# Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP

MEGHYN BIENVENU, CNRS and Université Paris Sud
BALDER TEN CATE, University of California, Santa Cruz
CARSTEN LUTZ, Universität Bremen
FRANK WOLTER, University of Liverpool

Ontology-based data access is concerned with querying incomplete data sources in the presence of domain-specific knowledge provided by an ontology. A central notion in this setting is that of an *ontology-mediated query*, which is a database query coupled with an ontology. In this article, we study several classes of ontology-mediated queries, where the database queries are given as some form of conjunctive query and the ontologies are formulated in description logics or other relevant fragments of first-order logic, such as the guarded fragment and the unary negation fragment. The contributions of the article are threefold. First, we show that popular ontology-mediated query languages have the same expressive power as natural fragments of disjunctive datalog, and we study the relative succinctness of ontology-mediated queries and disjunctive datalog queries. Second, we establish intimate connections between ontology-mediated queries and constraint satisfaction problems (CSPs) and their logical generalization, MMSNP formulas. Third, we exploit these connections to obtain new results regarding: (i) first-order rewritability and datalog rewritability of ontology-mediated queries; (ii) P/NP dichotomies for ontology-mediated queries; and (iii) the query containment problem for ontology-mediated queries.

Categories and Subject Descriptors: H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Languages, Theory

Additional Key Words and Phrases: Ontology-based data access, query answering, query rewriting

**ACM Reference Format:**
Meghyn Bienvenu, Balder Ten Cate, Carsten Lutz, and Frank Wolter. 2014. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. ACM Trans. Datab. Syst. 39, 4, Article 33 (December 2014), 44 pages.
DOI: http://dx.doi.org/10.1145/2661643

## 1. INTRODUCTION

Ontologies are logical theories that formalize domain-specific knowledge, thereby making it available for machine processing. Recent years have seen an increasing interest in using ontologies in data-intensive applications, especially in the context of intelligent systems, the Semantic Web, and in data integration. A much studied scenario is that of answering queries over an incomplete database under the open-world

M. Bienvenu was supported by the ANR project PAGODA (ANR-12-JS02-007-01). B. ten Cate was supported by NSF grants IIS-0905276 and IIS-1217869. C. Lutz was supported by the DFG SFB/TR 8 "Spatial Cognition."
Authors' addresses: M. Bienvenu (corresponding author), Laboratoire de Recherche en Informatique, Universite Paris-Sud, 15 Rue Georges Clemenceau, 91400 Orsay, France; email: meghyn@lri.fr; B. ten Cate, University of California Santa Cruz, 1156 High Street, Santa Cruz, CA 95064; C. Lutz, Fachbereich Informatik, Universitat Bremen, Bibliothekstraße 1, 28359 Bremen, Germany; F. Wolter, Department of Computer Science, University of Liverpool, Liverpool, Merseyside L69 3BX, UK.

DOI: http://dx.doi.org/10.1145/2661643

semantics, taking into account knowledge provided by an ontology [Calvanese et al. 1998, 2007; Calì et al. 2012]. We refer to this as *ontology-based data access* (OBDA).

There are several important use-cases for OBDA. A classical one is to enrich an incomplete data source with background knowledge in order to obtain a more complete set of answers to a query. For example, if a medical patient database contains the facts that patient1 has finding Erythema Migrans and patient2 has finding Lyme disease, and the ontology provides the background knowledge that a finding of Erythema Migrans is sufficient for diagnosing Lyme disease, then both patient1 and patient2 can be returned when querying for patients that have the diagnosis Lyme disease. This use of ontologies is central to query answering in the Semantic Web. OBDA can also be used to enrich the data schema (that is, the relation symbols allowed in the presentation of the data) with additional symbols to be used in a query. For example, a patient database may contain facts such as patient1 has diagnosis Lyme disease and patient2 has diagnosis Listeriosis, and an ontology could add the knowledge that Lyme disease and Listeriosis are both bacterial infections, thus enabling queries such as "return all patients with a bacterial infection" despite the fact that the data schema does not include a relation or attribute explicitly referring to bacterial infections. Especially in the biomedical domain, applications of this kind are fueled by the availability of comprehensive professional ontologies such as SNOMED CT and FMA. A third prominent application of OBDA is in data integration, where an ontology can be used to provide a uniform view on multiple data sources [Poggi et al. 2008]. This typically involves mappings from the source schemas to the schema of the ontology, which we will not explicitly consider here.

We may view the actual database query and the ontology as two components of one composite query that we call an *ontology-mediated query*. OBDA in the prior sense is then the problem of answering ontology-mediated queries. The database queries used in OBDA are typically unions of conjunctive queries, while the ontologies are specified in an ontology language that is either a description logic or, more generally, a suitable fragment of first-order logic. For popular choices of ontology languages, the data complexity of ontology-mediated queries can be coNP-complete, which has resulted in extensive research on finding tractable classes of ontology-mediated queries, as well as on finding classes of ontology-mediated queries that are amenable to efficient query answering techniques [Calvanese et al. 2006; Hustadt et al. 2007; Krisnadhi and Lutz 2007]. In particular, classes of ontology-mediated queries have been identified that admit an FO rewriting (i.e., that are equivalent to a first-order query) or, alternatively, admit a datalog rewriting. FO rewritings make it possible to answer ontology-mediated queries using traditional database management systems while datalog rewritings enable the use of datalog engines. This approach is considered one of the most promising for OBDA and is the subject of significant research activity; see, for example, Calvanese et al. [2007], Gottlob and Schwentick [2012], Kikot et al. [2012b], Kontchakov et al. [2010], Pérez-Urbina et al. [2010], and Rosati and Almatelli [2010].

The main aims of this article are: (i) to characterize the expressive power of ontology-mediated queries, both in terms of more traditional database query languages and from a descriptive complexity perspective; (ii) to make progress towards complete and decidable classifications of ontology-mediated queries with respect to their data complexity; and (iii) to establish decidability and tight complexity bounds for relevant reasoning problems such as query containment and deciding whether a given ontology-mediated query is FO rewritable or datalog rewritable.

We take an ontology-mediated query to be a triple $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S}$ is a *data schema*, $\mathcal{O}$ an ontology, and $q$ a query. Here, the data schema $\mathbf{S}$ fixes the set of relation symbols allowed in the data and the ontology $\mathcal{O}$ is a logical theory that may use the relation

symbols from $\mathbf{S}$ as well as additional symbols. The query $q$ can use any relation symbol that occurs in $\mathbf{S}$ or in $\mathcal{O}$. As ontology languages, we consider a range of standard description logics (DLs) and several fragments of first-order logic that embed ontology languages such as Datalog$^{\pm}$ [Calì et al. 2009], namely the guarded fragment (GFO), the unary negation fragment (UNFO), and the guarded negation fragment (GNFO). As query languages for $q$, we focus on unions of conjunctive queries (UCQs) and unary atomic queries (AQs). The latter are of the form $A(x)$, with $A$ a unary relation symbol, and correspond to what are traditionally called *instance queries* in the OBDA literature (note that $A$ may be a relation symbol from $\mathcal{O}$ that is not part of the data schema). These two query languages are among the most used in OBDA. In the following, we use $(\mathcal{L}, \mathcal{Q})$ to denote the query language that consists of all ontology-mediated queries $(\mathbf{S}, \mathcal{O}, q)$ with $\mathcal{O}$ specified in the ontology language $\mathcal{L}$ and $q$ specified in the query language $\mathcal{Q}$. For example, (GFO, UCQ) refers to ontology-mediated queries in which $\mathcal{O}$ is a GFO ontology and $q$ is a UCQ. We refer to such query languages $(\mathcal{L}, \mathcal{Q})$ as *ontology-mediated query languages* (OBDA languages).

In Section 3, we characterize the expressive power of OBDA languages in terms of natural fragments of (negation-free) disjunctive datalog. We first consider the basic description logic $\mathcal{ALC}$. We show that $(\mathcal{ALC}, \text{UCQ})$ has the same expressive power as monadic disjunctive datalog (abbreviated MDDlog) and that $(\mathcal{ALC}, \text{AQ})$ has the same expressive power as unary queries defined in a syntactic fragment of MDDlog that we call connected simple MDDlog. Similar results hold for various description logics that extend $\mathcal{ALC}$ with, for example, inverse roles, role hierarchies, and the universal role, all of which are standard operators included in the W3C standardized ontology language OWL2 DL. Turning to other fragments of first-order logic, we then show that (UNFO, UCQ) also has the same expressive power as MDDlog, while (GFO, UCQ) and (GNFO, UCQ) are strictly more expressive and coincide in expressive power with frontier-guarded disjunctive datalog, which is the DDlog fragment given by programs in which, for every atom $\alpha$ in the head of a rule, there is an atom $\beta$ in the rule body that contains all variables from $\alpha$.

An additional contribution of Section 3 is to analyze the relative succinctness of OBDA query languages and equi-expressive versions of datalog. We first argue that $(\mathcal{ALC}, \text{UCQ})$ is exponentially more succinct than MDDlog, with the lower bound being conditional on the assumption from complexity theory that ExpTime $\not\subseteq$ coNP/Poly. We then prove that $(\mathcal{ALCI}, \text{UCQ})$, with $\mathcal{ALCI}$ the extension of $\mathcal{ALC}$ with inverse roles, is at least exponentially more succinct than MDDlog (without any complexity-theoretic assumptions) and at most double exponentially more succinct. This latter result extends from $(\mathcal{ALCI}, \text{UCQ})$ to (UNFO, UCQ). Actually, we show that a single-exponential succinctness gap can already be observed between $(\mathcal{ALC}, \text{UCQ})$ and $(\mathcal{ALCI}, \text{UCQ})$, and leave open whether the additional exponential blowup encountered in our translation of $(\mathcal{ALCI}, \text{UCQ})$ to MDDlog can be avoided (we conjecture this is not the case). We also show that, in contrast to inverse roles, several other standard extensions of $\mathcal{ALC}$ do not seem to have an impact on succinctness. Regarding other fragments of FO, we establish that (GNFO, UCQ) is at least exponentially more succinct than frontier-guarded DDlog and at most double exponentially more succinct. The case of (GFO, UCQ) is a bit different since our translation from frontier-guarded DDlog into (GFO, UCQ) involves an exponential blowup, whereas this direction is polynomial in all other cases.

In Section 4, we study ontology-mediated queries from a *descriptive complexity* perspective. In particular, we establish an intimate connection between OBDA query languages, constraint satisfaction problems, and MMSNP. Recall that constraint satisfaction problems (CSPs) form a subclass of the complexity class NP that, although it contains NP-hard problems, is in certain ways more computationally well behaved.

The widely known Feder-Vardi conjecture [Feder and Vardi 1998] states there is a dichotomy between PTIME and NP for the class of all CSPs, that is, each CSP is either in PTIME or NP-hard. The conjecture thus asserts that there are no CSPs that are NP-intermediate in the sense of Ladner's theorem. Monotone monadic strict NP without inequality (abbreviated MMSNP) was introduced by Feder and Vardi as a logical generalization of CSP that enjoys similar computational properties [1998]. In particular, it was shown in Feder and Vardi [1998] and Kun [2007] that there is a dichotomy between PTIME and NP for MMSNP sentences if and only if the Feder-Vardi conjecture holds.

In Section 4, we first observe that ($\mathcal{ALC}$, UCQ) and many other OBDA languages based on UCQs have the same expressive power as the query language coMMSNP, which consists of all queries whose complement is definable by an MMSNP formula with free variables. In the spirit of descriptive complexity theory, we say ($\mathcal{ALC}$, UCQ) *captures* coMMSNP. In fact, this result is a consequence of those in Section 3 and the observation that MDDlog has the same expressive power as coMMSNP.

To establish a counterpart of (GFO, UCQ) and (GNFO, UCQ) in the MMSNP world, we introduce guarded monotone strict NP (abbreviated GMSNP) as a generalization of MMSNP; specifically, GMSNP is obtained from MMSNP by allowing guarded second-order quantification in the place of monadic second-order quantification, similarly as in the transition from MDDlog to frontier-guarded disjunctive datalog. The resulting query language coGMSNP has the same expressive power as frontier-guarded disjunctive datalog and therefore, in particular, (GFO, UCQ) and (GNFO, UCQ) capture coGMSNP. We observe that GMSNP has the same expressive power as the extension $\text{MMSNP}_2$ of MMSNP proposed in Madelaine [2009]. It follows from our results in Section 3 that GMSNP (and thus $\text{MMSNP}_2$) is strictly more expressive than MMSNP, closing an open problem from Madelaine [2009].

In the second part of Section 4, we consider OBDA languages based on atomic queries and establish a tight connection to (certain generalizations of) CSPs. This connection is most easily stated for *Boolean* atomic queries (BAQs), which take the form $\exists x\, A(x)$ with $A$ a unary relation symbol: we prove ($\mathcal{ALC}$, BAQ) captures the query language that consists of all Boolean queries definable as the complement of a CSP. Similarly, we show that ($\mathcal{ALCU}$, AQ), with $\mathcal{ALCU}$ the extension of $\mathcal{ALC}$ with the universal role, and where queries are unary rather than Boolean, captures the query language that consists of all unary queries definable as the complement of a *generalized CSP*, which is given by a finite collection of structures (instead of a single one) enriched in a certain way with a constant symbol.

The results of Section 4 have fundamental consequences for ontology-based data access. In fact, significant progress has been made in understanding CSPs and MMSNP formulas [Bulatov 2011; Bodirsky et al. 2012; Kun and Nesetril 2008], and the connection established in Section 4 enables the transfer of techniques and results from CSP and MMSNP to OBDA. This is investigated in Section 5, where we consider three applications of the results in Section 4.

We first consider the data complexity of query evaluation for OBDA languages. Ideally, one would like to classify the data complexity of every ontology-mediated query within a given OBDA language such as ($\mathcal{ALC}$, UCQ). Our aforementioned results tie this task to proving the Feder-Vardi conjecture. We obtain that there is a dichotomy between PTIME and CONP for ontology-mediated queries from ($\mathcal{ALC}$, UCQ) if and only if the Feder-Vardi conjecture holds, and similarly for many other OBDA languages based on UCQs. Additionally, we obtain the same result for ontology-mediated query languages based on atomic queries such as ($\mathcal{ALC}$, BAQ) and ($\mathcal{ALC}$, AQ). This even holds for ontology-mediated query languages based on very expressive descriptions logics such as ($\mathcal{SHIU}$, BAQ) and ($\mathcal{SHIU}$, AQ). We also consider the standard extension

$\mathcal{ALCF}$ of $\mathcal{ALC}$ with functional roles and note that, for query evaluation in $(\mathcal{ALCF}, \text{AQ})$, there is no dichotomy between PTIME and coNP unless PTIME = NP.

The second application of the connection between OBDA and MMSNP/CSP concerns query containment in a rather general form as recently introduced and studied in Bienvenu et al. [2012]. It was shown in Feder and Vardi [1998] that containment between MMSNP sentences is decidable. We use this result to prove that query containment is decidable for many OBDA languages based on UCQs, including $(\mathcal{ALC}, \text{UCQ})$ and $(\text{GFO}, \text{UCQ})$. For many OBDA languages based on atomic queries such as $(\mathcal{ALC}, \text{AQ})$, we additionally pinpoint the exact computational complexity of query containment as NEXPTIME-complete. The upper bound is obtained by transferring the easy to obtain result that containment between CSP problems is in NP. The lower bound is established by reduction of a NEXPTIME-complete tiling problem. We also show that, for $(\mathcal{ALCF}, \text{AQ})$, the query containment problem is undecidable.

As the third application, we consider FO rewritability and datalog rewritability of ontology-mediated queries. Taking advantage of recent results for CSPs [Larose et al. 2007; Freese et al. 2009; Bulatov 2009], we are able to show that FO rewritability and datalog rewritability, as properties of ontology-mediated queries, are decidable and NEXPTIME-complete for $(\mathcal{ALC}, \text{AQ})$ and $(\mathcal{ALC}, \text{BAQ})$. This result extends to several extensions of $\mathcal{ALC}$. For $(\mathcal{ALCF}, \text{AQ})$ both problems again turn out to be undecidable.

In Section 6, we consider the case where the data schema is not fixed in advance. In contrast to the setup assumed in the previous sections, it is thus not possible to disallow any relation symbol from occurring in the data. This case is natural in many OBDA applications, where the data is not under the control of the user. We show that all decidability and complexity results obtained in the previous sections also hold in the schema-free case. In particular, this is true for query containment, FO rewritability, and datalog rewritability. We also show that, for all OBDA languages considered in this article, there is a dichotomy between PTIME and coNP in the schema-free case if and only if there is such a dichotomy in the fixed schema case. Via the results from Sections 4 and 5, this yields a connection to the Feder-Vardi conjecture also for the schema-free case.

Omitted proofs and additional proof details can be found in the electronic appendix accessible in the ACM Digital Library.

*Related Work.* A connection between query answering in DLs and the negation-free fragment of disjunctive datalog was first discovered in the influential papers Motik [2006] and Hustadt et al. [2007], where it was used to obtain a resolution calculus for reasoning about DL ontologies and to answer atomic queries (AQs) in the presence of such ontologies. A different approach to reasoning about DL ontologies that also involves disjunctive datalog can be found in Rudolph et al. [2012]. In contrast to the current article, these previous works do not consider the expressive power of ontology-mediated queries nor their succinctness and descriptive complexity. A connection between DL-based OBDA and CSPs was first found and exploited in Lutz and Wolter [2012], in a setup different from the one studied in this article. In particular, instead of focusing on ontology-mediated queries that consist of a data schema, an ontology, and a database query, the work reported in Lutz and Wolter [2012] concentrates on ontologies while quantifying universally over all database queries and without fixing a data schema. It establishes links to the Feder-Vardi conjecture that are incomparable to the ones found in this article, and does not consider the expressive power, succinctness, or descriptive complexity of queries used in OBDA. To the best of our knowledge, a connection between OBDA and MMSNP has not been established before. Existing work that is related to our results on query containment, FO rewritability, and datalog rewritability in DL-based OBDA is discussed in the main body of the article.

This article is based on the conference paper Bienvenu et al. [2013b]. It adds succinctness considerations, the study of schema-free ontology-mediated queries, and detailed proofs.

## 2. PRELIMINARIES

*Schemas, Instances, and Queries.* A *schema* is a finite collection $\mathbf{S} = (S_1, \ldots, S_k)$ of relation symbols with associated arity. A *fact* over $\mathbf{S}$ is an expression of the form $S(a_1, \ldots, a_n)$, where $S \in \mathbf{S}$ is an $n$-ary relation symbol, and $a_1, \ldots, a_n$ are elements of some fixed, countably infinite set $\mathsf{const}$ of *constants*. An *instance* $\mathfrak{D}$ over $\mathbf{S}$ is a finite set of facts over $\mathbf{S}$. The *active domain* $\mathsf{adom}(\mathfrak{D})$ of $\mathfrak{D}$ is the set of all constants that occur in the facts in $\mathfrak{D}$. We will frequently use boldface notation for tuples, such as in $\mathbf{a} = (a_1, \ldots, a_n)$, and we denote the empty tuple by $()$.

A *query over* $\mathbf{S}$ is semantically defined as a mapping $q$ that associates with every instance $\mathfrak{D}$ over $\mathbf{S}$ a set of *answers* $q(\mathfrak{D}) \subseteq \mathsf{adom}(\mathfrak{D})^n$, where $n \geq 0$ is the *arity* of $q$. If $n = 0$, then we say that $q$ is a *Boolean query*, and we write $q(\mathfrak{D}) = 1$ if $() \in q(\mathfrak{D})$ and $q(\mathfrak{D}) = 0$ otherwise.

A prominent way of specifying queries is by means of first-order logic (FO). Specifically, each schema $\mathbf{S}$ and domain-independent FO formula $\varphi(x_1, \ldots, x_n)$ that uses only relation symbols from $\mathbf{S}$ (and possibly equality) give rise to the $n$-ary query $q_{\varphi,\mathbf{S}}$, defined by setting, for all instances $\mathfrak{D}$ over $\mathbf{S}$,

$$q_{\varphi,\mathbf{S}}(\mathfrak{D}) = \{(a_1, \ldots, a_n) \in \mathsf{adom}(\mathfrak{D})^n \mid \mathfrak{D} \models \varphi[a_1, \ldots, a_n]\}.$$

To simplify the exposition, we assume FO queries do not contain constants. The free variables of an FO query are called *answer variables*. We use FOQ to denote the set of all first-order queries as defined before. Similarly, we use CQ to refer to the class of *conjunctive queries*, that is, FOQs of the form $\exists \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$, where $\varphi$ is a conjunction of relational atoms with the relation potentially being equality. UCQ refers to the class of *unions of conjunctive queries*, that is, disjunctions of CQs with the same answer variables. Finally, AQ denotes the set of *atomic queries*, which are CQs of the very simple form $A(x)$ with $A$ a unary relation symbol. Each of these is called a *query language* that is defined abstractly as a set of queries. Besides FOQ, CQ, UCQ, and AQ, we consider various other query languages that are introduced later, including ontology-mediated ones and variants of datalog.

Two queries $q_1$ and $q_2$ over $\mathbf{S}$ are *equivalent*, written $q_1 \equiv q_2$, if, for every instance $\mathfrak{D}$ over $\mathbf{S}$, we have $q_1(\mathfrak{D}) = q_2(\mathfrak{D})$. We say that query language $\mathcal{Q}_2$ is *at least as expressive as* query language $\mathcal{Q}_1$, written $\mathcal{Q}_1 \preceq \mathcal{Q}_2$, if, for every query $q_1 \in \mathcal{Q}_1$ over some schema $\mathbf{S}$, there is a query $q_2 \in \mathcal{Q}_2$ over $\mathbf{S}$ with $q_1 \equiv q_2$; $\mathcal{Q}_1$ and $\mathcal{Q}_2$ *have the same expressive power* if $\mathcal{Q}_1 \preceq \mathcal{Q}_2 \preceq \mathcal{Q}_1$.

*Ontology-Mediated Queries.* We introduce the fundamentals of ontology-based data access. An *ontology language* $\mathcal{L}$ is a fragment of first-order logic (i.e., a set of FO sentences), and an *$\mathcal{L}$-ontology* $\mathcal{O}$ is a finite set of sentences from $\mathcal{L}$.[1] We introduce various concrete ontology languages throughout the article, including descriptions logics and the guarded fragment.

An *ontology-mediated query* over a schema $\mathbf{S}$ is a triple $(\mathbf{S}, \mathcal{O}, q)$, where $\mathcal{O}$ is an ontology and $q$ is a query over $\mathbf{S} \cup \mathsf{sig}(\mathcal{O})$, with $\mathsf{sig}(\mathcal{O})$ the set of relation symbols used in $\mathcal{O}$. Here, we call $\mathbf{S}$ the *data schema*. Note that the ontology can introduce symbols that are not in the data schema, which allows it to enrich the schema of the query $q$. Of course, we do not require that every relation symbol of the data schema actually

[1]Domain independence is not required. In fact, there are many ontology languages in which domain independence is not guaranteed. In contrast, FOQs should be domain independent to ensure the usual correspondence to relational algebra.

Table I. Example Ontology, Presented in (the guarded fragment of) First-Order Logic and the DL $\mathcal{ALC}$

$$\forall x(\ \exists y(\mathsf{HasFinding}(x,y) \wedge \mathsf{ErythemaMigrans}(y)) \rightarrow \exists y(\mathsf{HasDiagnosis}(x,y) \wedge \mathsf{LymeDisease}(y))\ )$$
$$\forall x(\ (\mathsf{LymeDisease}(x) \vee \mathsf{Listeriosis}(x)) \rightarrow \mathsf{BacterialInfection}(x)\ )$$
$$\forall x(\ \exists y.(\mathsf{HereditaryPredisposition}(y) \wedge \mathsf{HasParent}(x,y)) \rightarrow \mathsf{HereditaryPredisposition}(x))\ )$$

$$\exists \mathsf{HasFinding}.\mathsf{ErythemaMigrans} \sqsubseteq \exists \mathsf{HasDiagnosis}.\mathsf{LymeDisease}$$
$$\mathsf{LymeDisease} \sqcup \mathsf{Listeriosis} \sqsubseteq \mathsf{BacterialInfection}$$
$$\exists \mathsf{HasParent}.\mathsf{HereditaryPredisposition} \sqsubseteq \mathsf{HereditaryPredisposition}$$

occur in the ontology. We have explicitly included $\mathbf{S}$ in the specification of the ontology-mediated query to emphasize that the ontology-mediated query is one over $\mathbf{S}$-instances, even though $\mathcal{O}$ and $q$ might use additional relation symbols.

