}|$.*

The algorithm used to prove Theorem 2 is a generalization of the one used for Theorem 1. It also traverses the k-expression s_0 bottom-up, computing for each subexpression s a set of IOAs that additionally are annotated with information about partial homomorphisms from CQs in the UCQ q into models of $\mathcal{D}_s$ and $\mathcal{O}$. The generalization is not entirely straightforward. First, the homomorphism may map some variables to elements of the model that are 'outside' of the database $\mathcal{D}_s$ while IOAs only store information about constants from $\mathcal{D}_s$. And second, we cannot even memorize all homomorphisms that map all variables into $\mathcal{D}_s$ as there may be $|\mathcal{D}_s|^{|q|}$ many of them, too many for FPL. Our solution to the first issue is to modify the ontology and UCQ

so that variables need never be mapped outside of $\mathcal{D}$. We resolve the second issue by the observation that it suffices to memorize the *labels* of the constants in $\mathcal{D}_s$ to which variables are mapped, but not their precise identity. The generalized algorithm establishes Theorem 1 as a special case; we singled out the algorithm in the previous section for didactic reasons.

Assume that we are given as input an OMQ $Q(\bar{x}) = (\mathcal{O}, q) \in (\mathcal{ALCI}, \text{UCQ})$, a database $\mathcal{D}_0$, and a k-expression s_0 that generates $\mathcal{D}_0$. It is easy to see that we may assume w.l.o.g. that Q is Boolean, details are in the appendix. We may also assume the ontology $\mathcal{O}$ to contain a CI $\top \sqsubseteq A_\top$, with $A_\top$ not used anywhere else in $\mathcal{O}$, and that every CQ p in q contains the concept atom $A_\top(x)$ for every $x \in \mathtt{var}(p)$. This will prove useful for the constructions below.

An interpretation $\mathcal{I}$ is a *tree* if the undirected graph $G_\mathcal{I} = (V, E)$ with $V = \Delta^\mathcal{I}$ and $E = \{\{x, y\} \mid (x, y) \in r^\mathcal{I}$ for some $r\}$ is a tree and there are no self loops and multi-edges, the latter meaning that $(d, e) \in r_1^\mathcal{I}$ implies $(d, e) \notin r_2^\mathcal{I}$ for all distinct roles r_1, r_2. We call a model $\mathcal{I}$ of $\mathcal{D}_0$ *tree-extended* if it satisfies the following conditions:

- $r^\mathcal{I} \cap (\mathtt{adom}(\mathcal{D}_0) \times \mathtt{adom}(\mathcal{D}_0)) = \{(d, e) \mid r(d, e) \in \mathcal{D}_0\}$ for all role names r and
- if $\mathcal{I}$ is modified by setting $r^\mathcal{I} = r^\mathcal{I} \setminus (\mathtt{adom}(\mathcal{D}_0) \times \mathtt{adom}(\mathcal{D}_0))$ for all role names r, then the result is a disjoint union of trees and each of these trees contains exactly one constant from $\mathtt{adom}(\mathcal{D}_0)$.

The following is well-known, see for example (Lutz 2008).

Lemma 2. *If there is a model $\mathcal{I}$ of $\mathcal{D}_0$ and $\mathcal{O}$ such that $\mathcal{I} \not\models q$, then there is such a model $\mathcal{I}$ that is tree-extended.*

Lemma 2 provides the basis for the announced modification of $\mathcal{O}$ and the CQs in q: we extend $\mathcal{O}$ to an ontology $\mathcal{O}_q$ and q into a UCQ $\widehat{q}$ such that for every tree-extended model $\mathcal{I}$ of $\mathcal{D}_0$ and $\mathcal{O}_q$, $\mathcal{I} \models q$ iff $\mathcal{I}|_{\mathtt{adom}(\mathcal{D}_0)} \models \widehat{q}$. We then proceed to work with $\mathcal{O}_q$ and $\widehat{q}$ in place of $\mathcal{O}$ and q, which allows us to concentrate on homomorphisms from CQs in $\widehat{q}$ to models $\mathcal{I}$ of $\mathcal{D}$ and $\mathcal{O}_q$ that map all variables to $\mathtt{adom}(\mathcal{D}_0)$.

We next present the extension of $\mathcal{O}$. A Boolean or unary CQ p being a *tree* (without self-loops and multi-edges) is defined in the expected way. Note that a unary tree CQ p can be viewed as an $\mathcal{ALCI}$-concept C_p in an obvious way. For example, if $p(x) = \{A(x), r(y, x), s(y, z), B(z), r(y, z'), A(z')\}$, then $C_p = A \sqcap \exists r^-.(\exists s.B \sqcap \exists r.A)$. We use $\mathtt{trees}(q)$ to denote the set of all (Boolean) tree CQs that can be obtained from a CQ in q by first dropping atoms and then taking a contraction. When writing $p(x)$ with $p \in \mathtt{trees}(q)$ and $x \in \mathtt{var}(p)$, we mean p with x made answer variable. The ontology $\mathcal{O}_q$ is obtained from $\mathcal{O}$ by adding the following, and then converting to NNF:

- for every $p \in \mathtt{trees}(q)$ and $x \in \mathtt{var}(p)$, the CIs $A_{p(x)} \sqsubseteq C_{p(x)}$ and $C_{p(x)} \sqsubseteq A_{p(x)}$ where $A_{p(x)}$ is a fresh concept name;
- for every $p \in \mathtt{trees}(q)$, the CIs $A_p \sqsubseteq \exists r_p.C_p$, $C_p \sqsubseteq A_p$, and $\exists r.A_p \sqsubseteq A_p$ for every role r used in $\mathcal{O}$, where A_p is a fresh concept name and r_p a fresh role name.

Note that $|\mathcal{O}_q| \in |\mathcal{O}| \cdot 2^{O(|q| \cdot \log |q|)}$.