The semantics of an ontology-mediated query is given in terms of *certain answers*, defined next. A *finite relational structure* over a schema $\mathbf{S}$ is a pair $\mathfrak{B} = (\mathsf{dom}, \mathfrak{D})$, where $\mathsf{dom}$ is a nonempty finite set called the *domain* of $\mathfrak{B}$ and $\mathfrak{D}$ is an instance over $\mathbf{S}$ with $\mathsf{adom}(\mathfrak{D}) \subseteq \mathsf{dom}$. When $\mathbf{S}$ is understood, we use $\mathsf{Mod}(\mathcal{O})$ to denote the set of all *models* of $\mathcal{O}$, that is, all finite relational structures $\mathfrak{B}$ over $\mathbf{S} \cup \mathsf{sig}(\mathcal{O})$ such that $\mathfrak{B} \models \mathcal{O}$. Let $(\mathbf{S}, \mathcal{O}, q)$ be an ontology-mediated query with $q$ of arity $n$. The *certain answers to $q$ on an $\mathbf{S}$-instance $\mathfrak{D}$ given $\mathcal{O}$* is the set $\mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$ of tuples $\mathbf{a} \in \mathsf{adom}(\mathfrak{D})^n$ such that, for all $(\mathsf{dom}, \mathfrak{D}') \in \mathsf{Mod}(\mathcal{O})$ with $\mathfrak{D} \subseteq \mathfrak{D}'$ (that is, all models of $\mathcal{O}$ that extend $\mathfrak{D}$), we have $\mathbf{a} \in q(\mathfrak{D}')$.

An instance $\mathfrak{D}$ is called *consistent* with an ontology $\mathcal{O}$ if there exists a finite relational structure $(\mathsf{dom}, \mathfrak{D}')$ with $\mathfrak{D}' \supseteq \mathfrak{D}$ such that $(\mathsf{dom}, \mathfrak{D}') \in \mathsf{Mod}(\mathcal{O})$. Note that, if an $\mathbf{S}$-instance $\mathfrak{D}$ is not consistent with $\mathcal{O}$ and a query $q$ has arity $n$, then $\mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D}) = \mathsf{adom}(\mathfrak{D})^n$.

All ontology languages considered in this article enjoy finite controllability, meaning that finite relational structures can be replaced with unrestricted ones without changing the certain answers to unions of conjunctive queries [Baader et al. 2003; Bárány et al. 2010, 2012].

Every ontology-mediated query $Q = (\mathbf{S}, \mathcal{O}, q)$ can be semantically interpreted as a query $q_Q$ over $\mathbf{S}$ by setting $q_Q(\mathfrak{D}) = \mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$ for all $\mathbf{S}$-instances $\mathfrak{D}$. In other words, we jointly consider the ontology $\mathcal{O}$ and actual query $q$ to be an "overall query" $q_Q$. Taking this view one step further, each choice of an ontology language $\mathcal{L}$ and query language $\mathcal{Q}$ gives rise to a query language, denoted $(\mathcal{L}, \mathcal{Q})$, defined as the set of queries $q_{(\mathbf{S},\mathcal{O},q)}$ with $\mathbf{S}$ a schema, $\mathcal{O}$ an $\mathcal{L}$ ontology, and $q \in \mathcal{Q}$ a query over $\mathbf{S} \cup \mathsf{sig}(\mathcal{O})$. We refer to such query languages $(\mathcal{L}, \mathcal{Q})$ as *ontology-mediated query languages*, or *OBDA languages* for short.

*Example* 2.1. The upper half of Table I shows an ontology $\mathcal{O}$ formulated in the guarded fragment of FO. Consider the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q)$ with the following data schema and query.

$$\mathbf{S} = \{\mathsf{ErythemaMigrans}, \mathsf{LymeDisease}, \mathsf{Listeriosis}, \mathsf{HereditaryPredisposition}, \mathsf{HasFinding}, \mathsf{HasDiagnosis}, \mathsf{HasParent}\}$$
$$q(x) = \exists y(\mathsf{HasDiagnosis}(x,y) \wedge \mathsf{BacterialInfection}(y)).$$

For the instance $\mathfrak{D}$ over $\mathbf{S}$ that consists of the facts

$$\mathsf{HasFinding}(\mathsf{patient1}, \mathsf{jan12find1}) \qquad \mathsf{ErythemaMigrans}(\mathsf{jan12find1})$$
$$\mathsf{HasDiagnosis}(\mathsf{patient2}, \mathsf{may7diag2}) \qquad \mathsf{Listeriosis}(\mathsf{may7diag2})$$

we have $\mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D}) = \{\mathsf{patient1}, \mathsf{patient2}\}$.

Table II. First-Order Translation of $\mathcal{ALC}$ Concepts

| | |
|---|---|
| $\top^*(x) = \top$ | $(C \sqcap D)^*(x) = C^*(x) \wedge D^*(x)$ |
| $\bot^*(x) = \bot$ | $(C \sqcup D)^*(x) = C^*(x) \vee D^*(x)$ |
| $A^*(x) = A(x)$ | $(\exists R.C)^*(x) = \exists y\, R(x,y) \wedge C^*(y)$ |
| $(\neg C)^*(x) = \neg C^*(x)$ | $(\forall R.C)^*(x) = \forall y\, R(x,y) \rightarrow C^*(y)$ |

*Description Logics for Specifying Ontologies.* In description logic, schemas are generally restricted to relation symbols of arity one and two, called *concept names* and *role names*, respectively. Despite the potential presence of unary relations, we speak of *binary schemas*. We briefly review the basic description logic $\mathcal{ALC}$. Relevant extensions of $\mathcal{ALC}$ will be introduced later on in the article.

An *$\mathcal{ALC}$-concept* is formed according to the syntax rule

$$C, D ::= A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C,$$

where $A$ ranges over concept names and $R$ over role names. An *$\mathcal{ALC}$-ontology* $\mathcal{O}$ is a finite set of *concept inclusions* $C \sqsubseteq D$, with $C$ and $D$ $\mathcal{ALC}$-concepts. We define the semantics of $\mathcal{ALC}$-concepts by translation to FO formulas with one free variable, as shown in Table II. An $\mathcal{ALC}$-ontology $\mathcal{O}$ then translates into the set of FO sentences

$$\mathcal{O}^* = \{\forall x\, (C^*(x) \rightarrow D^*(x)) \mid C \sqsubseteq D \in \mathcal{O}\}.$$

In the lower half of Table I, we show the $\mathcal{ALC}$ version of the guarded fragment ontology displayed in the upper half. Note that, although the translation is equivalence preserving in this case, in general (and even on binary schemas), the guarded fragment is a more expressive ontology language than $\mathcal{ALC}$. For example, the guarded sentence $\forall x \forall y (R(x,y) \rightarrow S(x,x))$ stating that every individual with an $R$-successor is $S$-reflexive is not equivalent to any $\mathcal{ALC}$ ontology. Throughout the article, we do not explicitly distinguish between a DL ontology and its translation into FO.

We remark that, from a DL perspective, the previous definitions of instances and certain answers correspond to making the *standard name assumption* (SNA) in ABoxes, which in particular implies the *unique name assumption*. We make the SNA only to facilitate uniform presentation; the SNA is inessential for the results presented in this article since we do not consider query languages with inequality.

The following example provides some first intuition about how ontology-mediated queries based on description logics relate to more traditional query languages.

*Example* 2.2. Let $\mathcal{O}$ and $\mathbf{S}$ be as in Example 2.1. For $q_1(x) = \mathsf{BacterialInfection}(x)$, the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q_1)$ is equivalent to the union of conjunctive queries $\mathsf{LymeDisease}(x) \vee \mathsf{Listeriosis}(x)$. For $q_2(x) = \mathsf{HereditaryPredisposition}(x)$, the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q_2)$ is equivalent to the query defined by the datalog program

$$P(x) \leftarrow \mathsf{HereditaryPredisposition}(x) \quad \mathsf{goal}(x) \leftarrow P(x)$$
$$P(x) \leftarrow \mathsf{HasParent}(x,y) \wedge P(y)$$

but not to any first-order query.

Throughout the article, for any syntactic object $o$ we will use $|o|$ to denote the number of syntactic symbols used to write out $o$, and call $|o|$ the *size* of $o$. For example, the size $|q|$ of the conjunctive query $q$ from Example 2.1 is 15, including all parentheses and counting each relation name as one syntactic symbol. This also defines the size $|\mathcal{O}|$ of an ontology $\mathcal{O}$, the size $|Q|$ of an ontology-mediated query $Q = (\mathbf{S}, \mathcal{O}, q)$, and so on.

## 3. OBDA AND DISJUNCTIVE DATALOG

We show that, for many OBDA languages, there is a natural fragment of disjunctive datalog with exactly the same expressive power.

A *disjunctive datalog rule* $\rho$ has the form

$$S_1(\mathbf{x}_1) \vee \cdots \vee S_m(\mathbf{x}_m) \leftarrow R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$$

with $m \geq 0$ and $n > 0$. We refer to $S_1(\mathbf{x}_1) \vee \cdots \vee S_m(\mathbf{x}_m)$ as the *head* of $\rho$, and to $R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$ as the *body* of $\rho$. Every variable that occurs in the head of a rule $\rho$ is required to also occur in the body of $\rho$. Empty rule heads are denoted $\bot$. A *disjunctive datalog* (DDlog) *program* $\Pi$ is a finite set of disjunctive datalog rules with a selected *goal relation* $\mathsf{goal}$ that does not occur in rule bodies and only in *goal rules* of the form $\mathsf{goal}(\mathbf{x}) \leftarrow R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n)$. The *arity of* $\Pi$ is the arity of the $\mathsf{goal}$ relation. Relation symbols that occur in the head of at least one rule of $\Pi$ are *intensional* (IDB) *relations*, and all remaining relation symbols in $\Pi$ are *extensional* (*EDB*) *relations*. An $\mathbf{S}'$-instance, with $\mathbf{S}'$ the set of all (IDB and EDB) relation symbols in $\Pi$, is a *model* of $\Pi$ if it satisfies all rules in $\Pi$. We use $\mathsf{Mod}(\Pi)$ to denote the set of all models of $\Pi$.

Every DDlog program $\Pi$ of arity $n$ naturally defines an $n$-ary query $q_\Pi$ over the schema $\mathbf{S}$ that consists of the EDB relations of $\Pi$: for every instance $\mathfrak{D}$ over $\mathbf{S}$, we have

$$q_\Pi(\mathfrak{D}) = \{\mathbf{a} \in \mathsf{adom}(\mathfrak{D})^n \mid \mathsf{goal}(\mathbf{a}) \in \mathfrak{D}' \text{ for all } \mathfrak{D}' \in \mathsf{Mod}(\Pi) \text{ with } \mathfrak{D} \subseteq \mathfrak{D}'\}.$$

Note that the DDlog programs considered in this article are negation free. Restricted to this fragment, there is no difference between the different semantics of DDlog studied, for example in Eiter et al. [1997].

We use $\mathsf{adom}(x)$ in rule bodies as a shorthand for "$x$ is in the active domain of the EDB relations". Specifically, whenever we use $\mathsf{adom}$ in a rule of a DDlog program $\Pi$, we assume that $\mathsf{adom}$ is an IDB relation and that the program $\Pi$ includes all rules of the form $\mathsf{adom}(x) \leftarrow R(\mathbf{x})$, where $R$ is an EDB relation of $\Pi$ and $\mathbf{x}$ is a tuple of distinct variables that includes $x$.

For simplicity, we generally speak about equivalence of an ontology-mediated query $Q$ and a DDlog program $\Pi$, meaning equivalence of the queries $q_Q$ and $q_\Pi$.

A *monadic disjunctive datalog* (MDDlog) *program* is a DDlog program in which all IDB relations with the possible exception of $\mathsf{goal}$ are monadic. We use MDDlog to denote the query language that consists of all queries defined by an MDDlog program.

Note that, while the data complexity of query evaluation in MDDlog and in many OBDA languages is coNP-complete [Eiter et al. 1997],[2] there is a significant difference in combined complexity. For example, the evaluation of queries from $(\mathcal{ALC}, \text{AQ})$ is ExpTime-complete, and that of queries from $(\mathcal{ALCI}, \text{UCQ})$ is even 2ExpTime-complete regarding combined complexity [Lutz 2008]. The following theorem, which we prove here for the sake of completeness, shows that MDDlog has lower complexity. The lower bound proof was suggested to us by Thomas Eiter (please see Eiter et al. [2007] for closely related results).

Theorem 3.1. *Query evaluation in MDDlog is* $\Pi_2^p$*-complete regarding combined complexity. The lower bound holds already over binary schemas.*

Proof. The upper bound is immediate: given an MDDlog program $\Pi$, instance $\mathfrak{D}$, and candidate answer tuple $\mathbf{d}$, we can show $\mathbf{d} \notin q_\Pi(\mathfrak{D})$ by guessing an instance $\mathfrak{D}'$ with $\mathfrak{D} \subseteq \mathfrak{D}'$, $\mathsf{adom}(\mathfrak{D}') = \mathsf{adom}(\mathfrak{D})$, and $\mathsf{goal}(\mathbf{d}) \notin \mathfrak{D}'$, and verifying using an NP oracle that every rule in $\Pi$ is satisfied in $\mathfrak{D}'$.

[2] In the case of ontology-mediated queries, data complexity refers to the setup where both the ontology and the actual query are fixed.

For the lower bound, we give a reduction from 2QBF validity. Consider a 2QBF $\forall x_1 \dots x_m \exists y_1 \dots y_n \varphi$, where $\varphi$ is a 3CNF over clauses $c_1, \dots, c_k$. We create an MDDlog program $\Pi$ whose set of EDB relations consists of unary relations $C_1, \dots, C_k$ and binary relations $V_1, V_2, V_3$, and $\mathsf{start}$, and whose (monadic) IDB relations are $X_1, \dots, X_m$. For each clause $c_i$, we denote by $v_i^j$ $(1 \leq j \leq 3)$ the variable appearing in the $j$th literal of $c_i$, and we let $S_i$ denote the set of tuples in $\{0,1\}^3$ representing the seven truth assignments for $(v_i^1, v_i^2, v_i^3)$ that satisfy $c_i$. We encode $\varphi$ using the instance $\mathcal{D}_\varphi$ defined as follows:

$$\mathcal{D}_\varphi = \big\{C_i(a_i^b), V_1(a_i^b, b_1), V_2(a_i^b, b_2), V_3(a_i^b, b_3) \mid b = (b_1, b_2, b_3) \in S_i\big\} \cup \{\mathsf{start}(0,1)\}.$$

The program $\Pi$ consists of a set of rules that select a truth assignment for the universally quantified variables

$$X_i(u_0) \vee X_i(u_1) \leftarrow \mathsf{start}(u_0, u_1) \qquad 1 \leq i \leq m$$

and a goal rule to check whether the selected truth assignment can be extended to a model of $\varphi$:

$$\mathsf{goal}() \leftarrow \bigwedge_{1 \leq i \leq k} \big(C_i(z_i) \wedge V_1(z_i, v_i^1) \wedge V_2(z_i, v_i^2) \wedge V_3(z_i, v_i^3)\big) \wedge \bigwedge_{1 \leq \ell \leq m} X_\ell(x_\ell).$$

It is straightforward to show that $\forall x_1 \dots x_m \exists y_1 \dots y_n \varphi$ is valid iff $Q_\Pi(\mathfrak{D}_\varphi) = 1$. □

We now introduce a characterization of Boolean MDDlog based on colorings of instances and forbidden pattern problems in the style of Madelaine and Stewart [2007]; see also Kun and Nesetril [2008] and Bodirsky et al. [2012]. The forbidden patterns characterization will be used later in this section as a technical tool to prove nonexpressibility results. An extension of forbidden pattern problems will also be used later in the article to characterize non-Boolean MDDlog.

Let $\mathbf{S}$ be a schema and $\mathcal{C}$ a set of unary relation symbols (colors) $\{C_1, \dots, C_n\}$ that is disjoint from $\mathbf{S}$. A *$\mathcal{C}$-colored $\mathbf{S}$-instance* is an $\mathbf{S} \cup \mathcal{C}$-instance $\mathfrak{D}$ such that, for every $d \in \mathsf{adom}(\mathfrak{D})$, there is exactly one fact in $\mathfrak{D}$ of the form $C_i(d)$. An instance $\mathfrak{D}'$ is called a *$\mathcal{C}$-coloring* of an $\mathbf{S}$-instance $\mathfrak{D}$ if $\mathfrak{D}$ is the restriction of $\mathfrak{D}'$ to the schema $\mathbf{S}$. Given a set $\mathcal{F}$ of $\mathcal{C}$-colored $\mathbf{S}$-instances (called *forbidden patterns*), we define $\mathsf{Forb}(\mathcal{F})$ as the set of all $\mathbf{S}$-instances $\mathfrak{D}$ for which there exists a $\mathcal{C}$-coloring $\mathfrak{D}'$ of $\mathfrak{D}$ such that, for every $\mathfrak{F} \in \mathcal{F}$, there is no homomorphism from $\mathfrak{F}$ into $\mathfrak{D}'$ (written $\mathfrak{F} \not\to \mathfrak{D}'$). The forbidden patterns problem defined by $\mathcal{F}$ is to decide whether a given $\mathbf{S}$-instance belongs to $\mathsf{Forb}(\mathcal{F})$. We let FPP denote the set of all forbidden patterns problems and use coFPP to refer to the query language that consists of all Boolean queries $q_{\mathcal{F},\mathbf{S}}$ defined by setting

$$q_{\mathcal{F},\mathbf{S}}(\mathfrak{D}) = 1 \quad \text{iff} \quad \mathfrak{D} \notin \mathsf{Forb}(\mathcal{F}),$$

with $\mathcal{F}$ a set of $\mathcal{C}$-colored $\mathbf{S}$-instances.

It follows from a result in Madelaine and Stewart [2007] and Theorem 4.1 in Section 4 that coFPP queries correspond precisely to Boolean MDDLog queries. We provide a direct proof here as a warm-up to the other proofs in this section.

PROPOSITION 3.2. *coFPP and Boolean MDDlog have the same expressive power.*

PROOF. First consider a set $\mathcal{F}$ of $\{C_1, \dots, C_n\}$-colored $\mathbf{S}$-instances, and let $\Pi_\mathcal{F}$ be the MDDlog program that consists of the following rules:

$$\begin{aligned} C_1(x) \vee \dots \vee C_n(x) &\leftarrow \mathsf{adom}(x) \\ \bot &\leftarrow C_i(x) \wedge C_j(x) && \text{for } 1 \leq i < j \leq n \\ \mathsf{goal}() &\leftarrow \varphi_\mathfrak{D} && \text{for } \mathfrak{D} \in \mathcal{F} \end{aligned}$$

where $\varphi_{\mathfrak{D}}$ is obtained by taking the conjunction of facts in $\mathfrak{D}$ and treating the constants as variables. It can be verified that the queries $q_{\Pi,\mathbf{S}}$ and $q_{\mathcal{F},\mathbf{S}}$ are equivalent. Indeed, the first two types of rules generate all possible colorings of a given instance, and the goal rules check for the presence of a forbidden pattern.

Next consider an MDDlog program $\Pi$ whose set of EDB relations is $\mathbf{S}$ and whose set of non-goal IDB relations is $\mathbf{P}$. We may suppose without loss of generality that there is no rule in $\Pi$ whose head and body contain the same atom (such rules are tautologous and can be removed). Define the set of colors $\mathcal{C} = \{C_T \mid T \subseteq \mathbf{P}\}$, and let $\mathcal{F}$ be the set of all $\mathcal{C}$-colored $\mathbf{S}$-instances that can be obtained from a rule $\rho$ in $\Pi$ by:

(1) taking all facts obtained from an EDB atom in the body of $\rho$ by replacing each variable $x$ by a distinct constant $d_x$;
(2) adding a single fact $C_{T_x}(d_x)$ for each variable $x$, where the subset $T_x \subseteq \mathbf{P}$ is chosen such that it contains every IDB relation $R$ for which $R(x)$ appears in the body of $\rho$ and omits $R$ if the atom $R(x)$ appears in the head of $\rho$.

Intuitively, the forbidden patterns in $\mathcal{F}$ that are obtained from goal rules check for the satisfaction of the body of a goal rule, whereas those derived from non-goal rules check for the violation of such rules. Therefore, an $\mathbf{S}$-instance $\mathfrak{D}$ belongs to $\mathsf{Forb}(\mathcal{F})$ just in the case that there is a model of $\Pi$ and $\mathfrak{D}$ in which the goal relation is not derived. The equivalence of $q_{\mathcal{F},\mathbf{S}}$ and $q_{\Pi,\mathbf{S}}$ follows immediately. □

### 3.1. Ontologies Specified in Description Logics

We show that $(\mathcal{ALC}, \text{UCQ})$ has the same expressive power as MDDlog and identify a fragment of MDDlog that has the same expressive power as $(\mathcal{ALC}, \text{AQ})$. While in both cases the translation from MDDlog into ontology-mediated queries is linear, the backwards translations incur an exponential blowup (refer to the proof of Proposition 3.2 given before). The different combined complexity of query evaluation in $(\mathcal{ALC}, \text{UCQ})$ and MDDlog queries provides a first indication that this might be unavoidable. To give more concrete evidence, we argue that, unless ExpTime ⊆ coNP/Poly, an exponential blowup is indeed unavoidable. We additionally consider the extensions of $\mathcal{ALC}$ with inverse roles, role hierarchies, transitive roles, and the universal role, which we also relate to MDDlog and its fragments. To match the syntax of $\mathcal{ALC}$ and its extensions, we generally assume schemas to be binary throughout this section.[3]

*$(\mathcal{ALC}, \text{UCQ})$ and $(\mathcal{ALC}, \text{AQ})$.* The first main result of this section is the following.

Theorem 3.3. *$(\mathcal{ALC}, \text{UCQ})$ and MDDlog have the same expressive power. In fact:*

(1) *there is a polynomial $p$ such that, for every query $(\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALC}, \text{UCQ})$, there is an equivalent MDDlog program $\Pi$ with $|\Pi| \leq 2^{p(|\mathcal{O}|+|q|)}$;*
(2) *for every MDDlog program $\Pi$, there is an equivalent query $(\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALC}, \text{UCQ})$ with $|q| \in O(|\Pi|)$ and $|\mathcal{O}| \in O(|\Pi|)$.*

Proof. We start by proving Point 1. Thus, let $Q = (\mathbf{S}, \mathcal{O}, q)$ be an ontology-mediated query from $(\mathcal{ALC}, \text{UCQ})$. We first give some intuitions about answering $Q$ that guide our translation into an equivalent MDDlog program $\Pi$.

The definition of certain answers to $Q$ on an instance $\mathfrak{D}$ involves a quantification over all models of $\mathcal{O}$ that extend $\mathfrak{D}$. It turns out that, in the case of $(\mathcal{ALC}, \text{UCQ})$ queries, it suffices to consider a particular type of extensions of $\mathfrak{D}$ that we term *forest extension*. Intuitively, such an extension of $\mathfrak{D}$ corresponds to attaching tree-shaped structures to the elements of $\mathfrak{D}$. Formally, a relational structure $(\mathsf{dom}, \mathfrak{B})$ over a binary schema is

[3]In fact, this assumption is inessential for Theorems 3.3 and 3.6 (that speak about UCQs), but required for Theorems 3.4, 3.11, and 3.12 (that speak about AQs) to hold.

*tree shaped* if the directed graph $(\mathsf{dom}, \{(a,b) \mid R(a,b) \in \mathfrak{B} \text{ for some } R\})$ is a tree and there do not exist facts $R(a,b), S(a,b) \in \mathfrak{B}$ with $R \neq S$. A relational structure $\mathfrak{D}'$ is a forest extension of an instance $\mathfrak{D}$ if $\mathfrak{D} \subseteq \mathfrak{D}'$ and $\mathfrak{D}' \setminus \mathfrak{D}$ is a union of tree-shaped instances $\{\mathfrak{D}'_a \mid a \in \mathsf{adom}(\mathfrak{D})\}$ such that:

—$\mathsf{adom}(\mathfrak{D}'_a) \cap \mathsf{adom}(\mathfrak{D}) = \{a\}$ with $a$ the root of $\mathfrak{D}'_a$ and
—$\mathsf{adom}(\mathfrak{D}'_a) \cap \mathsf{adom}(\mathfrak{D}'_b) = \emptyset$ for $a \neq b$.

The fact that we need only consider models of $\mathcal{O}$ that are forest extensions of $\mathfrak{D}$ is helpful because it constrains the ways in which a CQ can be satisfied. Specifically, every homomorphism $h$ from $q$ to $\mathfrak{D}'$ gives rise to a decomposition of $q$ into a collection of components $q_0, \ldots, q_k$, where:

(i) the *core component* $q_0$ comprises all atoms of $q$ whose variables are sent by $h$ to elements of $\mathfrak{D}$, and
(ii) for each $\mathfrak{D}'_a$ in the image of $h$, there is a *noncore component* $q_i$, $1 \leq i \leq k$, that comprises all atoms of $q$ whose variables are sent by $h$ to elements of $\mathfrak{D}'_a$.

Note that the noncore components are pairwise variable-disjoint and share at most one variable with the core component. Also note that each noncore component $q_i$ is a homomorphic preimage of a tree. In other words, $q_i$ can be converted into a tree by exhaustively *eliminating forks*, that is, for all atoms $R(y_1, x), R(y_2, x) \in q_i$ with $y_1 \neq y_2$, identifying $y_1$ with $y_2$.

We now detail the translation of $Q$ into the MDDlog program $\Pi$. Let $\mathsf{sub}(\mathcal{O})$ be the set of subconcepts (that is, syntactic subexpressions) of concepts that occur in $\mathcal{O}$. For example, if we take the ontology $\mathcal{O}$ consisting of a single inclusion $\forall R.\exists S.\neg B \sqsubseteq A \sqcup D$, then $\mathsf{sub}(\mathcal{O})$ would contain the following concepts: $\forall R.\exists S.\neg B$, $\exists S.\neg B$, $\neg B$, $B$, $A \sqcup D$, $A$, $D$. Next, we let $\mathsf{tree}(q)$ denote the set of all CQs that can be obtained from a CQ $q'$ in the UCQ $q$ in the following way:

(1) perform exhaustive fork elimination in $q'$, resulting in a CQ $\widehat{q}'$;
(2) include in $\mathsf{tree}(q)$ any connected component of $\widehat{q}'$ that is tree shaped and has no answer variable;
(3) for every atom $R(x,y) \in \widehat{q}'$ such that the restriction $\widehat{q}'|_y$ of $\widehat{q}'$ to those variables that are reachable from $y$ (in $\widehat{q}'$ viewed as a directed graph) is tree shaped and has no answer variable, include $\{R(x,y)\} \cup \widehat{q}'|_y$ in $\mathsf{tree}(q)$, with $x$ as the only answer variable.

To illustrate the construction, suppose that the UCQ $q$ contains the following CQ $q'$.

$$\exists y_1 \cdots \exists y_8 \; P(y_1,y_2) \wedge S(y_1,y_3) \wedge R(y_2,y_4) \wedge R(y_3,y_4) \wedge S(y_4,y_5) \wedge R(y_6,y_7) \wedge S(y_6,y_8)$$

In step (1), we unify $y_2$ and $y_3$, leading to the query $\widehat{q}'$ with $y_3$ replaced by $y_2$. In step (2), we include in $\mathsf{tree}(q)$ the query $\exists y_6 \exists y_7 \exists y_8 \; R(y_6,y_7) \wedge S(y_6,y_8)$. Note that the other connected component of $\widehat{q}'$ is not included as it is not tree shaped due to the presence of atoms $P(y_1,y_2)$ and $S(y_1,y_2)$. In step (3), we add four additional queries: $\exists y_4 \exists y_5 \; R(y_2,y_4) \wedge S(y_4,y_5)$, $\exists y_5 \; S(y_4,y_5)$, $\exists y_7 \; R(y_6,y_7)$, and $\exists y_8 \; S(y_6,y_8)$.

It can be shown that the number of queries in $\mathsf{tree}(q)$ is bounded by $|q|$; see Lutz [2008] for full details. Moreover, we clearly have $|p| \leq |q|$ for all $p \in \mathsf{tree}(q)$. Set $\mathsf{cl}(\mathcal{O}, q) = \mathsf{sub}(\mathcal{O}) \cup \mathsf{tree}(q)$. A *type* (for $\mathcal{O}$ and $q$) is a subset of $\mathsf{cl}(\mathcal{O}, q)$. The CQs in $\mathsf{tree}(q)$ include all potential noncore components from the intuitive explanation given earlier. The answer variable of such a CQ (if any) represents the overlap between the core component and the noncore component.

We introduce a fresh unary relation symbol $P_\tau$ for every type $\tau$, and we denote by $\mathbf{S}'$ the schema that extends $\mathbf{S}$ with these additional symbols. In the MDDlog program

that we aim to construct, the relation symbols $P_\tau$ are used as IDB relations, and the symbols from $\mathbf{S}$ are the EDB relations.

We say that a relational structure $\mathfrak{B}$ over $\mathbf{S}' \cup \mathsf{sig}(\mathcal{O})$ is *type coherent* if $P_\tau(d) \in \mathfrak{B}$ just in the case that

$$\begin{aligned}\tau \;=\; & \{\varphi \in \mathsf{cl}(\mathcal{O}, q) \mid \varphi \text{ Boolean }, \mathfrak{B} \models \varphi\} \\ & \cup \{\varphi \in \mathsf{cl}(\mathcal{O}, q) \mid \varphi \text{ has one free variable}, \mathfrak{B} \models \varphi[d]\}.\end{aligned}$$

Set $k$ equal to the maximum of 2 and the width of $q$, that is, the number of variables that occur in $q$. By a *diagram*, we mean a conjunction $\delta(x_1, \ldots, x_n)$ of atomic formulas over the schema $\mathbf{S}'$, with $n \leq k$ variables. A diagram $\delta(\mathbf{x})$ is *realizable* if there exists a type-coherent $\mathfrak{B} \in \mathsf{Mod}(\mathcal{O})$ that satisfies $\exists \mathbf{x} \delta(\mathbf{x})$. A diagram $\delta(\mathbf{x})$ *implies* $q(\mathbf{x}')$, with $\mathbf{x}'$ a sequence of variables from $\mathbf{x}$, if every type-coherent $\mathfrak{B} \in \mathsf{Mod}(\mathcal{O})$ that satisfies $\delta(\mathbf{x})$ under some variable assignment satisfies $q(\mathbf{x}')$ under the same assignment.

The desired MDDlog program $\Pi$ consists of the following collections of rules.

$$\begin{aligned}\bigvee P_\tau(x) &\leftarrow \mathsf{adom}(x) && \\ \tau \subseteq \mathsf{cl}(\mathcal{O},q) \quad \bot &\leftarrow \delta(\mathbf{x}) && \text{for all non-realizable diagrams } \delta(\mathbf{x}) \\ \mathsf{goal}(\mathbf{x}') &\leftarrow \delta(\mathbf{x}) && \text{for all diagrams } \delta(\mathbf{x}) \text{ that imply } q(\mathbf{x}')\end{aligned}$$

Intuitively, these rules "guess" a (representation of a) forest extension $\mathfrak{D}'$ of $\mathfrak{D}$. Specifically, the types $P_\tau$ guessed in the first line determine which subconcepts of $\mathcal{O}$ are made true at each element of $\mathfrak{D}$. Since MDDlog does not support existential quantifiers, the $\mathfrak{D}'_a$ parts of $\mathfrak{D}'$ cannot be guessed explicitly. Instead, the CQs included in the guessed types $P_\tau$ determine those noncore component queries that homomorphically map into the $\mathfrak{D}'_a$ parts. The second line ensures coherence of the guesses and the last line guarantees that $q$ has the required homomorphism into $\mathfrak{D}'$. It is proved in the electronic appendix that $\Pi$ is indeed equivalent to $\mathcal{Q}$.

For the converse direction, let $\Pi$ be an MDDlog program that defines an $n$-ary query ($n \geq 0$). For each unary IDB relation $A$ of $\Pi$, we introduce two fresh unary relation symbols, denoted by $A$ and $\bar{A}$. The ontology $\mathcal{O}$ enforces that $\bar{A}$ represents the complement of $A$, that is, it consists of all inclusions of the form

$$\top \sqsubseteq (A \sqcup \bar{A}) \sqcap \neg(A \sqcap \bar{A}).$$

In addition, the ontology contains the inclusion $\top \sqsubseteq D$, for a fresh unary relation symbol $D$, enforcing that $D$ denotes the entire domain.

Let $q$ be the union of: (i) all $n$-ary conjunctive queries that constitute the body of a goal rule, as well as (ii) all $n$-ary conjunctive queries obtained from a non-goal rule of the form

$$A_1(\mathbf{x}_1) \vee \cdots \vee A_m(\mathbf{x}_m) \leftarrow R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$$

by taking the conjunctive query $q'(z_1, \ldots, z_n)$ defined by

$$\bar{A}_1(\mathbf{x}_1) \wedge \cdots \wedge \bar{A}_m(\mathbf{x}_m) \wedge R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n) \wedge D(z_1) \wedge \cdots \wedge D(z_n).$$

It can be shown that the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S}$ is the schema that consists of the EDB relations of $\Pi$, is equivalent to the query defined by $\Pi$. Indeed, if $\mathfrak{D}$ is an $\mathbf{S}$-instance, then every model of $\mathcal{O}$ and $\mathfrak{D}$ corresponds to an instance $\mathfrak{D}' \supseteq \mathfrak{D}$ over the schema consisting of the EDB relations and non-goal IDB relations of $\Pi$. Queries of type (ii) ensure that $q$ is trivially satisfied (i.e., returns all $n$-tuples of constants) in all models whose corresponding instance $\mathfrak{D}'$ violates some non-goal rule in $\Pi$. All other models of $\mathcal{O}$ and $\mathfrak{D}$ correspond to models $\mathfrak{D}' \supseteq \mathfrak{D}$ of the non-goal rules in $\Pi$, and for these we use queries of type (i) to identify those tuples of constants that must belong to the goal relation. □

Next, we characterize $(\mathcal{ALC}, \text{AQ})$ by a fragment of MDDlog. While atomic queries are regularly used in ontology-mediated queries, they also occur in several other forms. In particular, $(\mathcal{ALC}, \text{AQ})$ has the same expressive power as the OBDA language $(\mathcal{ALC}, \text{ConQ})$, where ConQ denotes the set of all *$\mathcal{ALC}$-concept queries*, that is, queries $C(x)$ with $C$ a (possibly compound) $\mathcal{ALC}$-concept. Specifically, each query $(\mathbf{S}, \mathcal{O}, q) \in (\mathcal{ALC}, \text{ConQ})$ with $q = C(x)$ can be expressed as a query $(\mathbf{S}, \mathcal{O}', A(x)) \in (\mathcal{ALC}, \text{AQ})$, where $A$ is a fresh concept name (that is, it does not occur in $\mathbf{S} \cup \mathsf{sig}(\mathcal{O})$) and $\mathcal{O}' = \mathcal{O} \cup \{C \sqsubseteq A\}$. As a consequence, $(\mathcal{ALC}, \text{AQ})$ also has the same expressive power as $(\mathcal{ALC}, \text{TCQ})$, where TCQ is the set of all CQs that take the form of a directed tree with a single answer variable at the root.

Each disjunctive datalog rule can be associated with an undirected graph whose nodes are the variables that occur in the rule and whose edges reflect co-occurrence of two variables in an atom in the rule body. We say a rule is *connected* if its graph is connected, and that a DDlog program is connected if all its rules are connected. An MDDlog program is *simple* if each rule contains at most one atom $R(\mathbf{x})$ with $R$ an EDB relation; additionally, we require that in this atom every variable occurs at most once. An MDDlog program is *unary* if its goal relation is unary.

THEOREM 3.4. *$(\mathcal{ALC}, \text{AQ})$ has the same expressive power as unary connected simple MDDlog. In fact:*

(1) *there is a polynomial $p$ such that, for every query $(\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALC}, \text{AQ})$, there is an equivalent unary connected simple MDDlog program $\Pi$ such that $|\Pi| \leq 2^{p(|\mathcal{O}|)}$;*
(2) *for every MDDlog program $\Pi$, there is an equivalent query $(\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALC}, \text{AQ})$ such that $|\mathcal{O}| \in O(|\Pi|)$.*

PROOF. Let $Q = (\mathbf{S}, \mathcal{O}, A_0(x))$ be a query from $(\mathcal{ALC}, \text{AQ})$. To define an equivalent MDDlog program $\Pi$, fix unary relation symbols $P_C$ and $P_{\neg C}$ for every $C \in \mathsf{sub}(\mathcal{O})$. Let $\mathbf{S}'$ be the schema that extends $\mathbf{S}$ with these symbols. A *type* $t(x)$ is a conjunction of atomic formulas of the form $P_D(x)$ such that, for each subconcept $C \in \mathsf{sub}(\mathcal{O})$, either $P_C(x)$ or $P_{\neg C}(x)$ occurs as a conjunct. A *diagram* is a conjunction $\delta(\mathbf{x})$ of atomic formulas over the schema $\mathbf{S}'$ that is of the form $t(x)$, $A(x) \wedge t(x)$ or $t_1(x) \wedge R(x, y) \wedge t_2(y)$, where $A$ and $R$ are from $\mathbf{S}$ and $t$, $t_1$, and $t_2$ are types. A diagram $\delta(\mathbf{x})$ is *realizable* if there is a $\mathfrak{B} \in \mathsf{Mod}(\mathcal{O})$ that satisfies $\exists \mathbf{x} \delta(\mathbf{x})$ and such that $P_C(a) \in \mathfrak{B}$ implies $\mathfrak{B} \models C[a]$. Now the MDDlog program $\Pi$ consists of the following rules.

$$\bigwedge_{C \in \mathsf{sub}(\mathcal{O})} (\, P_C(x) \vee P_{\neg C}(x) \,) \leftarrow \mathsf{adom}(x)$$
$$\bot \leftarrow \delta(\mathbf{x}) \qquad \text{for all non-realizable diagrams } \delta(\mathbf{x})$$
$$\mathsf{goal}(x) \leftarrow P_{A_0}(x)$$

Clearly, $\Pi$ is unary, connected, and simple. It can be shown as in the proof of Theorem 3.3 that $Q$ is equivalent to $\Pi$. Because of the atomic nature of AQs, though, the argument is substantially simpler. Details are omitted.

Conversely, let $\Pi$ be a unary connected simple MDDlog program. It is easy to rewrite each rule of $\Pi$ into an equivalent $\mathcal{ALC}$-concept inclusion, where goal is now regarded as a concept name. For example, $\mathsf{goal}(x) \leftarrow R(x, y)$ is rewritten into $\exists R.\top \sqsubseteq \mathsf{goal}$. Similarly, $\mathsf{goal}(x) \leftarrow R(y, z)$ is rewritten into $\top \sqsubseteq \forall R.\mathsf{goal}\}$, and $P_1(x) \vee P_2(y) \leftarrow R(x, y) \wedge P_3(x) \wedge P_4(y)$ is rewritten into $P_3 \sqcap \exists R.(P_4 \sqcap \neg P_2) \sqcap \neg P_1 \sqsubseteq \bot$. Let $\mathcal{O}$ be the resulting ontology and let $q = \mathsf{goal}(x)$. Then $\Pi$ is equivalent to the query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S}$ consists of the EDB relations in $\Pi$. □

Note that the connectedness condition is required since one cannot express MDDlog rules such as $\mathsf{goal}(x) \leftarrow A(x) \wedge B(y)$ in $(\mathcal{ALC}, \text{AQ})$. Multiple variable occurrences in EDB

relations have to be excluded because programs such as $\mathsf{goal}(x) \leftarrow A(x),\ \bot \leftarrow R(x,x)$ (return all elements in $A$ if the instance contains no reflexive $R$-edge, and return the active domain otherwise) also cannot be expressed in $(\mathcal{ALC}, \text{AQ})$.

We now observe that the blowup encountered in the translations from ontology-mediated queries to MDDlog in Theorems 3.3 and 3.4 can probably not be avoided.

THEOREM 3.5. *There is a family of queries $(Q_i)_{i\geq 0}$ from $(\mathcal{ALC}, AQ)$ with $|Q_i| \leq p(i)$ for all $i \geq 0$, $p$ a polynomial such that, unless* ExpTime $\subseteq$ coNP/Poly*, there is no family of MDDlog programs $(\Pi_i)_{i\geq 0}$ with:*

*(1)* $\Pi_i \equiv Q_i$ *for all* $i \geq 0$*;*
*(2)* $|\Pi_i| \leq p'(i)$ *for all* $i \geq 0$*, for some polynomial* $p'$*.*

PROOF. Let $M$ be a polynomially space-bounded alternating Turing machine (ATM) with input alphabet $\Sigma$ that solves an ExpTime-hard problem. The standard ExpTime-hardness proof for satisfiability in $\mathcal{ALC}$ shows that, for each $i \geq 0$, there is an ontology-mediated query $Q_i = (\mathbf{S}_i, \mathcal{O}_i, G_i)$ from $(\mathcal{ALC}, \text{AQ})$ with $\mathbf{S}_i = \{A_{j,\sigma} \mid j < i, \sigma \in \Sigma\}$, such that:

(i) the size of $Q_i$ is polynomial in $i$, and
(ii) on $\mathbf{S}_i$-instances of the form

$$\mathfrak{D}_w = \{S(a), A_{0,\sigma_0}(a), A_{1,\sigma_1}(a), \dots A_{n-1,\sigma_{n-1}}(a)\} \text{ where } w = \sigma_0 \cdots \sigma_{n-1} \in \Sigma^*,$$

we have $a \in Q_i(\mathfrak{D}_w)$ iff input $w$ is accepted by $M$.[4]

Note that MDDlog programs $\Pi$ can be evaluated (uniformly) in coNP on instances $\mathfrak{D}_w$: to check whether $\mathfrak{D}_w \not\models \Pi$, guess an extension of the IDBs that makes $\mathsf{goal}$ empty and check that all rules are satisfied. The latter can be done in polytime since checking homomorphisms into a singleton structure is trivial.

Assume, to the contrary of what we have to show, that there is a family of MDDLog programs $(\Pi_i)_{i\geq 0}$ that satisfies Properties (1) and (2) given before. Then the ExpTime-hard problem $L(M)$ can be solved in coNP/Poly: given an input $w \in \Sigma^*$ of length $n$, give $\Pi_i$ as an advice to the TM; the TM constructs $\mathfrak{D}_w$ and checks in coNP whether $\mathfrak{D}_w \models \Pi_i$. $\square$

*Extensions of $\mathcal{ALC}$.* We consider several OBDA languages that can be obtained from $(\mathcal{ALC}, \text{UCQ})$ and $(\mathcal{ALC}, \text{AQ})$ by replacing the $\mathcal{ALC}$ ontology language with one of its standard extensions and show that some of the resulting languages still have the same expressive power as MDDlog while others do not. We also analyze the succinctness of the extended OBDA languages. Note that all extensions of $\mathcal{ALC}$ considered in this section are fragments of the OWL2 DL profile of the W3C-standardized ontology language OWL2 [OWL Working Group 2009]. Some of the translations used in this section are folklore in the area of description logic, and when this is the case we usually confine ourselves to a proof sketch.

We review the relevant extensions of $\mathcal{ALC}$ only briefly and refer to Baader et al. [2003] for more details. $\mathcal{ALCI}$ is the extension of $\mathcal{ALC}$ with *inverse roles*, that is, with the operators $\exists R^-.C$ and $\forall R^-.C$ whose semantics is defined by the FO formulas $\exists y\, R(y,x) \wedge C^*(y)$ and $\forall y\, R(y,x) \rightarrow C^*(y)$, respectively (note the swap of $x$ and $y$ in the $R$-atom). $\mathcal{ALCH}$ extends $\mathcal{ALC}$ by admitting *role hierarchy* statements $R \sqsubseteq S$ in the ontology, where $R$ and $S$ are role names; the semantics of these statements is $\forall xy(R(x,y) \rightarrow S(x,y))$. $\mathcal{S}$ extends $\mathcal{ALC}$ by allowing *transitive role* statements $\mathsf{trans}(R)$ in the ontology, which require the role name $R$ to be interpreted as a transitive relation.

[4] $S$ stands for "start computation."

$\mathcal{ALCF}$ is the extension of $\mathcal{ALC}$ with *functional role* statements $\mathsf{func}(R)$ that require the role name $R$ to be interpreted as a partial function. Finally, $\mathcal{ALCU}$ is the extension with the *universal role* $U$ that is interpreted as the total relation $\mathsf{dom} \times \mathsf{dom}$ in any relational structure $\mathfrak{B}$ with domain $\mathsf{dom}$. Note that $U$ should be regarded as a logical symbol and is not a member of any schema.

We use the usual naming scheme to denote combinations of these extensions, for example, $\mathcal{ALCHI}$ for the extension of $\mathcal{ALC}$ with both inverse roles and role hierarchies, and $\mathcal{SHI}$ for the extension of $\mathcal{ALCHI}$ with transitive roles. In $\mathcal{ALCHI}$ and its extensions, the role hierarchy statements can also refer to inverse roles, as in $R \sqsubseteq S^-$, with the obvious semantics. The following result identifies a relevant extension of ($\mathcal{ALC}$, UCQ) that still has the same expressive power as MDDlog.

THEOREM 3.6. *($\mathcal{ALCHIU}$, UCQ) has the same expressive power as MDDlog. In fact, there is a polynomial $p$ such that:*

(1) *for every query $(\mathbf{S}, \mathcal{O}, q)$ from ($\mathcal{ALCHIU}$, UCQ), there is an equivalent query $Q'$ from ($\mathcal{ALCHU}$, UCQ) such that $|Q'| \leq p(|\mathcal{O}|) \cdot 2^{p(|q|)}$;*
(2) *for every query $(\mathbf{S}, \mathcal{O}, q)$ from ($\mathcal{ALCHU}$, UCQ), there is an equivalent MDDlog program $\Pi$ such that $|\Pi| \leq 2^{p(|\mathcal{O}|+|q|)}$.*

PROOF (SKETCH). To establish Point 1, we use a folklore technique for eliminating inverse roles [De Giacomo and Lenzerini 1994]. Let $Q = (\mathbf{S}, \mathcal{O}, q)$ be a query from ($\mathcal{ALCHIU}$, UCQ) and assume without loss of generality that:

(i) in concept inclusions, $\mathcal{O}$ uses only the operators $\neg$, $\sqcap$, and $\exists$, but neither $\sqcup$ nor $\forall$;
(ii) the role hierarchy statements in $\mathcal{O}$ are closed under inverse, that is, $R \sqsubseteq S \in \mathcal{O}$ implies $R^- \sqsubseteq S^- \in \mathcal{O}$, where $(R^-)^- := R$.

Introduce a fresh role name $R^{\mathsf{inv}}$ for every role name $R$ used in $\mathcal{O}$ (excluding the universal role $U$). These fresh symbols are not included in the schema $\mathbf{S}$. For each concept $C$ occurring in $\mathcal{O}$, let $C'$ be the concept obtained from $C$ by replacing every occurrence of an inverse role $R^-$ by $R^{\mathsf{inv}}$. The ontology $\mathcal{O}'$ consists of the following:

—all concept subsumptions $C' \sqsubseteq D'$ for $C \sqsubseteq D$ in $\mathcal{O}$;
—$C' \sqsubseteq \forall R^{\mathsf{inv}}.\exists R.C'$ for every existential restriction $\exists R.C$ in $\mathsf{sub}(\mathcal{O})$ with $R$ a role name;
—$C' \sqsubseteq \forall R.\exists R^{\mathsf{inv}}.C'$ for every existential restriction $\exists R^-.C$ in $\mathsf{sub}(\mathcal{O})$.

The UCQ $q'$ is obtained from $q$ by replacing every atom $R(x, y)$ with $R(x, y) \vee R^{\mathsf{inv}}(y, x)$ and then distributing conjunction over disjunction. It can be shown that the obtained query $Q' = (\mathbf{S}, \mathcal{O}', q')$ from ($\mathcal{ALCHU}$, UCQ) is equivalent to $Q$ and of the required size.

Point 2 can be established by a straightforward extension of the proof of Theorem 3.3 from ($\mathcal{ALC}$, UCQ) to ($\mathcal{ALCHU}$, UCQ), which requires almost no changes. □

Note that the elimination of inverse roles in Point 1 of Theorem 3.6 involves an exponential blowup. We now show that such a blowup is unavoidable.