Towards defining the UCQ $\widehat{q}$, consider all CQs that can be obtained as follows. Start with a contraction p of a CQ from q, then choose a set of variables $S \subseteq \mathtt{var}(p)$ such that

$$p^- = p \setminus \{r(x, y) \in p \mid x, y \in S\}$$

is a disjoint union of tree CQs each of which contains at most one variable from S and at least one variable that is not from S. Include in $\widehat{q}$ all CQs that can be obtained by extending $p|_S$ as follows:

1. for every maximal connected component p' of p^- that contains a (unique) variable $x_0 \in S$, add the atom $A_{p'(x_0)}(x_0)$;
2. for every maximal connected component p' of p^- that contains no variable from S, add the atom $A_{p'}(z)$ with z a fresh variable.

It is easy to verify that in Points 1 and 2 above, the CQ p' is in $\mathtt{trees}(q)$ and thus CIs for p' have been introduced in the construction of $\mathcal{O}_q$. The number of CQs in $\widehat{q}$ is single exponential in $|q|$ and each CQ is of size at most $|q|$.

Lemma 3. *For every tree-extended model $\mathcal{I}$ of $\mathcal{D}_0$ and $\mathcal{O}_q$, $\mathcal{I} \models q$ iff $\mathcal{I}|_{\mathtt{adom}(\mathcal{D}_0)} \models \widehat{q}$.*

Let $\mathcal{Q}$ denote the set of all subqueries p of CQs in $\widehat{q}$, that is, p can be obtained from a CQ in $\widehat{q}$ by dropping atoms. A *decorated IOA* for $\mathcal{O}_q$ is a triple $\gamma = (\mathtt{in}, \mathtt{out}, S)$ where $(\mathtt{in}, \mathtt{out})$ is an IOA and S is a set of pairs (p, f) with $p \in \mathcal{Q}$ and $f : \mathtt{var}(p) \to \{1, \dots, k\}$ a total function. We use $\gamma^\mathtt{in}$ to denote $\mathtt{in}$ and likewise for $\gamma^\mathtt{out}$ and γ^S. We argue in the appendix that the number of decorated IOAs is $2^{|\mathcal{O}| \cdot k^{O(|q| \log |q|)}}$.

Every database $\mathcal{D}$ and model $\mathcal{I}$ of $\mathcal{D}$ and $\mathcal{O}_q$ give rise to a decorated IOA $\gamma_{\mathcal{I},\mathcal{D}}$ for $\mathcal{O}_q$ where

- $\gamma^\mathtt{in}_{\mathcal{I},\mathcal{D}}$ and $\gamma^\mathtt{out}_{\mathcal{I},\mathcal{D}}$ are defined as in the previous section and
- γ^S contains all pairs (p, f) with $p \in \mathcal{Q}$ and f a function from $\mathtt{var}(p)$ to $\{1, \dots, k\}$ such that there is a homomorphism h from p to $\mathcal{I}$ with $f(x) = i$ iff $L_i(h(x)) \in \mathcal{D}$ for all variables $x \in \mathtt{var}(p)$.

For all subexpressions s of s_0, our algorithm computes

$$\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}_s} \mid \mathcal{I} \text{ is a tree-extended model of } \mathcal{D}_s \text{ and } \mathcal{O}_q\}.$$

By Lemma 3 and since every tree-extended model of $\mathcal{D}_0$ and $\mathcal{O}$ can be extended to a tree-extended model of $\mathcal{D}_0$ and $\mathcal{O}_q$, this implies that, as desired, $\mathcal{D}_0, \mathcal{O} \not\models q$ iff there is a decorated IOA $\gamma \in \Theta(s_0)$ such that γ^S contains no pair (p, f) with p a CQ in $\widehat{q}$. The sets $\Theta(s)$ are computed as follows:

$\boldsymbol{s = \iota(\mathcal{D})}$: $\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}} \mid \mathcal{I}$ is a tree-extended model of $\mathcal{D}$ and $\mathcal{O}_q\}$.

$\boldsymbol{s = s_1 \oplus s_2}$: $\Theta(s)$ contains a decorated IOA γ for each pair $(\gamma_1, \gamma_2) \in \Theta(s_1) \times \Theta(s_2)$ where, for $1 \le i \le k$:

$$\gamma^\mathtt{in}(i) = \gamma_1^\mathtt{in}(i) \cap \gamma_2^\mathtt{in}(i) \quad \gamma^\mathtt{in}(i) = \gamma_1^\mathtt{out}(i) \cup \gamma_2^\mathtt{out}(i)).$$

and $\gamma^S = \gamma_1^S \cup \gamma_2^S \cup \{(p_1 \cup p_2, f_1 \cup f_2) \mid (p_i, f_i) \in \gamma_i^S$ for $i \in \{1, 2\}$, $p_1 \cup p_2 \in \mathcal{Q}$ and $\mathtt{var}(p_1) \cap \mathtt{var}(p_2) = \emptyset\}$.

$\boldsymbol{s = \alpha^r_{i,j}(s')}$: $\Theta(s)$ contains the decorated IOAs γ that can be obtained by choosing a $\widehat{\gamma} \in \Theta(s')$ that satisfies Conditions a) and b) from Section 3.1 and then defining

- $\gamma^\mathtt{in} = \widehat{\gamma}^\mathtt{in}$ and $\gamma^\mathtt{out} = \widehat{\gamma}^\mathtt{out}$;

- γ^S to be $\widehat{\gamma}^S$ extended with all pairs (p', f) such that $p' \in \mathcal{Q}$ and for some $(p, f) \in \widehat{S}$, p' can be obtained from p by adding zero or more atoms $r(x, y)$, subject to the condition that $f(x) = i$ and $f(y) = j$.

$\boldsymbol{s = \rho_{i\to j}(s')}$: for each $\widehat{\gamma} \in \Theta(s')$, $\Theta(s)$ contains the decorated IOA γ obtained by defining

- $\gamma^{\mathtt{in}}$ and $\gamma^{\mathtt{out}}$ according to Conditions c) to e) in Section 3.1;
- γ^S to contain all pairs (p, f') such that for some $(p, f) \in \widehat{S}$, f' can be obtained by starting with f and then setting $f'(x) = j$ whenever $f(x) = i$.