THEOREM 3.7. *There is a family of queries $(Q_i)_{i\geq 0}$ from ($\mathcal{ALCI}$, UCQ) with $|Q_i| \leq p(i)$ for all $i \geq 0$, $p$ a polynomial such that there is no family of queries $(P_i)_{i\geq 0}$ from ($\mathcal{ALCHU}$, UCQ) with $P_i \equiv Q_i$ and $|P_i| < 2^{i/3}$ for all $i \geq 0$.*

PROOF. Without loss of generality, we can restrict our attention to a restricted class of instances. Fix the schema $\mathbf{S} = \{R, Y_0, Y_1, Y_2\}$ with $R$ a role name and $Y_0, Y_1, Y_2$ concept names. We use the composition $R^-; R$ of the inverse role $R^-$ with its base role $R$ to simulate a symmetric role, which is not available in the DLs considered here. The *counting instance of length $k$* is the $\mathbf{S}$-instance $\mathfrak{C}_k$ that consists of an $R^-; R$-path of length $k$, that is, a sequence of elements $a_0, \ldots, a_{2k}$ such that, for $0 < i < 2k$ with $i$ odd,

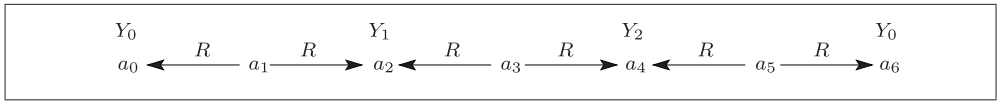



Fig. 1. Counting instance of length 3.

we have $R(a_i, a_{i-1})$, $R(a_i, a_{i+1}) \in \mathfrak{D}$. We additionally require that, for $0 \leq i \leq 2k$ with $i$ even, $\mathfrak{D}$ contains the fact $Y_j(a_i)$, where $j = i/2 \bmod 3$. An example can be found in Figure 1. In the electronic appendix we show the following, which clearly establishes Theorem 3.7: there is a polynomial $p$ such that, for every $k > 0$:

(1) there is a query $(Q_k)_{k\geq 1}$ from $(\mathcal{ALCI}, \text{UCQ})$ such that $|Q_k| \leq p(k)$ and $Q_k(\mathfrak{C}_\ell) = 1$ for all $\ell \geq k$ and
(2) for every query $Q$ from $(\mathcal{ALCHU}, \text{UCQ})$ such that $Q(\mathfrak{C}_\ell) = 1$ iff $\ell \geq k$, we have $|Q| > 2^{k/3}$.

The queries $Q_k = (\mathbf{S}, \mathcal{O}_k, q_k)$ from Point 1 are constructed by realizing a counter with exponentially many bits using a family of ontologies and CQs that was introduced in Lutz [2007, 2008] to prove that query evaluation in $(\mathcal{ALCI}, \text{UCQ})$ is 2ExpTime-hard regarding combined complexity. Point 2 is established using a pumping argument. □

Also note that, when composing Points 1 and 2 of Theorem 3.6, we obtain a double-exponential blowup for the translation from $(\mathcal{ALCHIU}, \text{UCQ})$ to MDDlog. We believe this is unavoidable and that the culprit is inverse roles, that is, every translation from $(\mathcal{ALCI}, \text{UCQ})$ to MDDlog necessarily incurs a double-exponential blowup. The following observation provides some evidence, but leaves open whether the complexity lies in the size of the MDDlog program or in the difficulty of computing it.

Theorem 3.8. *There is no algorithm that translates a given* $(\mathcal{ALCI}, \text{UCQ})$ *query into an equivalent MDDlog program and runs in single-exponential time, unless* NExpTime = 2ExpTime.

Proof. A *singleton instance* is an instance that comprises exactly one element. The combined complexity of deciding whether a Boolean query evaluates to true in a singleton instance is:

—in coNP for MDDlog programs (guess an extension of the IDB relations so that goal is made false; due to the instance being singleton, it is then trivial to check in polytime whether all non-goal rules are satisfied);
—2ExpTime-complete for queries from $(\mathcal{ALCI}, \text{UCQ})$ [Lutz 2008].

If the translation from Theorem 3.8 existed, then we could decide the query evaluation problem in $(\mathcal{ALCI}, \text{UCQ})$ by first translating to MDDlog and then using the coNP procedure. □

To prove that certain extensions of $(\mathcal{ALC}, \text{UCQ})$ cannot be expressed in MDDlog, we show the following sufficient condition for nonexpressibility. The statement refers to the notion of a *k-coloring*, simply defined as a $\mathcal{C}_k$-coloring, with $\mathcal{C}_k$ a fixed set of $k$ colors disjoint from the considered schema.

Lemma 3.9. *A Boolean query* $Q$ *over schema* $\mathbf{S}$ *does not belong to MDDlog if, for all* $k, n > 0$*, there exist* $\mathbf{S}$*-instances* $\mathfrak{D}_0$ *and* $\mathfrak{D}_1$ *with* $Q(\mathfrak{D}_0) = 0$ *and* $Q(\mathfrak{D}_1) = 1$ *such that, for every k-coloring* $\mathfrak{B}_0$ *of* $\mathfrak{D}_0$*, there exists a k-coloring* $\mathfrak{B}_1$ *of* $\mathfrak{D}_1$ *such that, from every subinstance of* $\mathfrak{B}_1$ *with at most n elements, there is a homomorphism to* $\mathfrak{B}_0$*.*

Proof. Assume for a contradiction that the $\mathbf{S}$-instances $\mathfrak{D}_0$ and $\mathfrak{D}_1$ described in the lemma exist for all $k, n > 0$, but that $Q$ is equivalent to some query in MDDlog.

Then, by Proposition 3.2 there is a set $\mathcal{F}$ of $\mathcal{C}$-colored $\mathbf{S}$-structures such that, for all $\mathbf{S}$-instances $\mathfrak{D}$, we have $Q(\mathfrak{D}) = 1$ if and only if $\mathfrak{D} \notin \mathsf{Forb}(\mathcal{F})$. Let $k_0 = |\mathcal{C}|$, and let $n_0$ be the maximal number of elements in the domain of any $\mathfrak{F} \in \mathcal{F}$. We can assume without loss of generality that $\mathcal{C} = \mathcal{C}_{k_0}$.

Take $\mathbf{S}$-instances $\mathfrak{D}_0$ and $\mathfrak{D}_1$ satisfying the conditions of the lemma for $k_0, n_0$. Since $Q(\mathfrak{D}_0) = 0$, there exists a $\mathcal{C}$-coloring $\mathfrak{B}_0$ of $\mathfrak{D}_0$ such that $\mathfrak{F} \not\to \mathfrak{B}_0$ for every $\mathfrak{F} \in \mathcal{F}$. There is thus a $\mathcal{C}$-coloring $\mathfrak{B}_1$ of $\mathfrak{D}_1$ such that, from every subinstance of $\mathfrak{B}_1$with at most $n_0$ elements, there is a homomorphism to $\mathfrak{B}_0$. Since $Q(\mathfrak{D}_1) = 1$, we know there must exist some $\mathfrak{F} \in \mathcal{F}$ such that $\mathfrak{F} \to \mathfrak{B}_1$. As $\mathfrak{F}$ contains at most $n_0$ elements, we can compose this homomorphism with the previous one to obtain a homomorphism of $\mathfrak{F}$ into $\mathfrak{B}_0$, contradicting the fact that $\mathfrak{D}_0 \in \mathsf{Forb}(\mathcal{F})$. □

We apply Lemma 3.9 to show that two standard extensions of $(\mathcal{ALC}, \text{UCQ})$ have expressive power beyond MDDlog.

THEOREM 3.10. *$(\mathcal{S}, \text{UCQ})$ and $(\mathcal{ALCF}, \text{UCQ})$ are strictly more expressive than MDDlog.*

PROOF. To separate $(\mathcal{S}, \text{UCQ})$ from $(\mathcal{ALC}, \text{UCQ})$, we show that the following ontology-mediated query $Q = (\mathbf{S}, \mathcal{O}, q)$ cannot be expressed in $(\mathcal{ALC}, \text{UCQ})$: $\mathbf{S}$ consists of two role names $R$ and $S$, $\mathcal{O} = \{\mathsf{trans}(R), \mathsf{trans}(S)\}$, and $q = \exists xy(R(x, y) \wedge S(x, y))$. Thus, $Q$ expresses that there are two elements $a$ and $b$ such that $b$ is reachable from $a$ both via an $R$-path and via an $S$-path.

We apply Lemma 3.9 to show that $Q$ cannot be expressed in MDDlog. Assume that $k, n > 0$ are given. Let $m = n - 1$ and $m' = k^{m+2} + 1$. Define $\mathfrak{D}_1$ and $\mathfrak{D}_0$ as follows.

—$\mathfrak{D}_1$ has elements $e$, $f$ and $a_1, \dots, a_m$ and $b_1, \dots, b_m$ and the atoms $R(e, a_1)$, $R(a_m, f)$ and $R(a_i, a_{i+1})$ for $1 \le i < m$, and $S(e, b_1)$, $S(b_m, f)$ and $S(b_i, b_{i+1})$ for $1 \le i < m$.
—$\mathfrak{D}_0$ has elements $e^1, \dots, e^{m'}$ and $f^1, \dots, f^{m'}$ as well as $a_1^j, \dots, a_m^j$ for $1 \le j \le m'$ and $b_1^{i,j}, \dots, b_m^{i,j}$ for $1 \le j < i \le m'$. The atoms of $\mathfrak{D}_0$ consist of:
  —$R(e^i, a_1^i)$, $R(a_m^i, f^i)$, and $R(a_j^i, a_{j+1}^i)$ for $1 \le i \le m'$ and $1 \le j < m$;
  —$S(e_i, b_1^{i,j})$ and $S(b_m^{i,j}, f_j)$ for $1 \le j < i \le m'$, and
  $S(b_\ell^{i,j}, b_{\ell+1}^{i,j})$ for $1 \le \ell < m$ and $1 \le j < i \le m'$.

It can be checked that $Q(\mathfrak{D}_0) = 0$ and $Q(\mathfrak{D}_1) = 1$, as required. Let $\mathfrak{B}_0$ be a $k$-coloring of $\mathfrak{D}_0$. Since $m' = k^{m+2} + 1$, we can find $i, i'$ with $i > i'$ such that the colorings of $e^i, a_1^i, \dots, a_m^i, f^i$ and $e^{i'}, a_1^{i'}, \dots, a_m^{i'}, f^{i'}$ coincide. Define a $k$-coloring $\mathfrak{B}_1$ of $\mathfrak{D}_1$ by taking the coloring of $e^i, a_1^i, \dots, a_m^i, f^i$ for $e, a_1, \dots, a_m, f$ and the coloring of $b_1^{i,i'}, \dots, b_m^{i,i'}$ for $b_1, \dots, b_m$.

Let $\mathfrak{B}_1'$ be a subset of $\mathfrak{B}_1$ containing at most $n$ elements. We define a function $h$ from $\mathsf{adom}(\mathfrak{B}_1')$ to $\mathsf{adom}(\mathfrak{B}_0)$ as follows.

—If $e \notin \mathsf{adom}(\mathfrak{B}_1')$, then let $h$ be the restriction of the following mapping to $\mathsf{adom}(\mathfrak{B}_1')$. $h(a_\ell) = a_\ell^{i'}$, $h(b_\ell) = b_\ell^{i,i'}$ and $h(f) = f^{i'}$.
—If $f \notin \mathsf{adom}(\mathfrak{B}_1')$, then let $h$ be the restriction of the following mapping to $\mathsf{adom}(\mathfrak{B}_1')$. $h(a_\ell) = a_\ell^i$, $h(b_\ell) = b_\ell^{i,i'}$ and $h(e) = e^i$.
—Otherwise there exists $a_{i_0} \notin \mathsf{adom}(\mathfrak{B}_1')$. Then let $h$ be the restriction of the following mapping to $\mathsf{adom}(\mathfrak{B}_1')$: $h(e) = e^i$, $h(a_\ell) = a_\ell^i$ for all $\ell < i_0$, $h(a_\ell) = a_\ell^{i'}$ for all $\ell > i_0$, $h(b_\ell) = b_\ell^{i,i'}$ for all $1 \le \ell \le k$, and $h(f) = f^{i'}$.

It can be verified that $h$ is a homomorphism from $\mathfrak{B}_1'$ to $\mathfrak{B}_0$.

To separate $(\mathcal{ALCF}, \text{UCQ})$ from MDDlog, we first observe that every MDDlog program $\Pi$ is preserved under homomorphisms, that is, when $\mathbf{a} \in q_\Pi(\mathfrak{D})$ and $h$ is a homomorphism from $\mathfrak{D}$ into the instance $\mathfrak{D}'$, then $h(\mathbf{a}) \in q_\Pi(\mathfrak{D}')$. However, this is not the case for the query $Q = (\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALCF}, \text{UCQ})$, where $\mathbf{S}$ consists of a single binary relation symbol $R$, $\mathcal{O} = \{\mathsf{func}(R)\}$, and $q = A(x)$. In fact, there trivially is a homomorphism $h$ from $\mathfrak{D} = \{R(a, b_1), R(a, b_2)\}$ to $\mathfrak{D}' = \{R(a, b)\}$ with $h(a) = a$, but we have: (i) $a \in Q(\mathfrak{D})$ since $\mathfrak{D}$ is not consistent with $\mathcal{O}$ (that is, due to the standard names assumption, there is no instance in $\mathsf{Mod}(\mathcal{O})$ that extends $\mathfrak{D}$) and (ii) $\mathfrak{D}'$ is consistent with $\mathcal{O}$, thus obviously $a \notin Q(\mathfrak{D}')$. □

We now reconsider the OBDA languages studied earlier, but replace UCQs as the query language with AQs. The next result, in turn, is interesting when contrasted with Theorem 3.10: when $(\mathcal{ALC}, \text{UCQ})$ is replaced with $(\mathcal{ALC}, \text{AQ})$, then the addition of transitive roles no longer increases the expressive power (but the addition of functional roles still does, since the relevant part of the proof of Theorem 3.10 uses only AQs). Moreover, inverse roles do not lead to a double-exponential blowup.

THEOREM 3.11. *$(\mathcal{SHI}, \text{AQ})$ has the same expressive power as unary connected simple MDDlog. In fact, there is a polynomial $p$ such that, for every query $Q = (\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{SHI}, \text{AQ})$, there is an equivalent unary connected simple MDDlog program $\Pi$ such that $|\Pi| \leq 2^{p(|\mathcal{O}|)}$.*

PROOF (SKETCH). It is sufficient to give an equivalence-preserving translation from $(\mathcal{SHI}, \text{AQ})$ into $(\mathcal{ALC}, \text{AQ})$ that runs in polynomial time. This amounts to recalling the following folklore results.

—Every query $Q = (\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{SHI}, \text{AQ})$ can be converted into a query $Q' = (\mathbf{S}, \mathcal{O}', q)$ from $(\mathcal{ALCHI}, \text{AQ})$ such that $\mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D}) = \mathsf{cert}_{q,\mathcal{O}'}(\mathfrak{D})$ for all $\mathbf{S}$-instances $\mathfrak{D}$ and $|Q'| \leq \mathsf{poly}(|Q|)$; in fact, $\mathcal{O}'$ is obtained from $\mathcal{O}$ by replacing each transitivity statement $\mathsf{trans}(R)$ with the concept inclusions $\forall R.C \sqsubseteq \forall R.\forall R.C$, for each $C \in \mathsf{sub}(\mathcal{O})$ [Horrocks and Sattler 1999].

—Every query $Q = (\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALCHI}, \text{AQ})$ can be converted into a query $Q' = (\mathbf{S}, \mathcal{O}', q)$ from $(\mathcal{ALC}, \text{AQ})$ such that $\mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D}) = \mathsf{cert}_{q,\mathcal{O}'}(\mathfrak{D})$ for all $\mathbf{S}$-instances $\mathfrak{D}$ and $|Q'| \leq \mathsf{poly}(|Q|)$; for the elimination of inverse roles, see proof of Theorem 3.6. We can then replace each role hierarchy statement $R \sqsubseteq S$ with the concept inclusions $\forall S.C \sqsubseteq \forall R.C$, for each $C \in \mathsf{sub}(\mathcal{O})$ [Horrocks and Sattler 1999]. □

Using the techniques in Simancik [2012] one can show that, in Theorem 3.11, $\mathcal{SHI}$ and $\mathcal{SHIU}$ can be extended with all complex role inclusions that are admitted in the description logic $\mathcal{SROIQ}$ underlying OWL2 DL.

Note that, by Theorem 3.6, adding the universal role to $(\mathcal{ALC}, \text{UCQ})$ does not increase the expressive power beyond MDDlog. The situation is different when we consider AQs. Specifically, while $(\mathcal{ALC}, \text{AQ})$ has the same expressive power as unary simple connected MDDlog, adding the universal role corresponds, on the MDDlog side, to dropping the requirement that rule bodies must be connected. For example, the MDDlog query $\mathsf{goal}(x) \leftarrow \mathsf{adom}(x) \wedge A(y)$ is not connected and can be expressed in $(\mathcal{ALCU}, \text{AQ})$ using the ontology $\mathcal{O} = \{\exists U.A \sqsubseteq \mathsf{goal}\}$ and the AQ $\mathsf{goal}(x)$.

THEOREM 3.12. *$(\mathcal{ALCU}, \text{AQ})$ and $(\mathcal{SHIU}, \text{AQ})$ both have the same expressive power as unary simple MDDlog. In fact, there is a polynomial $p$ such that:*

*(1) for every query $Q = (\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{SHIU}, \text{AQ})$, there is an equivalent unary simple MDDlog program $\Pi$ such that $|\Pi| \leq 2^{p(|Q|)}$;*

*(2) for every unary simple MDDlog program $\Pi$, there is a query $Q$ from $(\mathcal{ALCU}, \text{AQ})$ such that $|Q| \leq p(|\Pi|)$.*

PROOF (SKETCH). For Point 1, we first note that the translations described in the proof of Theorem 3.11 also work in the presence of the universal role (without any modifications) and incur only a polynomial blowup. It thus suffices to establish Point 1 for queries from $(\mathcal{ALCU}, \text{AQ})$. Assume that a query $Q = (\mathbf{S}, \mathcal{O}, q)$ from this language is given. We can translate $Q$ into an MDDlog program as in the proof of Theorem 3.4 with the only difference that diagrams can now also be of the (disconnected) form $t_1(x) \wedge t_2(y)$.

For Point 2, let $\Pi$ be a unary simple MDDlog program. As shown in the proof of Theorem 3.4, every connected rule in $\Pi$ can be translated into an equivalent $\mathcal{ALC}$ concept inclusion. Rules with nonconnected bodies can be translated using the universal role. For example,

$$P_1(x) \vee P_2(y) \leftarrow A(x) \wedge B(y)$$

is rewritten into $A \sqcap \exists U.(B \sqcap \neg P_2) \sqcap \neg P_1 \sqsubseteq \bot$. □

We briefly discuss *Boolean atomic queries* (BAQs), that is, queries of the form $\exists x.A(x)$, where $A$ is a unary relation symbol. Such queries are relevant in the context of constraint satisfaction problems and will pop up naturally in Section 4.2. The following result shows that BAQs behave very similarly to AQs.

THEOREM 3.13. *Theorems 3.4 to Theorem 3.12 hold if AQs are replaced by BAQs and unary goal relations by* 0*-ary goal relation, respectively.*

PROOF. We show the required modifications to the proof of Theorem 3.4. The remaining results are proved by similarly minor modifications and left to the reader. For the translation from $(\mathcal{ALC}, \text{BAQ})$ to Boolean connected simple MDDlog, the only difference to the program constructed in the proof of Theorem 3.4 is that rules of the form $\mathsf{goal}(x) \leftarrow P_{A_0}(x)$ are replaced by rules of the form $\mathsf{goal} \leftarrow P_{A_0}(x)$. Conversely, for the translation from Boolean connected simple MDDlog to $(\mathcal{ALC}, \text{BAQ})$, we use goal as a concept name, translating, for example, the rule $\mathsf{goal}() \leftarrow R(x, y) \wedge P(y)$ to the concept inclusion $\exists R.P \sqsubseteq \mathsf{goal}$. As the BAQ, we then use $\exists x.\mathsf{goal}(x)$. □

### 3.2. Ontologies Specified in First-Order Logic

Ontologies formulated in description logic are not able to speak about relation symbols of arity greater than two.[5] To address this issue, we consider fragments of first-order logic as an ontology language that does not restrict the arity of relation symbols, namely the unary negation fragment (UNFO), the guarded fragment (GFO), and the guarded negation fragment (GNFO) [Bárány et al. 2010, 2012; ten Cate and Segoufin 2011]. Note that UNFO and GFO generalize the description logic $\mathcal{ALC}$ and several of its extensions in different ways, and that GNFO is a common generalization of UNFO and GFO. It turns out that (UNFO, UCQ) corresponds to MDDlog and in this sense constitutes a natural counterpart of $(\mathcal{ALC}, \text{UCQ})$ on schemas of unrestricted arity. In contrast, both GFO and GNFO turn out to be equivalent in expressive power to a version of *guarded datalog*. We start by considering the unary negation fragment.

*The Unary Negation Fragment.* The *unary negation fragment of first-order logic* (UNFO) [ten Cate and Segoufin 2011] is the fragment of first-order logic that consists of those formulas generated from atomic formulas, including equality, using conjunction, disjunction, existential quantification, and *unary negation*, that is, negation applied to a formula with at most one free variable. Thus, for example, $\neg\exists xyR(x, y)$ belongs to

[5]There are actually a few DLs that can handle relations of unrestricted arity. An example is the language $\mathcal{DLR}_{\text{reg}}$ presented in Calvanese et al. [1998]. $\mathcal{DLR}_{\text{reg}}$ has constructors that cannot be captured by any of the fragments of first-order logic we consider in this article (such as number restrictions and regular expressions), and it would be of great interest to investigate this language within in our framework. But this is beyond the scope of the present article.

UNFO, whereas $\exists xy \neg R(x, y)$ does not. Note that the translation of $\mathcal{ALC}$-ontologies into FO formulas given in Table II actually produces UNFO formulas.

THEOREM 3.14. *(UNFO, UCQ) has the same expressive power as MDDlog. In fact:*

(1) *there is a polynomial $p$ such that, for every query $(\mathbf{S}, \mathcal{O}, q)$ from (UNFO, UCQ), there is an equivalent MDDlog program $\Pi$ such that $|\Pi| \leq 2^{2^{p(|\mathcal{O}|+|q|)}}$;*
(2) *for every MDDlog program $\Pi$, there is an equivalent query $(\mathbf{S}, \mathcal{O}, q)$ from (UNFO, UCQ) such that $|q| \in O(|\Pi|)$ and $|\mathcal{O}| \in O(|\Pi|)$.*

PROOF (SKETCH). Point (2) is a consequence of Theorem 3.3 and the fact that $(\mathcal{ALC}, \text{UCQ})$ is a fragment of (UNFO, UCQ). Here, we provide the translation from (UNFO, UCQ) to MDDlog. Let $Q = (\mathbf{S}, \mathcal{O}, q) \in (\text{UNFO}, \text{UCQ})$ be given. Every UNFO formula with at most one free variable is equivalent to a disjunction of formulas generated by the grammar

$$\varphi(x) ::= \top \mid \neg\varphi(x) \mid \exists \mathbf{y}(\psi_1(x, \mathbf{y}) \wedge \cdots \wedge \psi_n(x, \mathbf{y})),$$

where each $\psi_i(x, \mathbf{y})$ is either a relational atom or a formula with at most one free variable generated by the same grammar. Note that all generated formulas have at most one free variable and that no equality is used, although we allow it in the original definition of UNFO. For the ontology $\mathcal{O}$, we assume without loss of generality that it is a *single* sentence generated by the previous grammar, rather than a disjunction of such sentences, because $\mathsf{cert}_{q,\mathcal{O}_1 \vee \mathcal{O}_2}(\mathfrak{D})$ is the intersection of $\mathsf{cert}_{q,\mathcal{O}_1}(\mathfrak{D})$ and $\mathsf{cert}_{q,\mathcal{O}_2}(\mathfrak{D})$, and MDDlog is closed under taking intersections of queries. Let $k$ be the maximum of the number of variables in $\mathcal{O}$ and the number of variables in $q$. We denote by $\mathsf{cl}_k(\mathcal{O}, q)$ the set of all formulas $\varphi(x)$ of the form

$$\exists \mathbf{y}(\psi_1(x, \mathbf{y}) \wedge \cdots \wedge \psi_n(x, \mathbf{y}))$$

with $\mathbf{y} = (y_1, \ldots, y_m)$, $m \leq k$, $n \leq |\mathcal{O}| + |q|$, where each $\psi_i$ is either a relational atom that uses a symbol from $q$ or is of the form $\chi(x)$ or $\chi(y_i)$, for $\chi(z) \in \mathsf{sub}(\mathcal{O})$. Note that $\mathsf{cl}_k(\mathcal{O}, q)$ contains all CQs that use only symbols from $q$ and whose size is bounded by the number of atoms in $q$. Also the length of each formula in $\mathsf{cl}_k(\mathcal{O}, q)$ is polynomial in $|\mathcal{O}| + |q|$, and consequently the cardinality of $\mathsf{cl}_k(\mathcal{O}, q)$ is single exponential in $|\mathcal{O}| + |q|$. A *type* $\tau$ is a subset of $\mathsf{cl}_k(\mathcal{O}, q)$. We introduce a fresh unary relation symbol $P_\tau$ for each type $\tau$, and denote by $\mathbf{S}'$ the schema that extends $\mathbf{S}$ with these additional symbols. As in the proof of Theorem 3.3, we call a structure $\mathfrak{B}$ over $\mathbf{S}' \cup \mathsf{sig}(\mathcal{O})$ *type coherent* if, for all types $\tau$ and elements $d$ in the domain of $\mathfrak{B}$, we have $P_\tau(d) \in \mathfrak{B}$ just in the case that $\tau$ is the (unique) type realized at $d$ in $\mathfrak{B}$. Diagrams, realizability, and "implying $q$" are defined as in the proof of Theorem 3.3. It follows from ten Cate and Segoufin [2011] that it is decidable whether a diagram implies a query and whether a diagram is realizable. The MDDlog program $\Pi$ is defined as in the proof of Theorem 3.3, repeated here for sake of completeness.

$$\begin{aligned}
\bigvee_{\tau \subseteq \mathsf{cl}_k(\mathcal{O},q)} P_\tau(x) &\leftarrow \mathsf{adom}(x) && \\
\bot &\leftarrow \delta(\mathbf{x}) && \text{for all nonrealizable diagrams } \delta(\mathbf{x}) \\
\mathsf{goal}(\mathbf{x}') &\leftarrow \delta(\mathbf{x}) && \text{for all diagrams } \delta(\mathbf{x}) \text{ that imply } q(\mathbf{x}')
\end{aligned}$$

In the electronic appendix, we prove that the resulting MDDlog query $q_\Pi$ is equivalent to $Q$. □

By a straightforward extension of the translation in Table II, one can show that every $\mathcal{ALCI}$-ontology translates into an UNFO sentence with only a linear blowup. As discussed after Theorem 3.7, there are reasons to believe that $(\mathcal{ALCI}, \text{UCQ})$ is double

exponentially more succinct than MDDlog. Thus, the same holds for (UNFO, UCQ) and MDDlog.

*The Guarded Fragment and the Guarded Negation Fragment.* The *Guarded Fragment of first-order logic* (GFO) comprises all formulas built up from atomic formulas using the Boolean connectives and guarded quantification of the form $\exists \mathbf{x}(\alpha \wedge \varphi)$ and $\forall \mathbf{x}(\alpha \rightarrow \varphi)$, where, in both cases, $\alpha$ is an atomic formula (a "guard") that contains all free variables of $\varphi$. To simplify the presentation of the results, we consider here the equality-free version of the guarded fragment. We do allow one special case of equality, namely the use of trivial equalities of the form $x = x$ as guards, which is equivalent to allowing unguarded quantifiers applied to formulas with at most one free variable. This restricted form of equality is sufficient to translate every $\mathcal{ALC}$-ontology into an equivalent sentence of GFO.

We next use our forbidden patterns characterization of MDDlog to show that some (GFO, UCQ) queries cannot be expressed in MDDlog.

PROPOSITION 3.15. *The Boolean query*

*(†) there are $a_1, \ldots, a_n, b$, for some $n \geq 2$, such that $A(a_1)$, $B(a_n)$, and $P(a_i, b, a_{i+1})$ for all $1 \leq i < n$*

*is definable in (GFO, UCQ) and not in MDDlog.*

PROOF. Let $\mathbf{S}$ consist of unary relation symbols $A, B$ and a ternary relation symbol $P$, and let $Q$ be the $\mathbf{S}$-query defined by (†). It is easy to check that $Q$ can be expressed by the (GFO, UCQ) query $q_{\mathbf{S},\mathcal{O},\exists x U(x)}$, where

$$\begin{aligned}\mathcal{O} \;=\; & \forall xyz\,(P(x,z,y) \rightarrow (A(x) \rightarrow R(z,x))) \\ & \wedge\, \forall xyz\,(P(x,z,y) \rightarrow (R(z,x) \rightarrow R(z,y))) \\ & \wedge\, \forall xy\,(R(x,y) \rightarrow (B(y) \rightarrow U(y))).\end{aligned}$$

It thus remains to show that $Q$ cannot be expressed in MDDlog. We make use of Lemma 3.9. Assume $k, n$ are given and let $m = k^{2n} + 2n$. Define $\mathbf{S}$-instances $\mathfrak{D}_1$ and $\mathfrak{D}_0$ as follows.

—$\mathfrak{D}_1$ has elements $d_1, \ldots, d_m, e$ and the atoms $A(d_1)$, $B(d_m)$, and $P(d_i, e, d_{i+1})$ for $1 \leq i < m$.
—$\mathfrak{D}_0$ has elements $d_1, \ldots, d_m$, and $e_1, \ldots, e_{m-1}$ and the following atoms: $A(d_1)$, $B(d_m)$, and $P(d_i, e_j, d_{i+1})$ whenever $1 \leq i < m$, $1 \leq j < m$, and $j \neq i$.

It can be checked that $Q(\mathfrak{D}_1) = 1$ and $Q(\mathfrak{D}_0) = 0$, as required. Let $\mathfrak{B}_0$ be a $k$-coloring of $\mathfrak{D}_0$. Define a $k$-coloring $\mathfrak{B}_1$ of $\mathfrak{D}_1$ by giving all elements of $\{d_1, \ldots, d_m\}$ the same color as in $\mathfrak{B}_0$. Choose $i$ with $n < i < m - n$ in such a way that, for every sequence $d_{\ell+1}, \ldots, d_{\ell+n}$ with $\ell \geq 0$ and $\ell + n < m$, there exists a sequence $d_{\ell'+1}, \ldots, d_{\ell'+n}$ with $\ell' \geq 0$ and $\ell' + n < m$ such that the coloring of $d_{\ell+1}, \ldots, d_{\ell+n}$ coincides with that of $d_{\ell'+1}, \ldots, d_{\ell'+n}$ and $i \notin \{\ell' + 1, \ell' + 2, \ldots, \ell' + n - 1\}$. Such an $i$ exists since $m \geq k^{2n} + 2n$. Finally, give $e$ the color of $e_i$.

To show that $\mathfrak{B}_1$ has the required properties, let $\mathfrak{B}_1'$ be any subset of $\mathfrak{B}_1$ having at most $n$ elements. We define a function $h$ from $\mathsf{adom}(\mathfrak{B}_1')$ to $\mathsf{adom}(\mathfrak{B}_0)$ as follows.

—If $e \in \mathsf{adom}(\mathfrak{B}_1')$, then $h(e) = e_i$.
—If $d_1 \in \mathsf{adom}(\mathfrak{B}_1')$, then $h(d_1) = d_1$.
—If $d_m \in \mathsf{adom}(\mathfrak{B}_1')$, then $h(d_m) = d_m$.
—If $d_{\ell+1}, \ldots, d_{\ell+p}$ is a maximal sequence of elements from $\mathsf{adom}(\mathfrak{B}_1')$ with $\ell + p < m$, then let $\ell'$ be such that the coloring of $d_{\ell+1}, \ldots, d_{\ell+p}$ coincides with the coloring of $d_{\ell'+1}, \ldots, d_{\ell'+p}$ and $i \notin \{\ell' + 1, \ell' + 2, \ldots, \ell' + n - 1\}$. Set $h(d_{\ell+j}) = d_{\ell'+j}$ for every $1 \leq j \leq p$.

It is easily verified that $h$ defines a homomorphism from $\mathsf{adom}(\mathfrak{B}'_1)$ to $\mathsf{adom}(\mathfrak{B}_0)$. Applying Lemma 3.9, we can conclude that $Q$ is not equivalent to any MDDlog query. □

COROLLARY 3.16. *(GFO, UCQ) is strictly more expressive than MDDlog.*

As fragments of first-order logic, the unary negation fragment and the guarded fragment are incomparable in expressive power. They have a common generalization, known as the guarded negation fragment (GNFO) [Bárány et al. 2011]. This fragment is defined in the same way as UNFO except that, besides unary negation, we allow *guarded negation* of the form $\alpha \wedge \neg\varphi$, where the guard $\alpha$ is an atomic formula that contains all the variables of $\varphi$. Again, for simplicity, we consider here the equality-free version of the language, except that we allow the use of trivial equalities of the form $x = x$ as guards. As we will see, for the purpose of OBDA, GNFO is no more powerful than GFO. Specifically, (GFO, UCQ) and (GNFO, UCQ) are expressively equivalent to a natural generalization of MDDlog, namely *frontier-guarded DDlog*. Recall that a datalog rule is *guarded* if its body includes an atom that contains all variables which occur in the rule [Gottlob et al. 2002]. A weaker notion of guardedness, that we call here *frontier guardedness*, inspired by Baget et al. [2011] and Bárány et al. [2012], requires that, for each atom $\alpha$ in the head of the rule, there is an atom $\beta$ in the rule body such that all variables that occur in $\alpha$ occur also in $\beta$. We define a frontier-guarded DDlog query to be a query defined by a DDlog program in which every rule is frontier guarded. Observe that frontier-guarded DDlog subsumes MDDlog because the head of a rule in MDDlog contains at most one variable that has to occur in the body of the rule. We now show that both (GFO, UCQ) and (GNFO, UCQ) have the same expressive power as frontier-guarded DDlog. For understandng the following theorem, it is useful to recall that every sentence of GFO can be translated into an equivalent sentence of GNFO with only a polynomial blowup [Bárány et al. 2011].

THEOREM 3.17. *(GFO, UCQ) and (GNFO, UCQ) have the same expressive power as frontier-guarded DDlog. In fact, there is a polynomial $p$ such that:*

1. *for every query $(\mathbf{S}, \mathcal{O}, q)$ from (GNFO, UCQ), there is an equivalent frontier-guarded DDlog program $\Pi$ such that $|\Pi| \leq 2^{2^{p(|\mathcal{O}|+|q|)}}$;*
2. *for every frontier-guarded DDlog program $\Pi$, there is an equivalent query $(\mathbf{S}, \mathcal{O}, q)$ from (GNFO, UCQ) such that $|q| \in O(|\Pi|)$ and $|\mathcal{O}| \in O(|\Pi|)$;*
3. *for every query $(\mathbf{S}, \mathcal{O}, q)$ from (GNFO, UCQ), there is an equivalent query $(\mathbf{S}, \mathcal{O}', q')$ from (GFO, UCQ) such that $|q'| \leq |q| + 2^{p(|\mathcal{O}|)}$ and $|\mathcal{O}'| \leq p(|\mathcal{O}|)$.*

PROOF. We start with item (2) by describing the translation from frontier-guarded DDlog to (GNFO, UCQ). Let $\Pi$ be a frontier-guarded DDlog query. It is easily verified that, if we write out the implication symbol in a frontier-guarded DDlog rule using conjunction and negation, the resulting formula belongs to GNFO. Thus, we can take $\mathcal{O}$ to be the set of all non-goal rules of $\Pi$, viewed as a GNFO sentence, and let $q$ be the UCQ that consists of all bodies of rules whose conclusion contains the IDB relation goal. It is easy to check that the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S}$ is the schema consisting of all EDB relations, is equivalent to the frontier-guarded DDlog query $q_\Pi$.

For item (3), we make use of a result from Bárány et al. [2011] which states that, for every GNFO sentence $\phi$ over a schema $\mathbf{S}' \supseteq \mathbf{S}$, we can construct in polynomial time a GFO sentence $\psi$ and a positive-existential first-order sentence $\chi$, both over a possibly larger schema $\mathbf{T} = \mathbf{S}' \cup \{T_1, \ldots, T_n\}$, such that $\phi$ is logically equivalent to the existential second-order sentence $\exists T_1 \ldots T_n(\psi \wedge \neg\chi)$. By applying this translation to $\mathcal{O}$ and using the fact that a positive-existential FO sentence can be translated to a UCQ in exponential time, we obtain a GFO sentence $\mathcal{O}'$ and a Boolean UCQ $q''$, both over schema $\mathbf{T} = \mathbf{S}' \cup \{T_1, \ldots, T_n\}$, such that $\mathcal{O}$ is logically equivalent to $\exists T_1 \ldots T_n(\mathcal{O}' \wedge \neg q'')$.

Note that, even though the size of $q''$ may be exponential in the size of $\mathcal{O}$, this is only because $q''$ may consist of exponentially many CQs, and each CQ is itself of polynomial size. If $q$ is Boolean, it now follows that the (GNFO, UCQ) query $(\mathbf{S}, \mathcal{O}, q)$ is equivalent to the (GFO, UCQ) query $(\mathbf{S}, \mathcal{O}', q')$ where $q' = q \vee q''$. If, on the other hand, $q$ is an $n$-ary UCQ with $n > 0$, then one final step is needed: we turn the Boolean UCQ $q''$ into an $n$-ary UCQ by adding, for each free variable of $q$, an atom to the body of each CQ in $q''$ in all possible ways, thereby expressing that the variable takes a value from the active domain. Note that modification of $q''$ may involve an exponential blowup, but the resulting UCQ is still only single exponential in the size of $\mathcal{O}$ since each of its CQs is only of polynomial size.

Finally, for item (1), we show in the electronic appendix how to translate (GNFO, UCQ) to frontier-guarded DDlog. The translation is very much along the same lines as the translation from (UNFO, UCQ) to MDDlog, but with a more sophisticated notion of types. Note that, since every sentence of GFO is equivalent to a sentence of GNFO [Bárány et al. 2011] (which can be constructed in polynomial time), the translations from (GNFO, UCQ) to frontier-guarded DDlog apply to (GFO, UCQ) queries as well. □

Note that the translation from frontier-guarded DDlog to (GFO, UCQ) in Theorem 3.17 involves an exponential blowup, unlike all the translations of MDDlog versions to OBDA languages that we have seen before. We leave it open whether this blowup can be avoided.

## 4. CORRESPONDENCES TO MMSNP AND CSP

We first show that MDDlog captures coMMSNP and thus, by the results obtained in the previous section, the same is true for many OBDA languages based on UCQs. We also propose GMSNP, an extension of MMSNP inspired by frontier-guarded DDlog, and show that (GFO, UCQ) and (GNFO, UCQ) capture coGMSNP and that GMSNP has the same expressive power as a previously proposed extension of MMSNP called $\text{MMSNP}_2$. Then we turn to fragments of MDDlog that correspond to OBDA languages based on AQs and show that they capture CSPs (and generalizations thereof).

### 4.1. Correspondences to MMSNP

An *MMSNP formula* over schema $\mathbf{S}$ has the form $\exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ with $X_1, \ldots, X_n$ monadic second-order (SO) variables, $x_1, \ldots, x_m$ FO variables, and $\varphi$ a conjunction of quantifier-free formulas of the form

$$\psi = \alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta_1 \vee \cdots \vee \beta_m \text{ with } n, m \geq 0,$$

where each $\alpha_i$ is of the form $X_i(x_j)$, $R(\mathbf{x})$ (with $R \in \mathbf{S}$), or $x_j = x_k$, and each $\beta_i$ is of the form $X_i(x_j)$. Note that this presentation is syntactically different from, but semantically equivalent to, the original definition from Feder and Vardi [1998] that does not use the implication symbol and instead restricts the allowed polarities of atoms.

In order to use MMSNP as a query language, and in contrast to the standard definition, we admit free FO variables and speak of *sentences* to refer to MMSNP formulas without free variables. To connect with the query languages studied thus far, we are interested in queries obtained by the complements of MMSNP formulas: each MMSNP formula $\Phi$ over schema $\mathbf{S}$ with $n$ free variables gives rise to a query

$$q_\Phi(\mathfrak{D}) = \{\mathbf{a} \in \mathsf{adom}(\mathfrak{D})^n \mid (\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \not\models \Phi[\mathbf{a}]\},$$

where we set $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi$ to true when $\mathfrak{D}$ is the empty instance (that is, $\mathsf{adom}(\mathfrak{D}) = \emptyset$) and $\Phi$ is a sentence. We call the resulting query language *coMMSNP*. Note that the equality atoms in MMSNP allow us to express queries that require some

of the answer variables to be bound to the same domain element. This is needed for the following observation that coMMSNP has the same expressive power as MDDlog. We remark that equality atoms are not present in the original definition of MMSNP in Feder and Vardi [1998], but can easily be eliminated in MMSNP formulas without free variables by identifying variables that co-occur in an equality atom.

PROPOSITION 4.1. *coMMSNP and MDDlog have the same expressive power.*

PROOF. We start with the translation from coMMSNP to MDDlog. Let $\Phi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ be an MMSNP formula over schema $\mathbf{S}$ with free variables $y_1, \ldots, y_k$, and let $q_\Phi \in$ coMMSNP be the corresponding query. We can assume without loss of generality that all implications $\psi = \alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta_1 \vee \cdots \vee \beta_m$ in $\Phi$ satisfy the following properties: (i) every free variable $y_j$ appears in some atom $\alpha_i$ or $\beta_i$; and (ii) if $\alpha_i$ is an equality atom, then it takes the form $y_j = y_\ell$. In fact, we can achieve (i) by replacing violating implications $\psi$ with the set of implications $\psi'$ that can be obtained from $\psi$ by adding, for each variable $y_j$ that is not present in $\psi$, a body atom $S(\mathbf{x})$, where $S$ is a relation symbol that occurs in $\Phi$ and $\mathbf{x}$ is a tuple of variables that contains $y_j$ once and otherwise only fresh variables that do not occur in $\Phi$. To enforce condition (ii), for every equality atom $z_1 = z_2$ in which $z_2$ is not a free variable, we replace all occurrences of $z_2$ by $z_1$ and delete the atom.

We construct an MDDlog program $\Pi_\Phi$, in which the $X_i$ are treated as IDB relations, and additional IDB relations $\overline{X}_i$ are used to simulate the complements of the $X_i$. We include in $\Pi_\Phi$ the following rules that ensure each domain element is assigned to either $X_i$ or its complement, but not both.

$$X_i(z) \vee \overline{X}_i(z) \leftarrow \mathsf{adom}(z) \qquad (1 \leq i \leq n)$$
$$\bot \leftarrow X_i(z) \wedge \overline{X}_i(z) \qquad (1 \leq i \leq n)$$

To translate the implications in $\Phi$ into datalog rules, we first rewrite each implication $\psi$ in $\Phi$ with a nonempty head by adding $\overline{X}_i(z)$ to the body of $\psi$ for every head atom $X_i(z)$, and then replacing the head by $\bot$. We thus have a set of implications of the form $\vartheta \rightarrow \bot$. For each such implication $\psi$, we let $\sim_\psi$ be the smallest equivalence relation on $\{y_1, \ldots, y_k\}$ such that $y_j \sim_\psi y_\ell$ whenever $\psi$ contains an equality atom $y_j = y_\ell$, and we denote by $[y_j]_\psi$ the equivalence class under $\sim_\psi$ containing $y_j$. Then, for every implication $\vartheta \rightarrow \bot$, we include in $\Pi_\Phi$ the rule

$$\mathsf{goal}([y_1]_\psi, \ldots, [y_k]_\psi) \leftarrow \vartheta',$$

where $\vartheta'$ is obtained from $\vartheta$ by replacing each $y_j$ by $[y_j]_\psi$ and deleting all equality atoms. Note that, because of assumption (i) given before, every head variable $[y_j]_\psi$ occurs in some body atom, and by (ii), $\vartheta'$ contains no equality atoms. It is straightforward to show that $q_\Phi \equiv q_{\Pi_\Phi}$.

Conversely, let $\Pi$ be a $k$-ary MDDlog program whose set of EDB relations is $\mathbf{S}$. Reserve fresh variables $y_1, \ldots, y_k$ as free variables for the desired MMSNP formula, and let $X_1, \ldots, X_n$ be the IDB relations in $\Pi$ and $x_1, \ldots, x_m$ the FO variables in $\Pi$ that do not occur in the goal relation. Set $\Phi_\Pi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$, where $\varphi$ is the conjunction of all non-goal rules in $\Pi$ plus the implication $\vartheta' \rightarrow \bot$ for each rule $\mathsf{goal}(\mathbf{x}) \leftarrow \vartheta$ in $\Pi$. Here, $\vartheta'$ is obtained from $\vartheta$ by replacing each variable $x \in \mathbf{x}$ whose leftmost occurrence in the rule head is in the $i$th position with $y_i$, and then conjunctively adding $y_i = y_j$ whenever the $i$th and $j$th positions in the rule head have the same variable. It can be verified that, for all $\mathbf{S}$-instances $\mathfrak{D}$, we have $q_\Pi(\mathfrak{D}) = q_{\Phi_\Pi}(\mathfrak{D})$. □

Thus, the characterizations of OBDA languages in terms of MDDlog provided in Section 3 also establish the descriptive complexity of these languages by identifying them with (the complement of) MMSNP.

We now consider OBDA languages based on the guarded fragment and GNFO. By Proposition 3.15, (GFO, UCQ) and (GNFO, UCQ) are strictly more expressive than MDDlog and we cannot use Proposition 4.1 to relate these query languages to the Feder-Vardi conjecture. Theorem 3.17 suggests it would be useful to have a generalization of MMSNP that is equivalent to frontier-guarded DDlog. Such a generalization is introduced next.

A formula of *guarded monotone strict NP* (GMSNP) has the form $\exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ with $X_1, \ldots, X_n$ SO variables of any arity, $x_1, \ldots, x_n$ FO variables, and $\varphi$ a conjunction of formulas

$$\psi = \alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta_1 \vee \cdots \vee \beta_m \text{ with } n, m \geq 0,$$

where each $\alpha_i$ is of the form $X_i(\mathbf{x})$, $R(\mathbf{x})$ (with $R \in \mathbf{S}$), or $x = y$, and each $\beta_i$ is of the form $X_i(\mathbf{x})$. Additionally, we require that, for every head atom $\beta_i$, there is a body atom $\alpha_j$ such that $\alpha_j$ contains all variables from $\beta_i$. GMSNP gives rise to a query language coGMSNP in analogy with the definition of coMMSNP. It can be shown by a straightforward syntactic transformation that every MMSNP formula is equivalent to some GMSNP formula. Together with Proposition 3.15 and Theorem 3.17, this yields the second statement of the following lemma; the first statement can be proved similarly to Proposition 4.1.

THEOREM 4.2. *coGMSNP has the same expressive power as frontier-guarded DDlog and is strictly more expressive than coMMSNP.*

PROOF. The proof of the first part follows the lines of the proof of Proposition 4.1. The only notable difference is that, in place of the rules $X_i(z) \vee \overline{X}_i(z) \leftarrow \mathsf{adom}(z)$, we have rules of the form

$$X_i(\mathbf{z}) \vee \overline{X}_i(\mathbf{z}) \leftarrow R(\mathbf{u}),$$

where $R \in \mathbf{S}$ and all variables in $\mathbf{z}$ appear also in $\mathbf{u}$.

It thus remains to show that coGMSNP is strictly more expressive than coMMSNP. Note first that it is at least as expressive: we can convert any MMSNP formula into an equivalent one satisfying conditions (i) and (ii) from the proof of Proposition 4.1, and clearly every such MMSNP formula is also a GMSNP formula. To see that coGMSNP is indeed strictly more expressive than coMMSNP, note that, by Proposition 3.15, there is a (GFO, UCQ) query $q$ that is not expressible in MDDlog. By Proposition 4.1, $q$ is not expressible in coMMSNP; by Theorem 3.17 and the first part of Theorem 4.2, $q$ is expressible in coGMSNP. □

Although defined in a different way, GMSNP is essentially the same logic as $\text{MMSNP}_2$, which is studied in Madelaine [2009]. Specifically, $\text{MMSNP}_2$ is the extension of MMSNP in which monadic SO variables range over sets of domain elements and facts, and where atoms of the form $X(R(\mathbf{x}))$ are allowed in place of atoms $X(x)$ with $X$ an SO variable and $R$ from the data schema $\mathbf{S}$. Additionally, a guardedness condition is imposed requiring that, whenever an atom $X(R(\mathbf{x}))$ occurs in a rule head, then the atom $R(\mathbf{x})$ must occur in the rule body. Formally, the SO variables $X_i$ are interpreted in an instance $\mathfrak{D}$ as sets $\pi(X_i) \subseteq \mathsf{adom}(\mathfrak{D}) \cup \mathfrak{D}$ and $\mathfrak{D} \models_\pi X(R(x_1, \ldots, x_n))$ if $R(\pi(x_1), \ldots, \pi(x_n)) \in \pi(X)$. We observe the following.