Note that $\mathcal{Q}$ contains $A_\top(x)$ for every variable x in q and thus we can start with mapping subqueries that contain only a single variable x (and a dummy atom) in the $s = \iota(\mathcal{D})$ case. The algorithm achieves the intended goal.

Lemma 4. *For all subexpressions s of s_0, $\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}_s} \mid \mathcal{I}$ is a tree-extended model of $\mathcal{D}_s$ and $\mathcal{O}_q\}$.*

The running time is analyzed in the appendix.

3.3 $\mathbf{GF_2}$ with AQs

Theorem 3. *In (GF_2, AQ), an OMQ $Q = (\mathcal{O}, q)$ can be evaluated on a database $\mathcal{D}$ of cliquewidth k in time $2^{2^{O(|\mathcal{O}|)}\cdot k^2} \cdot |\mathcal{D}|$.*

As in Section 3.1, we may concentrate on the satisfiability of databases w.r.t. ontologies. To decide whether $\mathcal{D}, \mathcal{O} \models A(\bar{c})$, we then simply check whether $\mathcal{D} \cup \{\overline{A}(\bar{c})\}$ is unsatisfiable w.r.t. the ontology $\mathcal{O} \cup \{\forall \bar{x}\, (\overline{A}(\bar{x}) \to \neg A(\bar{x}))\}$. Note that this also captures 'binary AQs' $A(\bar{c})$.

Assume that we are given as input a GF_2-ontology $\mathcal{O}$, a database $\mathcal{D}_0$, and a k-expression s_0 that generates $\mathcal{D}_0$. For $i \in \{1, 2\}$, we use $\mathtt{cl}_i(\mathcal{O})$ to denote the set of all subformulas of $\mathcal{O}$ with at most i free variables, closed under single negation. For $i = 2$, we additionally assume that $\mathtt{cl}_i(\mathcal{O})$ contains all formulas $r(x, y)$ and $r(y, x)$ with r a role name that occurs in $\mathcal{D}_0$.[2] An *i-type for $\mathcal{O}$* is a set $t \subseteq \mathtt{cl}_i(\mathcal{O})$ such that for some model $\mathcal{I}$ of $\mathcal{O}$ and some $\bar{d} \in (\Delta^{\mathcal{I}})^i$,

$$t = \{\varphi(\bar{x}) \in \mathtt{cl}_i(\mathcal{O}) \mid \mathcal{I} \models \varphi(\bar{e}),\ \bar{e} \text{ a subtuple of } \bar{d} \text{ of length } |\bar{x}| \in \{0, \dots, i\}\}.$$

We then also denote t with $\mathtt{tp}^i_{\mathcal{I}}(\bar{d})$ and use $\mathtt{TP}_i$ to denote the set of all i-types for $\mathcal{O}$. For $t_1, t_2 \in \mathtt{TP}_1$ and $t \in \mathtt{TP}_2$, we say that t_1 and t_2 *are compatible with* t and write $t_1 \rightsquigarrow_t t_2$ if there is a model $\mathcal{I}$ of $\mathcal{O}$ and $d_1, d_2 \in \Delta^{\mathcal{I}}$ such that $\mathtt{tp}^1_{\mathcal{I}}(d_i) = t_i$ for $i \in \{1, 2\}$ and $\mathtt{tp}^2_{\mathcal{I}}(d_1, d_2) = t$. Note that every 1-type contains a 0-type and every 2-type contans a 0-type and two 1-types, identified by the subformulas that use free variable x and y, respectively.

We again traverse the k-expression s_0 bottom-up, computing for each subexpression s a representation of the models of the database $\mathcal{D}_s$ and ontology $\mathcal{O}$. This representation, however, is different from the ones in the previous sections and does not use IOAs. The reason is as follows. A concrete model $\mathcal{I}$ of $\mathcal{D}_s$ and $\mathcal{O}$ assigns a 1-type to every constant in $\mathcal{D}_s$ and thus a set T_i of 1-types to every label i. In $\mathcal{ALCI}$, we may represent each set T_i by the sets $\{\forall r.C \mid \exists t \in T_i : \forall r.C \in t\}$ and $\{C \in \mathtt{cl}^*(\mathcal{O}) \mid \forall t \in T_i : C \in t\}$ of an IOA because they contain all information that we need when using the $\alpha^r_{i,j}$ operator to introduce new edges. The virtue is that there are only single exponentially many IOAs. Now consider the GF_2-ontology $\mathcal{O}$ that contains the sentence

$$\forall x \forall y\, r(x, y) \to \bigwedge_{1 \le i \le n} (A_i(x) \leftrightarrow A_i(y)).$$

The 1-types in T_i may contain any combination of the formulas $A_1(x), \dots, A_n(x)$ and for dealing with the $\alpha^r_{i,j}$ operator we clearly have to know all of them. However, there are double exponentially many sets of such combinations, and this results in a double exponential upper bound.

We now give details of how we represent models. A *multi-edge* is a set of atoms $r(x, y)$ and $r(y, x)$. Let $\mathcal{M}$ denote the set of all multi-edges that only use role names that occur in $\mathcal{D}_0$. For a database $\mathcal{D}$ and distinct $a, b \in \mathtt{adom}(\mathcal{D})$, we use $M_{\mathcal{D}}(a, b)$ to denote the multi-edge that consists of all atoms $r(x, y)$ such that $\mathcal{D}$ contains the fact $r(a, b)$, and $r(y, x)$ such that $\mathcal{D}$ contains the fact $r(b, a)$.

A *model abstraction* is a pair $\gamma = (T, E)$ where

- $T : \{1, \dots, k\} \to 2^{\mathtt{TP}_1}$ is a partial function that associates each label with a set of 1-types;
- $E : \{1, \dots, k\}^2 \to 2^{\mathtt{TP}_1 \times \mathcal{M} \times \mathtt{TP}_1}$ is a partial function that associates each pair of labels (i, j), where $i \neq j$, with a set of triples (t_1, M, t_2) from $\mathtt{TP}_1 \times \mathcal{M} \times \mathtt{TP}_1$

such that all 1-types in the ranges of T and E contain exactly the same subformulas of $\mathcal{O}$ that are sentences, that is, they contain the same 0-type. We use γ^0 to denote this 0-type, γ^T to denote T, and likewise for γ^E. We use multi-edges in model abstractions rather than 2-types as the former allow us to add edges later when the operator $\alpha^r_{i,j}$ is applied whereas the latter fix all edges from the beginning.

Every database $\mathcal{D}$ and model $\mathcal{I}$ of $\mathcal{D}$ give rise to a model abstraction $\gamma_{\mathcal{I},\mathcal{D}}$ defined by setting, for $1 \le i, j \le k$,

$$\gamma^T_{\mathcal{I},\mathcal{D}}(i) = \{\mathtt{tp}^1_{\mathcal{I}}(c) \mid L_i(c) \in \mathcal{D}\}$$
$$\gamma^E_{\mathcal{I},\mathcal{D}}(i, j) = \{(\mathtt{tp}^1_{\mathcal{I}}(c), M_{\mathcal{D}}(c, c'), \mathtt{tp}^1_{\mathcal{I}}(c')) \mid L_i(c) \in \mathcal{D} \text{ and } L_j(c') \in \mathcal{D}\}.$$

Our algorithm computes, for each subexpression s of s_0,

$$\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}_s} \mid \mathcal{I} \text{ model of } \mathcal{D}_s \text{ and } \mathcal{O}\}.$$

It then remains to check whether $\Theta(s_0)$ is non-empty. The sets $\Theta(s)$ are computed as follows:

$\boldsymbol{s = \iota(\mathcal{D})}$: Set $\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}} \mid \mathcal{I}$ model of $\mathcal{D}$ and $\mathcal{O}\}$.

$\boldsymbol{s = s_1 \oplus s_2}$: $\Theta(s)$ contains a model abstraction $\gamma = (T, F)$ for each pair of model abstractions $(\gamma_1, \gamma_2) \in \Theta(s_1) \times \Theta(s_2)$ with $\gamma^0_1 = \gamma^0_2$ defined by setting, for $1 \le i, j \le k$,

$$\gamma^T(i) = \gamma^T_1(i) \cup \gamma^T_2(i)$$
$$\gamma^E(i, j) = \gamma^E_1(i, j) \cup \gamma^E_2(i, j) \cup \{(t_1, \emptyset, t_2) \mid t_1 \in \gamma^T_1(i) \text{ and } t_2 \in \gamma^T_2(j)\} \cup \{(t_1, \emptyset, t_2) \mid t_1 \in \gamma^T_2(i) \text{ and } t_2 \in \gamma^T_1(j)\}.$$

[2] Recall that x and y are the two fixed (and distinct) variables admitted in GF_2.

$\boldsymbol{s = \alpha^r_{i,j}(s')}$: $\Theta(s)$ contains a model abstraction γ for every model abstraction $\widehat{\gamma} \in \Theta(s')$ such that

(∗) for all $(t_1, M, t_2) \in \widehat{\gamma}^E(i,j)$, there is a $t \in \mathtt{TP}_2$ such that $M \cup \{r(x,y)\} \subseteq t$ and $t_1 \rightsquigarrow_t t_2$.

The model abstraction γ is defined like $\widehat{\gamma}$ except that

for all $(t_1, M, t_2) \in \widehat{\gamma}^E(i,j)$, add $r(x,y)$ to M, and
for all $(t_1, M, t_2) \in \widehat{\gamma}^E(j,i)$, add $r(y,x)$ to M.

$\boldsymbol{s = \rho_{i \to j}(s')}$: for each $\widehat{\gamma} \in \Theta(s')$, $\Theta(s)$ contains the model abstraction γ defined like $\widehat{\gamma}$ except that, for $1 \leq \ell \leq k$,

$$\gamma^T(i) = \emptyset \qquad \gamma^E(i,\ell) = \emptyset$$
$$\gamma^T(j) = \widehat{\gamma}^T(j) \cup \widehat{\gamma}^T(i) \qquad \gamma^E(\ell,i) = \emptyset$$
$$\gamma^E(j,\ell) = \widehat{\gamma}^E(j,\ell) \cup \widehat{\gamma}^E(i,\ell)$$
$$\gamma^E(\ell,j) = \widehat{\gamma}^E(\ell,j) \cup \widehat{\gamma}^E(\ell,i).$$

The algorithm achieves the intended goal and runs within the stated time bounds.

Lemma 5. *For all subexpressions s of s_0, $\Theta(s) = \{\gamma_{\mathcal{I},\mathcal{D}_s} \mid \mathcal{I}$ model of $\mathcal{D}_s$ and $\mathcal{O}\}$.*

4 Lower Bounds

We prove that for $(\mathrm{GF}_2, \mathrm{AQ})$ and $(\mathcal{ALC}, \mathrm{CQ})$, the double exponential running time in $|\mathcal{O}|$ cannot be improved unless the exponential time hypothesis (ETH) is false. The ETH states that if δ is the infimum of real numbers such that 3SAT can be solved in time $O(2^{\delta n})$, where n is the number of variables in the input formula, then $\delta > 0$. The following are the two main results of this section.

Theorem 4. *If the ETH is true, then there is no algorithm with running time $2^{2^{o(|Q|)}} \cdot \mathtt{poly}(|\mathcal{D}|)$ for OMQ evaluation in*

1. *(GF_2, AQ) on databases of cliquewidth 2;*
2. *$(\mathcal{ALC}, CQ)$ on databases of cliquewidth 3.*

Note that OMQ evaluation in $(\mathrm{GF}_2, \mathrm{AQ})$ and $(\mathcal{ALC}, \mathrm{CQ})$ are only ExpTime-complete in combined complexity on unrestricted databases. The former is folklore and the latter is proved in (Lutz 2008). Thus, Theorem 4 implies that, in these two OMQ languages, one has to pay for the running time to be polynomial in $|\mathcal{D}|$ on databases of bounded cliquewidth by an exponential increase in the running time in $|Q|$.