THEOREM 4.3. *GMSNP and $\text{MMSNP}_2$ have the same expressive power.*

Details for the proof of Theorem 4.3 can be found in the electronic appendix. In Madelaine [2009], it was left as an open question whether $\text{MMSNP}_2$ is more expressive than MMSNP, which is resolved by the preceding results.

COROLLARY 4.4. *$\text{MMSNP}_2$ is strictly more expressive than MMSNP.*

### 4.2. Correspondences to CSPs

We show that OBDA languages based on atomic queries capture CSPs (and generalizations thereof). The proofs employ the equivalences between OBDA languages and fragments of MDDlog that have already been established in Section 3. Recall that each instance $\mathfrak{B}$ over a schema $\mathbf{S}$ gives rise to a *constraint satisfaction problem*, that is, to decide, given an instance $\mathfrak{D}$ over $\mathbf{S}$, whether there is a homomorphism from $\mathfrak{D}$ to $\mathfrak{B}$ (written $\mathfrak{D} \to \mathfrak{B}$). In this context, the instance $\mathfrak{B}$ is also called the *template* of the CSP.

CSPs give rise to a query language coCSP in the spirit of the query language coMMSNP introduced in the previous section. In its basic version, this language is Boolean and turns out to have exactly the same expressive power as $(\mathcal{ALC}, \text{BAQ})$, where BAQ is the class of Boolean atomic queries of the form $\exists x\, A(x)$. To also cover non-Boolean AQs (that take the form $A(x)$), we consider two natural generalizations of CSPs. First, a *generalized CSP* is defined by a finite set $\mathcal{F}$ of templates, rather than a single template [Foniok et al. 2008]. The problem then consists in deciding, given an instance $\mathfrak{D}$, whether there is a template $\mathfrak{B} \in \mathcal{F}$ such that $\mathfrak{D} \to \mathfrak{B}$. Second, in a *(generalized) CSP with marked elements*, both the template(s) and the input instance are endowed with a tuple of distinguished domain elements [Feder et al. 2004; Alexe et al. 2011]. More precisely, we define an *n-ary marked* $\mathbf{S}$*-instance* as a tuple $(\mathfrak{D}, d_1, \ldots, d_n)$, where $\mathfrak{D}$ is an $\mathbf{S}$-instance and each $d_i$ belongs to $\mathsf{adom}(\mathfrak{D})$. Let $(\mathfrak{D}, \mathbf{d})$ and $(\mathfrak{B}, \mathbf{b})$ be $n$-ary marked $\mathbf{S}$-instances. A mapping $h$ is a homomorphism from $(\mathfrak{D}, \mathbf{d})$ to $(\mathfrak{B}, \mathbf{b})$, written $(\mathfrak{D}, \mathbf{d}) \to (\mathfrak{B}, \mathbf{b})$, if it is a homomorphism from $\mathfrak{D}$ to $\mathfrak{B}$ and $h(d_i) = b_i$ for $1 \leq i \leq n$. A (generalized) CSP with marked elements is then defined like a (generalized) CSP based on this extended notion of homomorphism.

We now introduce the query languages obtained from the different versions of CSPs, where generalized CSPs with marked elements constitute the most general case. Specifically, each finite set $\mathcal{F}$ of $n$-ary marked $\mathbf{S}$-instances gives rise to an $n$-ary query $\text{coCSP}(\mathcal{F})$ that maps every $\mathbf{S}$-instance $\mathfrak{D}$ to

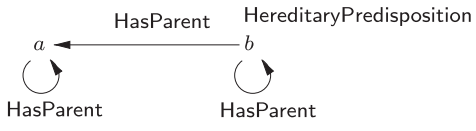
$$\big\{\mathbf{d} \in \mathsf{adom}(\mathfrak{D})^n \mid \forall (\mathfrak{B}, \mathbf{b}) \in \mathcal{F} : (\mathfrak{D}, \mathbf{d}) \not\to (\mathfrak{B}, \mathbf{b})\big\}.$$

The query language that consists of all such queries is called *generalized coCSP with marked elements*. The fragment of this query language that is obtained by admitting only sets of templates $\mathcal{F}$ without marked elements is called *generalized coCSP*, and the fragment induced by singleton sets $\mathcal{F}$ without marked elements is called *coCSP*.

*Example* 4.5. Selecting an illustrative fragment of Examples 2.1 and 2.2, let

$$\mathcal{O} = \{\exists \mathsf{HasParent}.\mathsf{HereditaryPredisposition} \sqsubseteq \mathsf{HereditaryPredisposition}\}, \text{ and}$$
$$\mathbf{S} = \{\mathsf{HereditaryPredisposition}, \mathsf{HasParent}\}.$$

Moreover, let $q_2(x) = \mathsf{HereditaryPredisposition}(x)$ be the query from Example 2.2. To identify a query in coCSP with marked elements that is equivalent to the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q_2)$, let $\mathfrak{B}$ be the following template.



We claim that, for all instances $\mathfrak{D}$ over $\mathbf{S}$ and for all $d \in \mathsf{adom}(\mathfrak{D})$, we have $d \in \mathsf{cert}_{q_2,\mathcal{O}}(\mathfrak{D})$ iff $(\mathfrak{D}, d) \not\to (\mathfrak{B}, a)$ and thus the query coCSP$(\mathfrak{B}, a)$ is as required. To see why, first observe that, if $(\mathfrak{D}, d) \to (\mathfrak{B}, a)$, then $\mathfrak{D}$ cannot contain $\mathsf{HereditaryPredisposition}(d)$ nor contain a chain of $\mathsf{HasParent}$ that starts with $d$ and ends at some $\mathsf{HereditaryPredisposition}$ fact. It follows that we can construct a model $\mathfrak{D}' \supseteq \mathfrak{D}$ of $\mathcal{O}$ in which $\mathsf{HereditaryPredisposition}(d) \notin \mathfrak{D}'$, and so $d \notin \mathsf{cert}_{q_2,\mathcal{O}}(\mathfrak{D})$. Conversely, if $d \notin \mathsf{cert}_{q_2,\mathcal{O}}(\mathfrak{D})$, then there exists a model $\mathfrak{D}' \supseteq \mathfrak{D}$ of $\mathcal{O}$ in which $\mathsf{HereditaryPredisposition}(d) \notin \mathfrak{D}'$. We can use this model to construct the desired homomorphism from $(\mathfrak{D}, d)$ to $(\mathfrak{B}, a)$.

The following theorem summarizes the connections between OBDA languages with (Boolean) atomic queries, MDDlog, and CSPs. Note that we consider binary schemas only.

THEOREM 4.6. *The following are lists of query languages that have the same expressive power:*

(1) $(\mathcal{ALCU}, \mathbf{AQ})$, $(\mathcal{SHIU}, \mathbf{AQ})$*, unary simple MDDlog, and generalized coCSP with one marked element;*
(2) $(\mathcal{ALC}, \mathbf{AQ})$, $(\mathcal{SHI}, \mathbf{AQ})$*, unary connected simple MDDlog, and generalized coCSPs with one marked element such that all templates have the same instance;*
(3) $(\mathcal{ALCU}, \mathbf{BAQ})$, $(\mathcal{SHIU}, \mathbf{BAQ})$*, Boolean simple MDDlog, and generalized coCSP;*
(4) $(\mathcal{ALC}, \mathbf{BAQ})$, $(\mathcal{SHI}, \mathbf{BAQ})$*, Boolean connected simple MDDlog, and coCSP.*

*Moreover, given the ontology-mediated query or monadic datalog program, the corresponding CSP template(s) can be constructed in exponential time.*

PROOF. The equivalences between OBDA languages and fragments of MDDlog have already been proved in Section 3. We establish the remaining equivalences.

We treat Points 1–4 in reverse order starting with Point 4 and proving that Boolean connected simple MDDlog and coCSP are equally expressive. Let $\Pi$ be a Boolean connected simple MDDlog program. A *type for* $\Pi$ is a set $\tau$ of IDBs and unary EDBs from $\Pi$. By $\mathsf{tp}(\Pi)$ we denote the set of all types for $\Pi$. We say $\tau \in \mathsf{tp}(\Pi)$ is *realizable* if there is a model of $\Pi$ in which some element $d$ satisfies exactly the unary relation symbols from $\tau$. Note this is equivalent to the singleton instance $\{A(d) \mid A \in \tau\}$ being a model of $\Pi$. For binary $R \in \mathbf{S}$, we call a pair $(\tau_1, \tau_2)$ of types *R-coherent* if there is a model $\mathfrak{D}$ of $\Pi$ and two elements $d_1, d_2$ from $\mathfrak{D}$ such that $R(d_1, d_2) \in \mathfrak{D}$ and $d_i$ satisfies exactly the unary relation symbols from $\tau_i$, $i \in \{1, 2\}$. Note this is equivalent to the two-element instance $\{R(d_1, d_2)\} \cup \bigcup_{i\in\{1,2\}}\{A(d_i) \mid A \in \tau_i\}$ being a model of $\Pi$. Given a set $T$ of realizable types, we define the *canonical model* $\mathfrak{B}_T$ for $T$ and $\Pi$ by setting

$$\mathfrak{B}_T = \{P(\tau) \mid \tau \in T, P \in \mathbf{S} \cap \tau\} \cup \{R(\tau_1, \tau_2) \mid \ \tau_1, \tau_2 \in T, (\tau_1, \tau_2) \text{ is } R\text{-coherent}, R \in \mathbf{S}\}.$$

Let $T$ be the set consisting of all realizable $\tau \in \mathsf{tp}(\Pi)$ such that $\mathsf{goal} \notin \tau$. One can show that, for every $\mathbf{S}$-instance $\mathfrak{D}$, we have $\mathfrak{D} \to \mathfrak{B}_T$ iff $q_\Pi(\mathfrak{D}) = 0$. Thus, the query defined by $\Pi$ is equivalent to the query coCSP$(\mathfrak{B}_T)$.[6]

Conversely, we associate with every $\mathbf{S}$-instance $\mathfrak{B}$ the simple connected MDDlog program

$$\begin{aligned}\Pi_{\mathfrak{B}} \ = \ & \{\bot \leftarrow P_d(x) \wedge P_{d'}(x) \mid d \neq d'\} \\ & \cup \{\bot \leftarrow P_d(x) \wedge P_{d'}(y) \wedge R(x, y) \mid R(d, d') \notin \mathfrak{B}, R \in \mathbf{S}\} \\ & \cup \{\bot \leftarrow P_d(x) \wedge B(x) \mid B(d) \notin \mathfrak{B}, B \in \mathbf{S}\},\end{aligned}$$

[6]In the limit case where $\Pi$ is such that $q_\Pi(\mathfrak{D}) = 1$ for all instances $\mathfrak{D}$, then the statement holds for all nonempty $\mathbf{S}$-instances.

where $P_d$, $d \in \mathsf{adom}(\mathfrak{B})$ are IDBs. For a CSP template $\mathfrak{B}$ over schema $\mathbf{S}$, the program

$$\Pi = \Pi_{\mathfrak{B}} \cup \Big\{ \bigvee_{d \in \mathsf{dom}(\mathfrak{B})} P_d(x) \leftarrow \mathsf{adom}(x) \Big\}$$

then defines a query equivalent to the query coCSP($\mathfrak{B}$).

For Point 3, we must show that Boolean simple MMDlog and generalized coCSP are equally expressive. The construction is similar to that of Point 4, except that we must deal with disconnected MDDlog programs and admit more than one CSP template. For the first direction, let $\Pi$ be a Boolean simple MMDlog program. By introducing IDB relations that represent maximal connected components of rule bodies, we can rewrite $\Pi$ into an equivalent program in which the only nonconnected rules are of the form $P(y) \leftarrow P(x) \wedge \mathsf{adom}(y)$ with $P$ an IDB relation. We assume $\Pi$ has this property. Let $\mathcal{C}$ be the set of IDB relations that occur in a rule of this form in $\Pi$. For any subset $D$ of $\mathcal{C}$, which intuitively represents a choice of disconnected rule bodies that homomorphically embed into a given instance, let $T(D)$ be the set of all realizable $\tau \in \mathsf{tp}(\Pi)$ such that $\mathsf{goal} \notin \tau$ and $\tau \cap \mathcal{C} = D$. We define $\mathcal{F}$ as the set of templates $\mathfrak{B}_{T(D)}$ with $D \subseteq \mathcal{C}$. It can be shown that, for every $\mathbf{S}$-instance $\mathfrak{D}$, we have $\mathfrak{D} \to \mathfrak{B}$ for some $\mathfrak{B} \in \mathcal{F}$ iff $q_\Pi(\mathfrak{D}) = 0$. Consequently, $\Pi$ is equivalent to coCSP($\mathcal{F}$).

For the other direction, let $\mathcal{F}$ be a set of $\mathbf{S}$-instances. Consider the programs $\Pi_{\mathfrak{B}}$ introduced earlier, and let $\Pi$ be the union of $\Pi_{\mathfrak{B}}$, for all $\mathfrak{B} \in \mathcal{F}$, and the following additional rules:

$$\Big\{ \bigvee_{\mathfrak{B} \in \mathcal{F}} P_{\mathfrak{B}}(x) \leftarrow \mathsf{adom}(x) \Big\}$$
$$\cup \Big\{ \bigvee_{d \in \mathsf{adom}(\mathfrak{B})} P_d(x) \leftarrow P_{\mathfrak{B}}(x), P_{\mathfrak{B}}(y) \leftarrow P_{\mathfrak{B}}(x) \wedge \mathsf{adom}(y) \mid \mathfrak{B} \in \mathcal{F} \Big\}.$$

Again, it can be shown that $\Pi$ is equivalent to the query coCSP($\mathcal{F}$).

For Point 2 we must show that unary connected simple MDDlog has the same expressive power as generalized coCSPs with one marked element such that all templates have the same instance. The construction is again similar to that of Point 4, except that we now have unary instead of Boolean MDDlog programs, templates that are marked instances instead of unmarked ones, and coCSP queries defined by a set of templates (based on the same instance) instead of a single one. For the first direction, assume $\Pi$ is a unary connected simple MDDlog program. Let $T$ be the set of all realizable types for $\Pi$ and define

$$\mathcal{F} = \{(\mathfrak{B}_T, \tau) \mid \tau \in T, \mathsf{goal} \notin \tau\}.$$

One can show that, for every $\mathbf{S}$-instance $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$, we have $(\mathfrak{D}, d) \to (\mathfrak{B}_T, \tau)$ for some $(\mathfrak{B}_T, \tau) \in \mathcal{F}$ iff $d \notin q_\Pi(\mathfrak{D})$. Thus, the query defined by $\Pi$ is equivalent to that defined by $\mathcal{F}$.

Conversely, assume $\mathcal{F}$ is a finite set of unary marked $\mathbf{S}$-instances based upon the $\mathbf{S}$-instance $\mathfrak{B}$. Define the program $\Pi$ by setting

$$\Pi = \Pi_{\mathfrak{B}} \cup \big\{ \mathsf{goal}(x) \leftarrow P_d(x) \mid d \neq b \text{ for all } b \text{ with } (\mathfrak{B}, b) \in \mathcal{F} \big\}$$
$$\cup \Big\{ \bigvee_{d \in \mathsf{dom}(\mathfrak{B})} P_d(x) \leftarrow \mathsf{adom}(x) \Big\}.$$

One can show that, for every $\mathbf{S}$-instance $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$, we have $(\mathfrak{D}, d) \to (\mathfrak{B}, b)$ for some $(\mathfrak{B}, b) \in \mathcal{F}$ iff $d \notin q_\Pi(\mathfrak{D})$. Thus $\Pi$ expresses the same query as $\mathcal{F}$.

For Point 1, we must show that unary simple MDDlog and generalized coCSP with one marked element are equally expressive. We start with the direction from MDDlog to CSP. The construction combines features of the constructions for Point 2 and Point 3. Assume that a unary simple MDDlog program $\Pi$ is given. We may again assume that the only nonconnected rules in $\Pi$ are of the form $P(y) \leftarrow P(x) \wedge \mathsf{adom}(y)$, where $P$ is an IDB relation. Let $\mathcal{C}$ be the set of IDB relations that occur in a rule of this form in $\Pi$. For any subset $D$ of $\mathcal{C}$, let $T'(D)$ be the set of all realizable $\tau \in \mathsf{tp}(\Pi)$ such that $\tau \cap \mathcal{C} = D$. Define the set $\mathcal{F}$ of templates as follows.

$$\mathcal{F} = \{(\mathfrak{B}_{T'(D)}, \tau) \mid D \subseteq \mathcal{C} \text{ and } \tau \in T'(D) \text{ and } \mathsf{goal} \notin \tau\}$$

One can show that, for every $\mathbf{S}$-instance $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$, there exists $(\mathfrak{B}_{T'(D)}, \tau) \in \mathcal{F}$ with $(\mathfrak{D}, d) \to (\mathfrak{B}_{T'(D)}, \tau)$ iff $d \notin q_\Pi(\mathfrak{D})$. Thus, the program $\Pi$ is equivalent to the query defined by $\mathcal{F}$.

Conversely, assume $\mathcal{F}$ is a finite set of unary marked $\mathbf{S}$-instances. Define for every $(\mathfrak{B}, b) \in \mathcal{F}$ a program $\Pi_{\mathfrak{B},b}$ by adding $\{\mathsf{goal}(x) \leftarrow P_d(x) \mid d \neq b\}$ to $\Pi_{\mathfrak{B}}$. Finally introduce fresh IDBs $P_{(\mathfrak{B},b)}$, $(\mathfrak{B}, b) \in \mathcal{F}$, and let $\Pi$ be the union of all $\Pi_{\mathfrak{B},b}$ and

$$\Big\{ \bigvee_{(\mathfrak{B},b)\in\mathcal{F}} P_{\mathfrak{B},b}(x) \leftarrow \mathsf{adom}(x) \Big\}$$

$$\cup \Big\{ \bigvee_{d\in\mathsf{adom}(\mathfrak{B})} P_d(x) \leftarrow P_{\mathfrak{B},b}(x), P_{\mathfrak{B},b}(y) \leftarrow P_{\mathfrak{B},b}(x) \wedge \mathsf{adom}(y) \mid (\mathfrak{B}, b) \in \mathcal{F} \Big\}.$$

One can show that, for every $\mathbf{S}$-instance $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$, we have $(\mathfrak{D}, d) \to (\mathfrak{B}, b)$ for some $(\mathfrak{B}, b) \in \mathcal{F}$ iff $d \notin q_\Pi(\mathfrak{D})$. Thus $\Pi$ is equivalent to the query coCSP($\mathcal{F}$).

Finally, we show that, in all four cases given the ontology-mediated query or monadic datalog program, the corresponding CSP template(s) can be constructed in exponential time. This is clear from the previous construction if the monadic datalog program is given. To prove the exponential upper bound for ontology-mediated queries it is sufficient to observe the following two points: (i) in the construction of MDDlog programs from ontology-mediated queries in the proofs of Theorems 3.4, 3.11, and 3.12, the number of IDBs in the constructed program is polynomial in the size of the input ontology-mediated query; (ii) the construction of CSP template(s) from MDDlog programs as described before is exponential only in the size of the schema $\mathbf{S}$ and the number of IDBs (but not in the size or number of rules). □

## 5. APPLICATIONS

We apply the correspondence results of the previous section to obtain results about the complexity of query evaluation, query containment, and FO and datalog rewritability for OBDA languages with UCQs and atomic queries. Since more is known about CSPs than about MMSNP, we obtain a richer set of results for OBDA languages based on atomic queries than for OBDA languages based on UCQs.

### 5.1. Query Evaluation and Dichotomies

For an $n$-ary query $q$, the evaluation problem is to decide, given an instance $\mathfrak{D}$ and an $n$-tuple $\mathbf{a}$ of elements from $\mathfrak{D}$, whether $\mathbf{a} \in q(\mathfrak{D})$. Our first result is that the Feder-Vardi dichotomy conjecture for CSPs is true if and only if there is a dichotomy between PTIME and CONP for query evaluation in ($\mathcal{ALC}$, UCQ), and the same is true for several other

OBDA languages. For brevity, we say that a query language *has a dichotomy between* PTIME *and* coNP, referring only implicitly to the evaluation problem.

Theorem 4.6 allows us to transfer dichotomy results from CSP to query evaluation for OBDA languages with atomic queries.

THEOREM 5.1. *($\mathcal{ALC}$, BAQ) has a dichotomy between* PTIME *and* coNP *iff the Feder-Vardi conjecture holds. The same is true for ($\mathcal{SHIU}$, AQ), and ($\mathcal{SHIU}$, BAQ).*

PROOF. Since $\mathcal{SHIU}$ ontologies can be replaced by $\mathcal{ALCU}$ ontologies in ontology-mediated queries due to Theorem 3.12, the "if" direction of (all cases mentioned in) Theorem 5.1 actually follows from Theorem 5.3. The "only if" direction is a consequence of Theorem 4.6. □

To extend Theorem 5.1 to OBDA languages with UCQs we exploit the fact that the Feder-Vardi dichotomy conjecture can equivalently be stated for MMSNP sentences [Feder and Vardi 1998; Kun 2007]. We also require that every MMSNP formula is polynomially equivalent to an MMSNP sentence.

PROPOSITION 5.2. *Every MMSNP formula is polynomially equivalent to an MMSNP sentence.*

Proposition 5.2 is proved in the electronic appendix using an extension of forbidden pattern problems characterizing MMSNP formulas. Now the following result follows from Proposition 4.1 and Theorems 3.3, 3.6, and 3.14.

THEOREM 5.3. *($\mathcal{ALC}$, UCQ) has a dichotomy between* PTIME *and* coNP *iff the Feder-Vardi conjecture holds. The same is true for ($\mathcal{ALCHIU}$, UCQ) and (UNFO, UCQ).*

Recall that ($\mathcal{ALCF}$, UCQ) is an extension of ($\mathcal{ALC}$, UCQ) that was shown in Section 3 to be more expressive than ($\mathcal{ALC}$, UCQ) itself. It was already proved in Lutz and Wolter [2012, Theorem 27] that, compared to ontology-mediated queries based on $\mathcal{ALC}$, the functional roles of $\mathcal{ALCF}$ dramatically increase the computational power. This is true even for atomic queries.

THEOREM 5.4 ([LUTZ AND WOLTER 2012]). *For every* NP*-Turing machine M, there is a query Q from ($\mathcal{ALCF}$, AQ) such that the complement of the word problem of M has the same complexity as evaluating Q, up to polynomial-time reductions. Consequently, ($\mathcal{ALCF}$, AQ) does not have a dichotomy between* PTIME *and* coNP *(unless* PTIME = NP*).*

($\mathcal{S}$, UCQ) is another extension of ($\mathcal{ALC}$, UCQ) that was identified in Section 3 to be more expressive than ($\mathcal{ALC}$, UCQ) itself. We leave it as an interesting open question whether ($\mathcal{S}$, UCQ) has a dichotomy between PTIME and coNP if the Feder-Vardi conjecture holds. Another open question of interest is whether Theorem 5.3 can be extended to (GFO, UCQ) and (GNFO, UCQ), that is, whether GMSNP (equivalently, $\text{MMSNP}_2$) has a dichotomy between PTIME and NP if the Feder-Vardi conjecture holds. While this question is implicit already in Madelaine [2009], the results established in this article underline its significance from a different perspective.

### 5.2. Query Containment

We apply the correspondence results from earlier to obtain results about the query containment problem for OBDA languages. Specifically, the following general containment problem was proposed in Bienvenu et al. [2012] as a powerful tool for OBDA: given ontology-mediated queries $(\mathbf{S}, \mathcal{O}_i, q_i)$, $i \in \{1, 2\}$, decide whether, for all $\mathbf{S}$-instances $\mathfrak{D}$,

we have $\mathsf{cert}_{q_1,\mathcal{O}_1}(\mathfrak{D}) \subseteq \mathsf{cert}_{q_2,\mathcal{O}_2}(\mathfrak{D})$.[7] Applications include the optimization of ontology-mediated queries and managing the effects on query answering of replacing an ontology with a new, updated version. In terms of OBDA languages such as $(\mathcal{ALC}, \text{UCQ})$, the preceding problem corresponds to query containment in the standard sense: an $\mathbf{S}$-query $q_1$ is *contained in* an $\mathbf{S}$-query $q_2$, written $q_1 \subseteq q_2$, if, for every $\mathbf{S}$-instance $\mathfrak{D}$, we have $q_1(\mathfrak{D}) \subseteq q_2(\mathfrak{D})$. Note that there are also less general (and computationally simpler) notions of query containment in OBDA that do not fix the data schema [Calvanese et al. 1998].