We now prove Theorem 4. A major contribution of the original paper bringing forward the ETH is the sparsification lemma, which implies the following useful consequence of the ETH (Impagliazzo, Paturi, and Zane 2001):

(∗) If the ETH is true, then 3SAT cannot be solved in time $2^{o(m)}$, with m the number of clauses of the input formula.

Both results in Theorem 4 are proved by contradiction against (∗), i.e. we show that an algorithm for these OMQ evaluation problems with running time $2^{2^{o(|Q|)}} \cdot \mathtt{poly}(|\mathcal{D}|)$ can be used to design an algorithm for 3SAT with running time $2^{o(m)}$, contradicting (∗). To achieve this, it suffices to give a polynomial time reduction from 3SAT to OMQ evaluation that produces an ontology of size $O(\log m)$ where m is the number of clauses in the input formula. For both points of Theorem 4, we first use a known reduction from 3SAT to List Coloring (Theorem 5) and then reduce List Coloring to OMQ evaluation. In List Coloring, the input is a graph in which every vertex is associated with a list of colors, and the question is whether it is possible to assign to each vertex a color from its list and obtain a valid coloring of the graph. Formally: Given an undirected graph $G = (V, E)$ and a set $C_v \subseteq \mathbb{N}$ of colors for each node $v \in V$, decide whether there is a map $f : V \to \mathbb{N}$ such that $f(v) \in C_v$ for every $v \in V$ and $f(u) \neq f(v)$ for every $\{u, v\} \in E$.

The following is proved in (Jansen 1996), Theorem 11. The graphs of cliquewidth 2 that are used in the proof are (non-disjoint) unions of two cliques.

Theorem 5. *There is a polynomial time reduction from* 3SAT *to* List Coloring *that turns a formula with m clauses into an instance of* List Coloring *with a connected graph of cliquewidth 2 and $O(m)$ colors.*

To prove Point 1 of Theorem 4, we reduce List Coloring to OMQ evaluation in $(\mathrm{GF}_2, \mathrm{AQ})$. To highlight the idea of the reduction, we first sketch the proof of a slightly weaker version of Point 1 of Theorem 4, namely that there is no algorithm with the stated running time on databases of cliquewidth 3 (instead of 2). Moreover, we use queries of the form $\exists x\, B(x)$ rather than AQs. As an intermediate, we consider the problem Precoloring Extension, where the input is a number of colors k and a graph in which some nodes are already colored, and the question is whether the coloring can be extended to a k-coloring of the whole graph. Consider the following standard reduction from List Coloring to Precoloring Extension (Fellows et al. 2011). For every vertex v and every color $c \notin C_v$, add a new vertex that is precolored with c and is only adjacent to v. This reduction increases the cliquewidth by at most 1, so when starting from Theorem 5, the resulting graphs have cliquewidth at most 3. It remains to reduce Precoloring Extension to OMQ evaluation.

Without the precoloring, we may view the input graph to Precoloring Extension as a database $\mathcal{D}$ by replacing every vertex by a constant and every edge $\{u, v\}$ by two facts $r(u, v)$ and $r(v, u)$. Let us assume that the number of colors is a power of 2, say 2^ℓ. Each of the 2^ℓ colors is assigned a unique bit string of length ℓ. Introduce concept names $A^j_1, \ldots, A^j_\ell$ for $j \in \{0, 1\}$ and add the sentence $\forall x(A^0_i(x) \leftrightarrow \neg A^1_i(x))$ to the ontology $\mathcal{O}$, which says that every constant in $\mathcal{D}$ should be assigned a unique sequence of ℓ bits, representing a color. To express that adjacent vertices should be colored differently, we just have to say that their bit strings should differ in at least one place: $\forall x \forall y\,(r(x, y) \to \bigvee_{i=1}^{\ell} A^0_i(x) \leftrightarrow A^1_i(y))$. For every vertex v that is precolored by a color c, we add facts $A^j_i(v)$ that represent this color. As the query, choose $\exists x\, B(x)$ with B a fresh concept name. This query is implied if and only if $\mathcal{D}$ is not satisfiable w.r.t. $\mathcal{O}$, which is the case if and only if we started with a no-instance of Precoloring Extension.

The following lemma improves the reduction to use only databases of cliquewidth 2 and AQs.

Lemma 6. *There is a polynomial time reduction from* List Coloring *on connected graphs of cliquewidth 2 to OMQ*

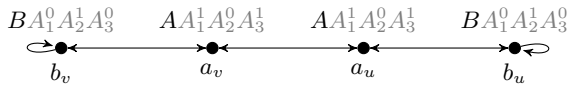

Figure 1: Black: Fragment of $\mathcal{D}$ that represents edge $\{u, v\} \in E$. Gray: Concept names that could appear in an interpretation where the coloring has a defect, for $\ell = 3$.

evaluation in (GF_2, AQ) *on databases of cliquewidth 2, where the constructed ontology is of size* $O(\log(k))$ *with* k *the number of distinct colors of the input graph.*

We are now ready to prove Point 1 of Theorem 4.

Proof. Assume for a contradiction that there is an algorithm with running time $2^{2^{o(|\mathcal{O}|)}} \cdot \mathtt{poly}(|\mathcal{D}|)$ for OMQ evaluation in $(\mathrm{GF}_2, \mathrm{AQ})$ on databases of cliquewidth at most 2. Then the following is an algorithm for 3SAT. Given a formula in 3CNF with m clauses, apply the reduction from Theorem 5 followed by the one from Lemma 6. This yields an OMQ $Q \in (\mathrm{GF}_2, \mathrm{AQ})$ where the ontology $\mathcal{O}$ is of size $O(\log(m))$, a database $\mathcal{D}$ of cliquewidth 2 and an answer candidate $a \in \mathtt{adom}(\mathcal{D})$. Use the algorithm for OMQ evaluation in $(\mathrm{GF}_2, \mathrm{AQ})$ whose existence we assumed on input Q, $\mathcal{D}$ and a. The running time of the first step is polynomial. The running time of the second step is $2^{2^{o(\log m)}} \cdot \mathtt{poly}(|\mathcal{D}|)$, which is $2^{o(m)}$, contradicting the ETH. ❑

Interestingly, the reduction does not rely on the additional expressive power of GF_2 compared to $\mathcal{ALC}$, but rather on the ability of GF_2 to express certain statements more succinctly. To demonstrate this, we introduce the extension $\mathcal{ALC}^=$ of $\mathcal{ALC}$ with the following constructor:

If $A_1, \dots, A_n$ are concept names, then $\exists r.[A_1, \dots, A_n]_=$ is a concept. The semantics is defined as follows:

$$\exists r.[A_1, \dots, A_n]_=^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \text{ and } d \in A_i^{\mathcal{I}} \Leftrightarrow e \in A_i^{\mathcal{I}} \text{ for all } i \in \{1, \dots, n\}\}.$$

It is easy to see that the expressive power of $\mathcal{ALC}$ and of $\mathcal{ALC}^=$ are identical, as the additional constructor $\exists r.[A_1, \dots, A_n]_=$ can be expressed as

$$\bigsqcup_{C_1 \in \{A_1, \neg A_1\}} \dots \bigsqcup_{C_n \in \{A_n, \neg A_n\}} \big(C_1 \sqcap \dots \sqcap C_n \sqcap \exists r.(C_1 \sqcap \dots \sqcap C_n)\big).$$

Note that the above concept is exponentially larger than the original one. In GF_2, in contrast, the new constructor can easily be expressed using only a linear size formula.

In the reduction used to prove Lemma 6, we may replace the GF_2 ontology by the following $\mathcal{ALC}^=$-ontology:

$$\begin{aligned}\mathcal{O} = &\{\exists s.[A_1, \dots, A_n]_= \sqsubseteq D\} \cup \\ &\{B \sqcap \exists r.[A_1, \dots, A_n, B]_= \sqsubseteq D\} \cup \\ &\{\exists r.D \sqsubseteq D, D \sqsubseteq \forall r.D, \exists s.D \sqsubseteq D, D \sqsubseteq \forall s.D\} \cup \\ &\{\top \sqsubseteq A_i^0 \sqcup A_i^1, A_i^0 \sqcap A_i^1 \sqsubseteq \bot \mid i \in \{1, \dots, \ell\}\}.\end{aligned}$$

Thus, Point 1 of Theorem 4 still holds if $(\mathrm{GF}_2, \mathrm{AQ})$ is replaced with $(\mathcal{ALC}^=, \mathrm{AQ})$.

To prove Point 2 of Theorem 4, we use a reduction that is similar to the one presented before Lemma 6. This time we are only allowed to use an $\mathcal{ALC}$ ontology, but also CQs in place of AQs, which again allows us to express succinctly that adjacent nodes have to be colored differently.

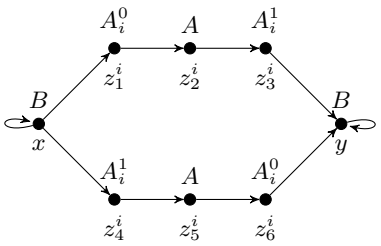

Figure 2: Gadget i of CQ q. Every edge represents an r-fact. The variables x and y are shared among gadgets.

Lemma 7. *There is a polynomial time reduction from* List Coloring *on connected graphs of cliquewidth 2 to OMQ evaluation in* $(\mathcal{ALC}, \mathrm{CQ})$ *on databases of cliquewidth 3, where the constructed OMQ is of size* $O(\log k)$*,* k *the number of distinct colors of the input graph.*

Proof. Let $G = (V, E)$ be an undirected graph of cliquewidth 2 and $C_v \subseteq \mathbb{N}$ a list of possible colors for every $v \in V$. We construct a database $\mathcal{D}$ of cliquewidth 3 and a Boolean OMQ $Q = (\mathcal{O}, q)$ from $(\mathcal{ALC}, \mathrm{CQ})$ such that G and C_v are a yes-instance for List Coloring if and only if $\mathcal{D} \not\models Q$. In contrast to what was described above, we do not carry out a reduction from List Coloring to Pre-coloring Extension as a separate first step, but build it directly into the main reduction. Let k be the number of distinct colors that occur, w.l.o.g. let $\bigcup_{v \in V} C_v = \{1, \dots, k\}$. Define $\ell = \lceil \log k \rceil$ and $L = \{1, \dots, 2^\ell\}$.

We use concept names A_i^j, $i \in \{1, \dots, \ell\}$ and $j \in \{0, 1\}$, another concept name B, one role name r, and constants a_v, b_v, a_v^c, and b_v^c for every $v \in V$ and $c \in L \setminus C_v$. The database is defined as

$$\begin{aligned}\mathcal{D} = &\{r(a_v, a_u), r(a_u, a_v) \mid \{u, v\} \in E\} \cup \\ &\{r(a_v, a_v^c), r(a_v^c, a_v) \mid v \in V, c \in L \setminus C_v\} \cup \\ &\{A_i^j(a_v^c) \mid v \in V, c \in L \setminus C_v \text{ and the } i\text{-th bit of } c \text{ is } j\} \cup \\ &\{r(a_v, b_v), r(b_v, a_v), r(b_v, b_v) \mid v \in V\} \cup \\ &\{r(a_v^c, b_v^c), r(b_v^c, a_v^c), r(b_v^c, b_v^c) \mid v \in V, c \in L \setminus C_v\} \cup \\ &\{A(a_v), A(a_v^c) \mid v \in V, c \in L \setminus C_v\} \cup \\ &\{B(b_v), B(b_v^c) \mid v \in V, c \in L \setminus C_v\}.\end{aligned}$$

Here, every constant a_v represents a vertex of G, and every constant a_v^c is a constant adjacent to a_v that is precolored with the color c. Furthermore, for every constant a_v (resp. a_v^c) there is a constant b_v (resp. b_v^c) such that b_v (resp. b_v^c) is adjacent only to a_v (resp. a_v^c). We refer to the b_v and b_v^c as the *copies* of a_v and a_v^c. Every copy has an r-selfloop and a unary marker B, and every non-copy has a unary marker A. Figure 1 depicts a part of the database. We argue in the appendix that $\mathcal{D}$ has cliquewidth 3.