It was proved in Feder and Vardi [1998] that containment of MMSNP sentences is decidable. In the electronic appendix, we prove the following extension of this result to MMSNP formulas.

PROPOSITION 5.5. *The containment problem for MMSNP formulas is polynomial-time reducible to the containment problem for MMSNP sentences.*

We thus obtain the following result for OBDA languages.

THEOREM 5.6. *Query containment is decidable for the OBDA languages* $(\mathcal{ALC}, \textit{UCQ})$, $\mathcal{ALCHIU}, \textit{UCQ})$, *and* $(\textit{UNFO}, \textit{UCQ})$.

Note this result is considerably stronger than those in Bienvenu et al. [2012] that considered only containment of ontology-mediated queries $(\mathbf{S}, \mathcal{O}, q)$ with $q$ an atomic query, since already this basic case turned out to be technically intricate. The treatment of CQs and UCQs was left open, including all cases stated in Theorem 5.6.

We established decidability results for query containment in OBDA languages based on UCQs. For OBDA languages based on AQs and BAQs, we even obtain a tight complexity bound. It is easy to see that query containment in coCSP is characterized by homomorphisms between templates, that is, the answers to $\text{coCSP}(\mathcal{F})$ are contained in those to $\text{coCSP}(\mathcal{F}')$ just in the case that, for every $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$, there is some $(\mathfrak{B}', \mathbf{b}') \in \mathcal{F}'$ such that $(\mathfrak{B}, \mathbf{b}) \to (\mathfrak{B}', \mathbf{b}')$. Consequently, it is straightforward to show that query containment for generalized coCSP with marked elements is NP-complete. Thus, Theorem 4.6 yields the following NEXPTIME upper bound for query containment in OBDA languages. We obtain a matching lower bound by a reduction from a NEXPTIME-complete tiling problem; details are given in the electronic appendix.

THEOREM 5.7. *Query containment in* $(\mathcal{SHIU}, \textit{AQ})$ *and* $(\mathcal{SHIU}, \textit{BAQ})$ *is in* NEXPTIME. *It is* NEXPTIME*-hard already for* $(\mathcal{ALC}, \textit{AQ})$ *and for* $(\mathcal{ALC}, \textit{BAQ})$.

Undecidability of query containment for $\mathcal{ALCF}$ is proved in Bienvenu et al. [2012] under the slightly different definition of query containment used there (see Footnote 7). The following undecidability result is proved in the electronic appendix, where we show how the gap between the definitions of query containment can be bridged.

THEOREM 5.8. *Query containment is undecidable for* $(\mathcal{ALCF}, \textit{BAQ})$ *and for* $(\mathcal{ALCF}, \textit{AQ})$.

### 5.3. FO and Datalog Rewritability

One prominent approach to answering ontology-mediated queries is to make use of existing relational database systems or datalog engines, eliminating the ontology by query rewriting [Calvanese et al. 2007; Eiter et al. 2012; Grau et al. 2013]. Formally, a query over a schema $\mathbf{S}$ is said to be *FO rewritable* if equivalent to some FO-query over

[7] In fact, this definition is slightly different from the one used in Bienvenu et al. [2012]. There, containment is defined only over instances $\mathfrak{D}$ that are consistent with respect to both $\mathcal{O}_1$ and $\mathcal{O}_2$.

$\mathbf{S}$, and is *datalog rewritable* if equivalent to some datalog query over $\mathbf{S}$.[8] We observe that, for ontology-mediated queries, FO rewritability implies datalog rewritability.

PROPOSITION 5.9. *If $Q = (\mathbf{S}, \mathcal{O}, q)$ is an ontology-mediated query with $\mathcal{O}$ formulated in equality-free FO and $q$ a UCQ, then $q_Q$ is preserved by homomorphisms. Consequently, it follows from Rossman [2008] that if $q_Q$ is FO rewritable, then $q_Q$ is rewritable into a UCQ (thus into datalog).*

PROOF. Let $\mathfrak{D}_1$ and $\mathfrak{D}_2$ be two instances over the same schema such that there exists a homomorphism $h : \mathfrak{D}_1 \to \mathfrak{D}_2$. Suppose for the sake of contradiction that $\mathbf{a} \in q_Q(\mathfrak{D}_1)$ but $h(\mathbf{a}) \notin q_Q(\mathfrak{D}_2)$. Then there is a finite relational structure $(\mathsf{dom}_2, \mathfrak{D}'_2) \models \mathcal{O}$ such that $\mathfrak{D}_2 \subseteq \mathfrak{D}'_2$ and $h(\mathbf{a}) \notin q(\mathfrak{D}'_2)$. Let $(\mathsf{dom}_1, \mathfrak{D}'_1)$ be the inverse image of $(\mathsf{dom}_2, \mathfrak{D}'_2)$ under $h$. More precisely, $\mathsf{dom}_1 = \mathsf{adom}(\mathfrak{D}_1) \cup (\mathsf{dom}_2 \setminus \mathsf{adom}(\mathfrak{D}_2))$, and $\mathfrak{D}'_1$ contains all facts whose $\widehat{h}$-image is a fact of $\mathfrak{D}'_2$, where $\widehat{h}$ is the map that extends $h$ by sending every element of $\mathsf{adom}(\mathfrak{D}'_2) \setminus \mathsf{adom}(\mathfrak{D}_2)$ to itself. Clearly $\mathfrak{D}_1 \subseteq \mathfrak{D}'_1$. Furthermore, $\mathbf{a} \notin q(\mathfrak{D}'_1)$ because $\widehat{h} : \mathfrak{D}'_1 \to \mathfrak{D}'_2$ is a homomorphism and $q$ is preserved by homomorphisms. To obtain a contradiction against $\mathbf{a} \in q_Q(\mathfrak{D}_1)$, it therefore only remains to show that $(\mathsf{dom}_1, \mathfrak{D}'_1) \models \mathcal{O}$. It is known that equality-free first-order sentences are preserved by passing from a structure to its quotient under an equivalence relation that is a congruence. By construction, the kernel of the map $\widehat{h}$ is a congruence relation on the structure $(\mathsf{dom}_1, \mathfrak{D}'_1)$ and its quotient is isomorphic to $(\mathsf{dom}_2, \mathfrak{D}'_2)$. □

Example 2.2 illustrates that ontology-mediated queries are not always rewritable into an FO-query, and the same holds for datalog rewritability. It is a central problem to decide, given an ontology-mediated query, whether it is FO rewritable and whether it is datalog rewritable (and to construct a rewriting when it exists). By leveraging the CSP connection, we show that both problems are decidable and pinpoint their complexities.

On the CSP side, FO rewritability corresponds to FO definability, and datalog rewritability to datalog definability. These notions have been extensively investigated, culminating in the following results (that are rephrased in terms of coCSP queries).

THEOREM 5.10. *Deciding, for a given instance $\mathfrak{B}$, whether coCSP($\mathfrak{B}$) is FO rewritable is* NP*-complete [Larose et al. 2007]. The same is true for datalog rewritability [Freese et al. 2009].*[9]

By combining the preceding theorem with Theorem 4.6, we obtain NEXPTIME upper bounds for deciding FO rewritability and datalog rewritability of queries from $(\mathcal{SHI}, \text{BAQ})$.

To capture the more important AQs rather than only BAQs, we show that Theorem 5.10 can be lifted, in a natural way, to queries based on generalized CSPs with marked elements. The general idea is to eliminate constants by replacing them with fresh unary relation symbols, as made precise in the following. For every $n > 0$ and schema $\mathbf{S}$, we fix a sequence $P_1, \ldots, P_n$ of unary relation symbols that do not appear in $\mathbf{S}$. We then associate to every $n$-ary marked $\mathbf{S}$-instance $(\mathfrak{B}, b_1, \ldots, b_n)$ the (unmarked) instance $(\mathfrak{B}, b_1, \ldots, b_n)^c = \mathfrak{B} \cup \{P_1(b_1), \ldots, P_n(b_n)\}$. Given a set $\mathcal{F} = \{(\mathfrak{B}_1, \mathbf{b}_1), \ldots, (\mathfrak{B}_m, \mathbf{b}_m)\}$ of $n$-ary marked instances, we use $\mathcal{F}^c$ to denote the set $\{(\mathfrak{B}_1, \mathbf{b}_1)^c, \ldots, (\mathfrak{B}_m, \mathbf{b}_m)^c\}$. The following central proposition relates each query coCSP($\mathcal{F}$) to the queries coCSP($(\mathfrak{B}, \mathbf{b})^c$) with $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$, thus transitioning from constants to unary relation symbols and from

[8] By datalog query, we mean a DDLog query defined by a disjunction-free DDLog program, that is, a DDLog program in which the head of each rule is a single atom.

[9] An NP algorithm for datalog definability is implicit in Freese et al. [2009], based on results from Barto and Kozik [2009]; see also Bulatov [2009]. We thank Benoit Larose and Libor Barto for pointing this out.

multiple templates to a single template. Note that the queries coCSP($(\mathfrak{B}, \mathbf{b})^c$) are over the schema $\mathbf{S} \cup \{P_1, \ldots, P_n\}$. We call $n$-ary marked $\mathbf{S}$-instances $(\mathfrak{B}_1, \mathbf{b}_1)$ and $(\mathfrak{B}_2, \mathbf{b}_2)$ homomorphically incomparable if there are no homomorphisms from $(\mathfrak{B}_1, \mathbf{b}_1)$ to $(\mathfrak{B}_2, \mathbf{b}_2)$ and from $(\mathfrak{B}_2, \mathbf{b}_2)$ to $(\mathfrak{B}_1, \mathbf{b}_1)$.

PROPOSITION 5.11. *For every finite set $\mathcal{F}$ of mutually homomorphically incomparable $n$-ary marked $\mathbf{S}$-instances:*

(1) *coCSP($\mathcal{F}$) is FO rewritable iff coCSP($(\mathfrak{B}, \mathbf{b})^c$) is FO rewritable for every $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$;*
(2) *coCSP($\mathcal{F}$) is datalog rewritable iff coCSP($(\mathfrak{B}, \mathbf{b})^c$) is datalog rewritable for every $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$.*

We split the proof of Point 1 of Proposition 5.11 into two parts, first showing how FO rewritability of generalized coCSP with marked elements can be reduced to FO rewritability of generalized coCSP without marked elements, and next giving the reduction from generalized to plain coCSP.

LEMMA 5.12. *Let $\mathcal{F}$ be a finite set of $n$-ary marked $\mathbf{S}$-instances. Then coCSP($\mathcal{F}$) is FO rewritable iff coCSP($\mathcal{F}^c$) is FO rewritable.*

PROOF. Let $\mathcal{F}$ be a finite set of $n$-ary marked $\mathbf{S}$-instances, and suppose that coCSP($\mathcal{F}^c$) is equivalent to the FO sentence $\varphi$. Let $x_1, \ldots, x_n$ be distinct variables not appearing in $\varphi$, and let $\varphi'$ be the formula obtained from $\varphi$ by replacing every subformula of the form $P_i(x)$ by $x = x_i$, and if no such subformula exists, conjoining the atom $x_i = x_i$ (such conjuncts merely ensure that $x_i$ appears in $\varphi'$). It can be checked that the FO-query $\varphi'$ is equivalent to coCSP($\mathcal{F}$).

For the converse, we make use of a characterization of FO rewritability of generalized coCSPs with marked elements using finite obstruction sets. Let $\mathcal{F}$ be a finite set of $n$-ary marked $\mathbf{S}$-instances. A set $\Omega$ of $n$-ary marked $\mathbf{S}$-instances is an *obstruction set for* $\mathcal{F}$ if, for all $n$-ary marked $\mathbf{S}$-instances $(\mathfrak{D}, \mathbf{d})$, the following conditions are equivalent:

—there exists $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$ such that $(\mathfrak{D}, \mathbf{d}) \to (\mathfrak{B}, \mathbf{b})$;
—there does not exist $(\mathfrak{G}, \mathbf{g}) \in \Omega$ such that $(\mathfrak{G}, \mathbf{g}) \to (\mathfrak{D}, \mathbf{d})$.

It is known that, for any finite set of instances $\mathcal{F}$, coCSP($\mathcal{F}$) is FO rewritable if and only if $\mathcal{F}$ has a finite obstruction set. This was shown in Atserias [2005] for (unmarked) instances, and follows easily from results in Rossman [2008] even for the case of marked instances. Finally, it was shown in Alexe et al. [2011, Proposition A.2 (1)] that if $\mathcal{F}$ has a finite obstruction set, then so does $\mathcal{F}^c$. □

LEMMA 5.13. *Let $\mathcal{F}$ be a finite set of $\mathbf{S}$-instances.*

—*If coCSP($\mathfrak{B}$) is FO rewritable for all $\mathfrak{B} \in \mathcal{F}$, then coCSP($\mathcal{F}$) is FO rewritable.*
—*Conversely, if all instances in $\mathcal{F}$ are mutually homomorphically incomparable and coCSP($\mathcal{F}$) is FO rewritable, then coCSP($\mathfrak{B}$) is FO rewritable for every $\mathfrak{B} \in \mathcal{F}$.*

PROOF. For the first statement, choose for every $\mathfrak{B} \in \mathcal{F}$ an FO-sentence $\varphi_{\mathfrak{B}}$ such that $\mathfrak{D} \models \varphi_{\mathfrak{B}}$ iff $\mathfrak{D} \not\to \mathfrak{B}$ for all $\mathbf{S}$-instances $\mathfrak{D}$. Let $\varphi$ be the conjunction over all $\varphi_{\mathfrak{B}}$ with $\mathfrak{B} \in \mathcal{F}$. Then, for all $\mathbf{S}$-instances $\mathfrak{D}$, we have $\mathfrak{D} \models \varphi$ iff $\mathfrak{D} \not\to \mathfrak{B}$ for every $\mathfrak{B} \in \mathcal{F}$, as required.

To prove the other direction, we require the notion of a critical obstruction: an $\mathbf{S}$-instance $\mathfrak{A}$ is called a *critical obstruction* for a finite set of $\mathbf{S}$-instances $\mathcal{G}$ iff $\mathfrak{A} \not\to \mathfrak{B}$ for every $\mathfrak{B} \in \mathcal{G}$ but for any proper subinstance $\mathfrak{A}' \subsetneq \mathfrak{A}$ there exists $\mathfrak{B} \in \mathcal{F}$ such that $\mathfrak{A}' \to \mathfrak{B}$. It can be verified that $\mathcal{G}$ has a finite obstruction set iff it has only finitely many critical obstructions (up to isomorphism).

Let $\mathcal{F}$ be a finite set of mutually homomorphically incomparable $\mathbf{S}$-instances such that coCSP($\mathcal{F}$) is FO rewritable. Assume for a contradiction that coCSP($\mathfrak{B}_0$) is not

FO rewritable for some $\mathfrak{B}_0 \in \mathcal{F}$. Then $\mathfrak{B}_0$ possesses an infinite set $\mathcal{C}$ of (pairwise nonisomorphic) critical obstructions. Let $\mathfrak{B}'_0 \subseteq \mathfrak{B}_0$ be such that $\mathfrak{B}'_0 \not\to \mathfrak{B}$ for every $\mathfrak{B} \in \mathcal{F} \setminus \{\mathfrak{B}_0\}$ but, for every proper subinstance $\mathfrak{B}''_0 \subsetneq \mathfrak{B}'_0$, there is some $\mathfrak{B} \in \mathcal{F} \setminus \{\mathfrak{B}_0\}$ with $\mathfrak{B}''_0 \to \mathfrak{B}$. Note that such a subinstance $\mathfrak{B}'_0$ must exist because of our assumption that the instances in $\mathcal{F}$ are mutually homomorphically incomparable. It is easily verified that the infinitely many instances obtained by taking the disjoint union of $\mathfrak{B}'_0$ and some $\mathfrak{A} \in \mathcal{C}$ are all critical obstructions for $\mathcal{F}$. Thus coCSP($\mathcal{F}$) is not FO rewritable, and we have derived a contradiction. □

Point 2 of Proposition 5.11 is a consequence of the following lemma.

LEMMA 5.14. *Let $\mathcal{F}$ be a finite set of $n$-ary marked* **S***-instances.*

—*If coCSP*$((\mathfrak{B}, \mathbf{b})^c)$ *is datalog rewritable for all* $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$*, then coCSP*($\mathcal{F}$) *is datalog rewritable.*

—*Conversely, if all* $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$ *are mutually homomorphically incomparable and coCSP*($\mathcal{F}$) *is datalog rewritable, then coCSP*$((\mathfrak{B}, \mathbf{b})^c)$ *is datalog rewritable for every* $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$.

PROOF. For the first statement, let $\mathcal{F}$ be a finite set of $n$-ary marked **S**-instances, and suppose that each coCSP$((\mathfrak{B}, \mathbf{b})^c)$ is datalog rewritable. Since datalog queries are known to be closed under conjunction, coCSP($\mathcal{F}^c$) must also be datalog rewritable. Let $\Pi$ be a datalog program whose corresponding query $q_\Pi$ is equivalent to coCSP($\mathcal{F}^c$). We construct a new datalog program $\Pi'$ that uses the same IDB relations from $\Pi$, except that the arity of each relation symbol (included the goal relation) is increased by $n$. The rules of $\Pi'$ are obtained by applying the following operations to each rule in $\Pi$.

—Fix a sequence of distinct fresh variables $\mathbf{y} = y_1, \ldots, y_n$ and replace each IDB atom $R(\mathbf{x})$ by $R(\mathbf{x}, \mathbf{y})$.

—Let $\sim$ be the least equivalence relation over the rule variables satisfying the following property: if $P_i(z)$ appears in the rule body, then $z \sim y_i$. Drop all $P_i$ atoms and merge all variables appearing in the same equivalence class under $\sim$.

—For each variable $y_i$ that appears only in the head, add $\mathsf{adom}(y_i)$ to the rule body.

It can be verified that, for every **S**-instance $\mathfrak{D}$ and tuple $\mathbf{d} \in \mathsf{adom}(\mathfrak{D})^n$: $\mathbf{d} \in q_{\Pi'}(\mathfrak{D})$ iff $q_\Pi((\mathfrak{D}, \mathbf{d})^c) = 1$. From this and the fact that $\Pi$ defines coCSP($\mathcal{F}^c$), we can show that $\mathbf{d} \in q_{\Pi'}(\mathfrak{D})$ iff $(\mathfrak{D}, \mathbf{d}) \not\to (\mathfrak{B}, \mathbf{b})$ for all $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$.

For the second statement, we make use of a known characterization of datalog rewritability in terms of *obstruction sets of bounded treewidth* [Feder and Vardi 1998]. Recall from the proof of Lemma 5.12 the notion of an obstruction set for a set of instances. We also recall (cf. Hell et al. [1996]) that an (unmarked) instance has treewidth (at most) $k$, if it admits a tree decomposition such that each bag of the tree decomposition has size at most $k + 1$. We extend this definition to marked instances by saying that a *marked* instance $(\mathfrak{D}, \mathbf{d})$ has treewidth (at most) $k$ if $\mathfrak{D}$ has a tree decomposition in which every bag contains the elements $\mathbf{d}$ and at most $k + 1$ other elements.

Let $\mathcal{F}$ be a finite set of $n$-ary marked **S**-instances, and suppose that coCSP($\mathcal{F}$) is equivalent to a datalog program whose rules contain at most $k$ variables. Then, by a standard argument (refer to Feder and Vardi [1998]), we have that $\mathcal{F}$ has an obstruction set of treewidth $k$. Indeed, consider any marked instance $(\mathfrak{D}, \mathbf{d}) \in$ coCSP($\mathcal{F}$). Then, by definition of the semantics of datalog, there exists some finite derivation of $\mathsf{goal}(\mathbf{d})$. From this derivation (viewed in a top-down fashion), we can read off a conjunctive query that is satisfied by $\mathbf{d}$ in $\mathfrak{D}$, and that implies the truth of the datalog query. The canonical instance of this conjunctive query is then an obstruction for $\mathcal{F}$ that can be shown to have treewidth at most $k$ (here, by the *canonical instance* of a conjunctive query we mean the instance whose elements are the variables in the CQ and whose

facts are the atoms of the CQ). Rather than spelling out the details, we illustrate the construction with an example. Let $\mathfrak{D} = \{R(a, b), R(b, a), P(a)\}$, and consider the datalog $\Pi$ program consisting of the following rules.

$$\begin{array}{lll} \text{EvenDist}(x) & \leftarrow & \text{P}(x) \\ \text{EvenDist}(x) & \leftarrow & \text{R}(x, y) \wedge \text{OddDist}(y) \\ \text{OddDist}(x) & \leftarrow & \text{R}(x, y) \wedge \text{EvenDist}(y) \\ \text{goal}(x) & \leftarrow & \text{EvenDist}(x) \end{array}$$

Clearly $a \in Q_\Pi(\mathfrak{D})$. There are many witnessing derivations, each giving rise to a corresponding conjunctive query. The conjunctive queries in question are of the form $q_i(x) = \exists y_1 \dots y_{2k+1}(R(x, y_1) \wedge R(y_1, y_2) \wedge \dots \wedge R(y_{2k-1}, y_{2k}), P(y_{2k}))$. Note that, in this example, the canonical instances of these conjunctive queries all have treewidth 1. In general, the treewidth is bounded by the maximum number of variables occurring in a rule of the datalog program. By constructing in this way an obstruction for each $(\mathfrak{D}, \mathbf{d}) \in \text{coCSP}(\mathcal{F})$, we obtain a (possibly infinite) obstruction set of bounded treewidth.

By Alexe et al. [2011, Proposition A.2 (1)], we have that $\mathcal{F}$ has an obstruction set of bounded treewidth if and only if $\mathcal{F}^c$ has an obstruction set of bounded treewidth (although it is not explicitly stated, it can easily be verified that the relevant construction used there preserves bounded treewidth). Thus, we have that $\mathcal{F}^c$ has an obstruction set of some bounded treewidth, say $k$.

From the fact that the marked instances in $\mathcal{F}$ are pairwise homomorphically incomparable, it follows that also the instances in $\mathcal{F}^c$ are pairwise homomorphically incomparable. We can use this, together with the fact that $\mathcal{F}^c$ has an obstruction set of treewidth $k$, to show that in fact, each $\mathfrak{B} \in \mathcal{F}^c$ has an obstruction set of treewidth $k$: for the sake of contradiction, suppose that $\mathfrak{B} \in \mathcal{F}^c$ does not have an obstruction set of treewidth $k$. Then, in particular, the set of all instances of treewidth at most $k$ that do not admit a homomorphism into $\mathfrak{B}$ is not an obstruction set for $\mathfrak{B}$. It follows that there is an instance $\mathfrak{D}$ such that $\mathfrak{D} \not\to \mathfrak{B}$ and, for all instances $\mathfrak{A}$ of treewidth at most $k$, if $\mathfrak{A} \to \mathfrak{D}$ then $\mathfrak{A} \to \mathfrak{B}$. Now consider the disjoint union $\mathfrak{D} \uplus \mathfrak{B}$. Clearly $\mathfrak{D} \uplus \mathfrak{B} \not\to \mathfrak{B}$. Furthermore, since $\mathcal{F}^c$ consists of pairwise homomorphically incomparable instances, also $\mathfrak{D} \uplus \mathfrak{B} \not\to \mathfrak{B}'$ for all other $\mathfrak{B}' \in \mathcal{F}^c$. Since $\mathcal{F}^c$ has an obstruction set of treewidth $k$, there is an instance $\mathfrak{C}$ of treewidth at most $k$ such that $\mathfrak{C} \to \mathfrak{D} \uplus \mathfrak{B}$ and such that, for all $\mathfrak{B}' \in \mathcal{F}^c$, $\mathfrak{C} \not\to \mathfrak{B}'$. In particular, $\mathfrak{C} \not\to \mathfrak{B}$. However, since $\mathfrak{C}$ has treewidth at most $k$, each connected component of $\mathfrak{C}$ that homomorphically maps to $\mathfrak{D}$ (being also of treewidth at most $k$) homomorphically maps to $\mathfrak{B}$ and, therefore, since $\mathfrak{C} \to \mathfrak{D} \uplus \mathfrak{B}$, we have that $\mathfrak{C} \to \mathfrak{B}$, a contradiction.