Next, we construct the ontology, which has two purposes. First, we assign a unique color to every constant of the database. For every $i \in \{1, \dots, \ell\}$, we include the CIs $\top \sqsubseteq A_i^0 \sqcup A_i^1$ and $A_i^0 \sqcap A_i^1 \sqsubseteq \bot$. The second purpose is to assure that every copy b_v (resp. b_v^c) is colored using the anti-color of the color of a_v (resp. a_v^c). Here, the *anti-color*

of a color c is the color obtained from c by flipping every bit, e.g. the anti-color of 10010 is 01101. Since the copies b_v and b_v^c carry the concept name B, this can be achieved using the following CIs for every $i \in \{1, \dots, \ell\}$ and $j \in \{0, 1\}$:

$$A \sqcap B \sqsubseteq \bot \qquad B \sqcap A_i^j \sqsubseteq \forall r.(B \sqcup A_i^{1-j}).$$

Now we construct the (Boolean) CQ q, whose purpose it is to detect a defect in the coloring. It is a union of ℓ gadgets, with gadget number i checking that the colors of two adjacent vertices agree on bit number i. We use two variables x and y, as well as variables $z_1^i, z_2^i, \dots, z_6^i$ for every $i \in \{1, \dots, \ell\}$. The variables x and y are shared between all the gadgets, whereas the variables $z_1^i, z_2^i, \dots, z_6^i$ belong to gadget number i. The i-th gadget is displayed in Figure 2. It is clear that the OMQ $(\mathcal{O}, q)$ is of size $O(\log k)$.

Claim. G and C_v are a yes-instance for LIST COLORING if and only if $\mathcal{D} \not\models Q$.

We only show the 'if' direction here as it demonstrates how the constructed CQ q can identify defects in the coloring. Let $\mathcal{I}$ be a model of $\mathcal{O}$ and $\mathcal{D}$ such that $\mathcal{I} \not\models q$. We show that G and the C_v are a yes-instance of LIST COLORING. Since $\mathcal{I}$ is a model of $\mathcal{O}$, every a_v is assigned a unique color c_v, encoded using the A_i^j. We define $f(v) = c_v$ for every $v \in V$. If any two adjacent constants a_u and a_v were colored with the same color $c = j_1 j_2 \dots j_\ell \in \{0, 1\}^\ell$, then we can construct a homomorphism h from q to $\mathcal{I}$ by $h(x) = b_v, h(y) = b_u$ and, if $j_i = 0$, then

$$h(z_1^i) = a_v, h(z_2^i) = a_u, h(z_3^i) = b_u$$
$$h(z_4^i) = b_v, h(z_5^i) = a_v, h(z_6^i) = a_u$$

and if $j_i = 1$, then

$$h(z_1^i) = b_v, h(z_2^i) = a_v, h(z_3^i) = a_u$$
$$h(z_4^i) = a_v, h(z_5^i) = a_u, h(z_6^i) = b_u\,.$$

Figure 1 depicts the case where $c = 101$. But we assumed that $\mathcal{I} \not\models q$, so every two adjacent a_v and a_u have different colors. Thus, f witnesses that G and the C_v are a yes-instance for LIST COLORING. ❑

With Lemma 7 at hand, the proof for Point 2 of Theorem 4 continues exactly as the proof of Point 1.

5 Bounded Treewidth

We prove two results that concern OMQ evaluation on databases of bounded treewidth, mainly to contrast with the results obtained for databases of bounded cliquewidth.

Theorem 6. *In (GF_2, AQ), an OMQ $Q = (\mathcal{O}, q)$ can be evaluated on a database $\mathcal{D}$ of treewidth k in time $2^{O(|\mathcal{O}| \cdot k^2)} \cdot |\mathcal{D}|$.*

Note that we obtain single exponential running time while Theorem 4 states that when treewidth is replaced by cliquewidth, we cannot be better than double exponential.

Theorem 7. *In (GF_2, UCQ), evaluating an OMQ $Q = (\mathcal{O}, q)$ on a database $\mathcal{D}$ of treewidth k is possible in time $2^{|\mathcal{O}| \cdot k^{O(|q| \log(|q|))}} \cdot |\mathcal{D}|$.*

Theorem 7 is relevant because we leave open the corresponding case for cliquewidth, that is, we do not even know whether OMQ evaluation in $(\mathrm{GF}_2, \mathrm{UCQ})$ on databases of bounded cliquewidth is in PTIME in data complexity. We remark that the overall running time in Theorem 7 cannot be improved to single exponential because OMQ evaluation in $(\mathcal{ALCI}, \mathrm{CQ})$ is 2EXPTIME-complete in combined complexity on databases of the form $\{A(c)\}$ (Lutz 2007).

We give a single algorithm that yields both Theorem 6 and 7. Assume that we are given as input an OMQ $Q = (\mathcal{O}, q) \in (\mathrm{GF}_2, \mathrm{UCQ})$, and a database $\mathcal{D}_0$ of treewidth k. As in the proof of Theorem 2, we may assume w.l.o.g. that Q is Boolean. In contrast to the case of cliquewidth, we do not need to assume that a tree decomposition is given as part of the input. In fact, a tree decomposition $T = (V, E, (B_v)_{v \in V})$ of $\mathcal{D}_0$ of width at most $2k + 1$ can be computed in time $2^{O(k)} \cdot |\mathtt{adom}(\mathcal{D}_0)|$ (Korhonen 2021).

Our algorithm traverses the tree decomposition T bottom-up. For every node v, we compute a representation of the models of $\mathcal{D}_v$ and $\mathcal{O}$, where $\mathcal{D}_v$ is the restriction of $\mathcal{D}_0$ to the constants that occur in the bags of the subtree of T rooted at v. Along with the representations of models, we keep track of partial homomorphisms from CQs in q to these models (we actually rewrite the query beforehand, in the style of Section 3.2). Details are in the appendix.

6 Conclusion

We leave open the interesting question whether OMQ evaluation in $(\mathrm{GF}_2, (\mathrm{U})\mathrm{CQ})$ on databases of bounded cliquewidth is in PTIME in data complexity. While all other OMQ languages studied in this paper are easily translated into MSO_1, this is not the case for $(\mathrm{GF}_2, (\mathrm{U})\mathrm{CQ})$. In fact, it is straightforward to express the problem MONOCHROMATIC TRIANGLE[3] in $(\mathrm{GF}_2, \mathrm{CQ})$ and to our knowledge it is open whether this problem can be expressed in MSO_1.

It would also be interesting to consider natural extensions of $\mathcal{ALCI}$. We believe that adding role inclusions has no impact on the obtained results. It is less clear what happens for transitive roles, number restrictions, and nominals. In the appendix, we make the following observation for the extension $\mathcal{ALCF}$ of $\mathcal{ALC}$ with (globally) functional roles.

Theorem 8. *OMQ evaluation in $(\mathcal{ALCF}, CQ)$ on databases of treewidth 2 is* CONP*-hard in data complexity if the unique name assumption is not made.*

The proof relies on an observation from (Figueira 2016). We conjecture that when the unique name assumption is made, fixed-parameter linearity is regained and can be proved using the methods in this paper.

It would also be very interesting to pursue the idea of decomposing databases into a (hopefully small) part of high cliquewidth and parts of low cliquewidth, with the aim of achieving efficiency in practical applications.

Acknowledgements

Supported by the DFG CRC 1320 EASE - Everyday Activity Science and Engineering.

[3] Can the edges of a given graph be colored with two colors without generating a monochromatic triangle?

References

Akiba, T.; Sommer, C.; and Kawarabayashi, K. 2012. Shortest-path queries for complex networks: exploiting low tree-width outside the core. In *Proc. of EDBT*, 144–155. ACM.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Bienvenu, M., and Ortiz, M. 2015. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web*, 218–307.

Bienvenu, M.; ten Cate, B.; Lutz, C.; and Wolter, F. 2014. Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. *ACM Transactions on Database Systems* 39(4):33:1–33:44.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; and Rosati, R. 2009. Ontologies and databases: The DL-Lite approach. In *Reasoning Web*, volume 5689 of *LNCS*, 255–356.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2022. *Introduction to algorithms*. MIT press.

Corneil, D. G., and Rotics, U. 2005. On the relationship between clique-width and treewidth. *SIAM Journal on Computing* 34(4):825–847.

Courcelle, B., and Engelfriet, J. 2012. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press.

Courcelle, B.; Makowsky, J. A.; and Rotics, U. 2000. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.* 33(2):125–150.

Downey, R. G., and Fellows, M. R. 2013. *Fundamentals of parameterized complexity*, volume 4. Springer.

Fellows, M. R.; Fomin, F. V.; Lokshtanov, D.; Rosamond, F. A.; Saurabh, S.; Szeider, S.; and Thomassen, C. 2011. On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput.* 209(2):143–153.

Figueira, D. 2016. Semantically acyclic conjunctive queries under functional dependencies. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, 847–856.

Ganian, R.; Hlinený, P.; Langer, A.; Obdrzálek, J.; Rossmanith, P.; and Sikdar, S. 2014. Lower bounds on the complexity of mso_1 model-checking. *J. Comput. Syst. Sci.* 80(1):180–194.

Grädel, E. 1999. On the restraining power of guards. *J. Symb. Log.* 64(4):1719–1742.

Haga, A.; Lutz, C.; Sabellek, L.; and Wolter, F. 2021. How to approximate ontology-mediated queries. In *Proc. of KR*, 323–333.

Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.* 63(4):512–530.

Jansen, K. 1996. Complexity results for the optimum cost chromatic partition problem. *Universität Trier, Mathematik/Informatik, Forschungsbericht* 96-41.

Korhonen, T. 2021. A single-exponential time 2-approximation algorithm for treewidth. *arXiv preprint arXiv:2104.07463*.

Kreutzer, S., and Tazari, S. 2010. Lower bounds for the complexity of monadic second-order logic. In *Proc. of LICS*, 189–198. IEEE Computer Society.

Lutz, C.; Sabellek, L.; and Schulze, L. 2022. Ontology-mediated querying on databases of bounded cliquewidth. *CoRR* abs/2205.02190.

Lutz, C. 2007. Inverse roles make conjunctive queries hard. In *Proc. of DL2007*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Lutz, C. 2008. Two upper bounds for conjunctive query answering in SHIQ. In *Proc. of DL2008*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Maniu, S.; Cheng, R.; and Senellart, P. 2017. An indexing framework for queries on probabilistic graphs. *ACM Trans. Database Syst.* 42(2):13:1–13:34.

Maniu, S.; Senellart, P.; and Jog, S. 2019. An experimental study of the treewidth of real-world graph data. In *Proc. of ICDT*, volume 127 of *LIPIcs*, 12:1–12:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Oum, S.-I. 2008. Approximating rank-width and clique-width quickly. *ACM Transactions on Algorithms (TALG)* 5(1):1–20.

Wei, F. 2010. TEDI: efficient shortest path query answering on graphs. In *Proc. ofSIGMOD*, 99–110. ACM.

Zhou, Y.; Cuenca Grau, B.; Nenov, Y.; Kaminski, M.; and Horrocks, I. 2015. PAGOdA: Pay-as-you-go ontology query answering using a datalog reasoner. *J. Artif. Intell. Res.* 54:309–367.