To summarize, we have that, for each $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$, $(\mathfrak{B}, \mathbf{b})^c$ has an obstruction set of bounded treewidth. It was shown in Feder and Vardi [1998] that, for any (unmarked) instance $\mathfrak{A}$, coCSP($\mathfrak{A}$) is datalog rewritable if and only if $\mathfrak{A}$ has an obstruction set of bounded treewidth. Therefore, we have that for each $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$, coCSP($(\mathfrak{B}, \mathbf{b})^c$) is datalog rewritable. □

Note that every set of marked instances $\mathcal{F} = \{(\mathfrak{B}_1, \mathbf{b}_1), \dots, (\mathfrak{B}_n, \mathbf{b}_n)\}$ has a subset $\mathcal{F}' = \{(\mathfrak{B}'_1, \mathbf{b}'_1), \dots, (\mathfrak{B}'_m, \mathbf{b}'_m)\}$ that consists of homomorphically incomparable marked instances and is such that coCSP($\mathcal{F}$) is equivalent to coCSP($\mathcal{F}'$). We use this observation to establish the announced lifting of Theorem 5.10.

THEOREM 5.15. *FO rewritability and datalog rewritability are* NP*-complete for generalized CSPs with marked elements.*

PROOF. To decide whether a generalized CSP with marked elements given as a set of templates $\mathcal{F} = \{(\mathfrak{B}_1, \mathbf{b}_1), \dots, (\mathfrak{B}_n, \mathbf{b}_n)\}$ is FO rewritable, it suffices to first guess a subset $\mathcal{F}' \subseteq \mathcal{F}$ and then to verify that: (i) coCSP($(\mathfrak{B}, \mathbf{b})^c$) is FO rewritable for each $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}'$;

and (ii) for each $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$ there is a $(\mathfrak{B}', \mathbf{b}') \in \mathcal{F}'$ such that $(\mathfrak{B}, \mathbf{b}) \to (\mathfrak{B}', \mathbf{b}')$. By Theorem 5.10, this can be done in NP. To see why this procedure is correct, suppose first that there is a subset $\mathcal{F}' \subseteq \mathcal{F}$ that satisfies conditions (i) and (ii). By the previous observation, there exists a subset $\mathcal{F}'' \subseteq \mathcal{F}'$ of homomorphically incomparable instances such that coCSP($\mathcal{F}''$) is equivalent to coCSP($\mathcal{F}'$) which, by (ii), is also equivalent to coCSP($\mathcal{F}$). Because of condition (i), we know that, for every instance $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}''$, coCSP($(\mathfrak{B}, \mathbf{b})^c$) is FO rewritable. We can thus apply Proposition 5.11 to conclude that coCSP($\mathcal{F}''$) or, equivalently, coCSP($\mathcal{F}$), is FO rewritable. Conversely, if coCSP($\mathcal{F}$) is FO rewritable, then we may guess a subset $\mathcal{F}' \subseteq \mathcal{F}$ of homomorphically incomparable instances such that coCSP($\mathcal{F}'$) is equivalent to coCSP($\mathcal{F}$). Condition (i) must be satisfied because of Proposition 5.11, and condition (ii) holds because coCSP($\mathcal{F}'$) and coCSP($\mathcal{F}$) are equivalent. datalog rewritability can be decided analogously. □

From Theorems 4.6 and 5.15, we obtain a NEXPTIME upper bound for deciding FO rewritability and datalog rewritability of ontology-mediated queries based on DLs and (B)AQs. The corresponding lower bounds are proven in the electronic appendix by reduction from the same NEXPTIME-hard tiling problem as used for the lower bound for query containment.

THEOREM 5.16. *FO rewritability and datalog rewritability can be decided in* NEXPTIME *for* $(\mathcal{SHIU}, \boldsymbol{AQ})$ *and* $(\mathcal{SHIU}, \boldsymbol{BAQ})$. *Both problems are* NEXPTIME*-hard for* $(\mathcal{ALC}, \boldsymbol{AQ})$ *and* $(\mathcal{ALC}, \boldsymbol{BAQ})$.

In the prior exposition, we have concentrated on deciding the existence of FO rewritings and datalog rewritings. In practice, it is often also important to actually construct such rewritings, if they exist. We briefly survey known results and then argue that the proof of Theorem 5.16 together with the existing algorithms for deciding FO rewritability and datalog rewritability of CSPs yield an approach to effectively construct FO rewritings.

For the inexpressive fragment DL-Lite of $\mathcal{ALCIH}$ which underpins the OWL 2 profile OWL2 QL and for which FO rewritings always exist, efficient FO rewriting algorithms have been developed and implemented in a number of tools [Calvanese et al. 2007; Pérez-Urbina et al. 2010; Rosati and Almatelli 2010; Chortaras et al. 2011; Rodriguez-Muro and Calvanese 2012; Kikot et al. 2012a]. Note that DL-Lite is a Horn logic, that is, it does not include any form of disjunction. For more expressive Horn ontology languages, FO rewritings and datalog rewritings are studied, for example, in Gottlob and Schwentick [2012], Eiter et al. [2012], Bienvenu et al. [2013a], and Bárány et al. [2013]. In contrast, for non-Horn ontology languages such as $\mathcal{ALC}$ and its extensions considered in this article, we are not aware of any decidability results for FO rewritability or datalog rewritability, or any approach that aims at constructing FO rewritings. A first significant step towards practical algorithms that compute datalog rewritings for such logics is presented in Grau et al. [2013].

We now sketch how our results yield an approach for effectively computing FO rewritings and datalog rewritings. For simplicity, we concentrate on the OBDA language ($\mathcal{ALC}$, BAQ). Let $Q$ be a query from ($\mathcal{ALC}$, BAQ) that has an FO rewriting. Then we can construct a concrete such rewriting by first generating a template $\mathfrak{B}$ such that $Q$ and coCSP($\mathfrak{B}$) are equivalent, using the construction from the proof of Theorem 4.6. Next, we construct a finite obstruction set $\mathcal{G}$ for CSP($\mathfrak{B}$) using Larose et al. [2007]. Every instance $\mathfrak{A} \in \mathcal{G}$ can be viewed as a Boolean conjunctive query $\mathfrak{A}^q$ in an obvious way, replacing the elements of $\mathsf{adom}(\mathfrak{A})$ with FO variables. It can be verified that the union of all CQs $\mathfrak{A}^q$, $\mathfrak{A} \in \mathcal{G}$, is an FO rewriting of $Q$. Using our results for generalized CSPs with marked instances, the sketched procedure can be generalized to queries from ($\mathcal{SHIU}$, BAQ $\cup$ AQ).

Now let $Q$ be a query from ($\mathcal{ALC}$, BAQ) that has a datalog rewriting. We again start with constructing the corresponding template $\mathfrak{B}$. It is implicit in Barto and Kozik [2009] that, if the complement of a CSP is rewritable into a datalog program, then it is rewritable into a datalog program in which each rule comprises at most $\max\{3, r\}$ distinct variables, where $r$ is the maximum arity of relations in the template. Consequently and since $\mathfrak{B}$ uses only relations of arity at most two, coCSP($\mathfrak{B}$) can be rewritten into a datalog program whose IDB relations have arity at most three and where each rule body has at most three distinct variables. It is shown in Feder and Vardi [1998] how to construct a concrete such program, called the canonical (3,3)-datalog program for $\mathfrak{B}$. Again, this procedure can be generalized to queries from ($\mathcal{SHIU}$, BAQ $\cup$ AQ).

Implemented naively, the aforesaid rewriting constructions can probably not be expected to perform well in practice since the template $\mathfrak{B}$ associated with $Q$ can be of exponential size, even for simple $\mathcal{ALC}$-ontologies. It is an interesting open research question whether the approach can be improved to yield an algorithm for constructing FO rewritings that performs well on real-world ontologies.

Modulo a minor difference in the treatment of instances that are not consistent (see Footnote 7), it follows from a result in Lutz and Wolter [2012] that FO rewritability is undecidable for ($\mathcal{ALCF}$, AQ) and ($\mathcal{ALCF}$, BAQ). In the electronic appendix, we show how to bridge the difference and further prove the undecidability of datalog rewritability.

THEOREM 5.17. *FO rewritability and datalog rewritability are undecidable for* ($\mathcal{ALCF}$, *AQ*) *and* ($\mathcal{ALCF}$, *BAQ*).

## 6. SCHEMA-FREE ONTOLOGY-MEDIATED QUERIES

To investigate the relationship between ontology-mediated queries and other database query languages, we have until now adopted from the database world the assumption that every query comes with a fixed finite data schema $\mathbf{S}$. In applications of ontology-based data access, this is not always realistic because the instances to be queried tend to not be under the control of the user. Therefore, it is of interest to also study the case where queries have to be answered without fixing a data schema in advance. In particular, this means it is not possible to exclude certain symbols that are used in the ontology and the query from occurring in the data.

In the following, we assume a countably infinite set $\mathbf{S}^\infty$ of relation symbols is fixed once and for all, that instances consist of finite sets of facts over $\mathbf{S}^\infty$, and that ontologies as well as queries can use symbols from $\mathbf{S}^\infty$ only. For FO ontologies, $\mathbf{S}^\infty$ contains infinitely many relation symbols of any arity. For DL ontologies, it contains infinitely many concept and role names (unary and binary relation symbols). A *schema-free ontology-mediated query* is an ontology-mediated query $(\mathbf{S}^\infty, \mathcal{O}, q)$, where the signatures of $\mathcal{O}$ and $q$ are contained in $\mathbf{S}^\infty$. For a given ontology-mediated query language $(\mathcal{L}, \mathcal{Q})$, we now distinguish between the *schema-free queries in* $(\mathcal{L}, \mathcal{Q})$ that take the form $(\mathbf{S}^\infty, \mathcal{O}, q)$ with $\mathcal{O}$ an ontology in $\mathcal{L}$ and $q$ a query in $\mathcal{Q}$, and the *fixed-schema queries* in $(\mathcal{L}, \mathcal{Q})$, based on a fixed finite schema $\mathbf{S}$ as investigated so far.

We investigate the extent to which the decidability and complexity results of the previous sections still hold for schema-free ontology-mediated queries. Clearly, all decidability results for query containment and all complexity upper bound results for query containment, FO rewritability, and datalog rewritability still hold since the schema-free query $(\mathbf{S}^\infty, \mathcal{O}, q)$ behaves in exactly the same way as the fixed-schema query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S} = \mathsf{sig}(\mathcal{O}) \cup \mathsf{sig}(q)$. For the same reason, if for a language $(\mathcal{L}, \mathcal{Q})$ there is no dichotomy between PTIME and NP for schema-free queries in $(\mathcal{L}, \mathcal{Q})$, then there is no such dichotomy for fixed-schema queries in $(\mathcal{L}, \mathcal{Q})$. More work is required, however, to transfer complexity lower bound proofs and to prove the converse direction

for dichotomies: if there is no dichotomy between PTIME and NP for fixed-schema queries in $(\mathcal{L}, \mathcal{Q})$, then there is no such dichotomy for schema-free queries in $(\mathcal{L}, \mathcal{Q})$.

Regarding dichotomies, we prove that Theorem 5.1 still holds for schema-free ontology-mediated queries in $(\mathcal{ALC}, \text{BAQ})$, that is, there is a dichotomy between PTIME and coNP for such queries if and only if there is such a dichotomy for fixed-schema queries from $(\mathcal{ALC}, \text{BAQ})$ which, by Theorem 5.1, is the case if and only if the Feder-Vardi conjecture holds. Using the same approach, we can also show for more expressive schema-free OBDA languages that there is a dichotomy between PTIME and coNP if and only if such a dichotomy holds for the corresponding fixed-schema language.

THEOREM 6.1. *$(\mathcal{ALC}, \text{BAQ})$ has a dichotomy between* PTIME *and* coNP *for schema-free queries iff the Feder-Vardi conjecture holds.*

PROOF. As observed earlier, if $(\mathcal{ALC}, \text{BAQ})$ has no dichotomy between PTIME and coNP for schema-free queries, then it does not have such a dichotomy for fixed-schema queries. Therefore, by Theorem 5.1, it remains to prove that every query in coCSP is polynomially equivalent to some schema-free query $(\mathbf{S}^\infty, \mathcal{O}, \exists x.A(x))$. Assume a CSP template $\mathfrak{B}$ over schema $\mathbf{S}$ is given. To construct a polynomially equivalent schema-free query, we first construct an equivalent fixed-schema query $(\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALC}, \text{BAQ})$ and then modify the construction to obtain a polynomially equivalent schema-free query. The construction of $(\mathbf{S}, \mathcal{O}, q)$ is virtually identical to the translation of a CSP template into an MDDlog program in the proof of Theorem 4.6. Take for every $d \in \mathsf{adom}(\mathfrak{B})$ a fresh concept name $A_d \notin \mathbf{S}$ and a fresh concept name $A \notin \mathbf{S}$, set $q = \exists x.A(x)$, and let

$$\mathcal{O} = \{A_d \sqcap A_{d'} \sqsubseteq A \mid d \neq d'\}\{A_d \sqcap \exists R.A_{d'} \sqsubseteq A \mid R(d,d') \notin \mathfrak{B}, R \in \mathbf{S}\} \cup$$
$$\cup \{A_d \sqcap B \sqsubseteq A \mid B(d) \notin \mathfrak{B}, B \in \mathbf{S}\} \cup \left\{\top \sqsubseteq \bigsqcup_{d \in \mathsf{adom}(\mathfrak{B})} A_d\right\}.$$

It is straightforward to show that the query defined by $\mathfrak{B}$ is equivalent to that defined by $(\mathbf{S}, \mathcal{O}, q)$. However, $\mathcal{O}$ and $q$ cannot be used without modification in the schema-free case since now the symbols $A_d$, $d \in \mathsf{adom}(\mathfrak{B})$ and $A$ have to be from $\mathbf{S}^\infty$ and can therefore occur in the input instances $\mathfrak{D}$. To resolve this issue, we replace the unary relations $A_d$ by compound concepts. In detail, take, for every $d \in \mathsf{adom}(\mathfrak{B})$, a fresh binary relation symbol $R_d$ and a unary relation symbol $A_d$, all from $\mathbf{S}^\infty$. Set $H_d = \forall R_d.A_d$ and let $\mathcal{O}'$ be the result of replacing every $A_d$ in $\mathcal{O}$ by $H_d$, for $d \in \mathsf{adom}(\mathfrak{B})$. For any instance $\mathfrak{D}$, the concepts $H_d$ can take arbitrary values in some model $\mathfrak{D}' \supseteq \mathfrak{D}$, independently from the values of $R_d$ and $A_d$ in $\mathfrak{D}$. This observation is formalized as follows.

*Fact 1.* For all $\mathbf{S}^\infty$-instances $\mathfrak{D}$ and all subsets $U_d$ of $\mathsf{adom}(\mathfrak{D})$, $d \in \mathsf{adom}(\mathfrak{B})$, there exists a model $\mathfrak{D}' \supseteq \mathfrak{D}$ such that $\mathfrak{D}' \models H_d(a)$ iff $a \in U_d$ holds for all $d \in \mathsf{adom}(\mathfrak{B})$ and all $a \in \mathsf{dom}(\mathfrak{D}')$.

We now show that deciding $q_{\mathbf{S}^\infty,\mathcal{O}',\exists x.A(x)}(\mathfrak{D}) = 0$ for $\mathbf{S}^\infty$-instances $\mathfrak{D}$ is polynomially equivalent to deciding $\mathfrak{D} \to \mathfrak{B}$ for $\mathbf{S}$-instances $\mathfrak{D}$. First assume that an $\mathbf{S}$-instance $\mathfrak{D}$ is given and that we want to decide $\mathfrak{D} \to \mathfrak{B}$. By Fact 1, this is the case iff $q_{\mathbf{S}^\infty,\mathcal{O}',\exists x.A(x)}(\mathfrak{D}) = 0$. Conversely, assume that an $\mathbf{S}^\infty$-instance $\mathfrak{D}$ is given and we want to decide $q_{\mathbf{S}^\infty,\mathcal{O}',\exists x.A(x)}(\mathfrak{D}) = 0$. If there exists a fact $A(a) \in \mathfrak{D}$, then output $q_{\mathbf{S}^\infty,\mathcal{O}',\exists x.A(x)}(\mathfrak{D}) \neq 0$. Otherwise, again by Fact 1, $q_{\mathbf{S}^\infty,\mathcal{O}',\exists x.A(x)}(\mathfrak{D}) = 0$ iff $\mathfrak{D}' \to \mathfrak{B}$ for the $\mathbf{S}$-reduct $\mathfrak{D}'$ of $\mathfrak{D}$. □

Next we consider query containment for schema-free ontology-mediated queries. Recall that complexity upper bounds and decidability results transfer from fixed-schema query languages, to schema-free query languages, since $(\mathbf{S}^\infty, \mathcal{O}, q)$ behaves in exactly the same way as the fixed-schema query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S} = \mathsf{sig}(\mathcal{O}) \cup \mathsf{sig}(q)$. For the

converse direction, we prove a general polynomial reduction of query containment for fixed-schema OBDA languages to query containment for the corresponding schema-free languages. We say that an ontology language $\mathcal{L}$ *can express emptiness* if, for every relation symbol $R$, there exists a sentence $\varphi_{R=\emptyset}$ in $\mathcal{L}$ which states that $R$ is empty. Clearly, all DLs introduced in this article, UNFO, GFO, and GNFO namely, can express emptiness.

THEOREM 6.2. *Assume $\mathcal{L}$ is a fragment of FO that can express emptiness. Then query containment for fixed-schema queries in $(\mathcal{L}, UCQ)$ and $(\mathcal{L}, AQ)$ can be polynomially reduced to query containment for schema-free queries in $(\mathcal{L}, UCQ)$ and $(\mathcal{L}, AQ)$, respectively.*

PROOF. Assume queries $Q_1 = (\mathbf{S}, \mathcal{O}_1, q_1)$ and $Q_2 = (\mathbf{S}, \mathcal{O}_2, q_2)$ are given. By renaming relation symbols in $\mathcal{O}_1, q_1, \mathcal{O}_2, q_2$ that are not from $\mathbf{S}$, we can achieve that $(\mathsf{sig}(\mathcal{O}_1) \cup \mathsf{sig}(q_1)) \cap (\mathsf{sig}(\mathcal{O}_2) \cup \mathsf{sig}(q_2)) \subseteq \mathbf{S}$. Let

$$\mathcal{O}_2' = \{\varphi_{R=\emptyset} \mid R \in (\mathsf{sig}(\mathcal{O}_1) \cup \mathsf{sig}(q_1)) \setminus \mathbf{S}\}.$$

The theorem follows if we can show that $Q_1$ is contained in $Q_2$ iff $(\mathbf{S}', \mathcal{O}_1, q_1)$ is contained in $(\mathbf{S}', \mathcal{O}_2', q_2)$, where $\mathbf{S}' = \mathbf{S} \cup \mathsf{sig}(\mathcal{O}_1) \cup \mathsf{sig}(q_1) \cup \mathsf{sig}(\mathcal{O}_2) \cup \mathsf{sig}(q_2)$. First assume $(\mathbf{S}', \mathcal{O}_1, q_1)$ is not contained in $(\mathbf{S}', \mathcal{O}_2', q_2)$ and $\mathfrak{D}$ is an $\mathbf{S}'$-instance with $\mathbf{a} \in \mathsf{cert}_{q_1,\mathcal{O}_1}(\mathfrak{D})$ and $\mathbf{a} \notin \mathsf{cert}_{q_2,\mathcal{O}_2'}(\mathfrak{D})$. Then the signature of $\mathfrak{D}$ does not contain any symbols from $(\mathsf{sig}(\mathcal{O}_1) \cup \mathsf{sig}(q_1)) \setminus \mathbf{S}$ since $\mathfrak{D}$ is consistent with $\mathcal{O}_2'$. Obtain $\mathfrak{D}'$ from $\mathfrak{D}$ by removing all facts that involve a non-$\mathbf{S}$-symbol. Clearly, we still have $\mathbf{a} \in \mathsf{cert}_{q_1,\mathcal{O}_1}(\mathfrak{D}')$ and $\mathbf{a} \notin \mathsf{cert}_{q_2,\mathcal{O}_2}(\mathfrak{D}')$ and so $(\mathbf{S}, \mathcal{O}_1, q_1)$ is not contained in $(\mathbf{S}, \mathcal{O}_2, q_2)$, as required. The converse direction is trivial. □

It follows that Theorems 5.6 and 5.7 hold for the corresponding schema-free OBDA languages as well. We close this section by considering FO rewritability and datalog rewritability and proving an analog of Theorem 5.16 for schema-free queries.

THEOREM 6.3. *FO rewritability and datalog rewritability can be decided in* NEXPTIME *for schema-free queries in $(\mathcal{SHIU}, AQ \cup BAQ)$. Both problems are* NEXPTIME*-hard for $(\mathcal{ALC}, AQ)$ and $(\mathcal{ALC}, BAQ)$.*

We also remark that, in general, the complexity of FO rewritability is not robust under moving from fixed-schema queries to schema-free queries. A concrete example is provided by the description logic $\mathcal{EL}$, which is a fragment of $\mathcal{ALC}$ and the logical underpinning of the OWL 2 profile OWL 2 EL [Baader et al. 2005]. The complexity of deciding FO rewritability of fixed-schema queries in $(\mathcal{EL}, \text{AQ})$ is EXPTIME-complete, but deciding FO rewritability of schema-free queries in $(\mathcal{EL}, \text{AQ})$ is PSPACE-complete [Bienvenu et al. 2013a]. Note that the reduction given before cannot be applied to $\mathcal{EL}$ since the concepts $\forall R.G$ are not $\mathcal{EL}$ concepts.

## 7. CONCLUSION

We have introduced a new framework for studying ontology-based data access, resting on the observation that ontology-mediated queries are closely related to disjunctive datalog, to MMSNP, and to CSP. We have shown that many fundamental questions about OBDA can be addressed within the framework, including the complexity of query containment, the complexity of FO rewritability, and of datalog rewritability, and PTIME/NP dichotomies for the data complexity of query evaluation.

There are many remaining research problems that can be explored within our framework. Immediate problems that arise from the results presented in this article include the following.

—Are FO rewritability and datalog rewritability decidable for standard OBDA languages based on UCQs such as $(\mathcal{ALC}, \text{UCQ})$? It follows from the results in Sections 3 and 4 that this problem is equivalent to the question whether FO rewritability and datalog rewritability are decidable for monadic disjunctive datalog and, equivalently, MMSNP.
—What is the computational complexity of deciding query containment for OBDA languages based on UCQs? For $(\mathcal{ALC}, \text{UCQ})$, decidability follows from the decidability of query containment for MMSNP, but tight complexity bounds do not appear to be known.
—Is a PTIME/NP dichotomy for query evaluation in (GFO, UCQ) equivalent to the Feder-Vardi conjecture, or is it possible to prove a nondichotomy result? Is query containment decidable for (GFO, UCQ)? Are FO rewritability and datalog rewritability decidable for (GFO, UCQ)? As explained in Section 4, resolving these questions is equivalent to answering them for GMSNP and $\text{MMSNP}_2$.
—What is the status of OBDA languages such as $(\mathcal{S}, \text{UCQ})$ and of OBDA languages based on ontology languages with nominals? Do they have the same expressive power as natural fragments of disjunctive datalog?
—There are several open questions regarding the succinctness of OBDA languages. For example, is there really a double-exponential succinctness gap beween $(\mathcal{ALCI}, \text{UCQ})$ (respectively, (UNFO, UCQ)) and MDDlog as well as between (GNFO, UCQ) and frontier-guarded DDlog, or can the translations be improved by one exponential? Is the exponential blowup in the translation of frontier-guarded DDlog to (GFO, UCQ) avoidable?
—In this article, we have focused on ontology-mediated queries based on atomic queries and UCQs. Other query languages frequently used in OBDA are conjunctive queries (CQs) and positive existential queries (PEQs). Since every PEQ is equivalent to a UCQ, all expressivity, decidability, and data complexity results trivially transfer from UCQs to PEQs. However, PEQs could well behave differently regarding succinctness. For CQs, the results in this article imply a dichotomy between PTIME and coNP for $(\mathcal{ALC}, \text{CQ})$ if and only if the Feder-Vardi conjecture holds. It is an interesting open problem whether there is a natural characterization of $(\mathcal{ALC}, \text{CQ})$ in terms of disjunctive datalog.

Another interesting research direction is to depart from monotonic OBDA languages by admitting some form of nonmonotonic negation in the ontology language or the query language. This typically requires replacing the certain answers semantics used in this article with a semantics tailored specifically towards nonmonotonicity; see Hernich et al. [2013] for a recent example. In particular, it would be interesting to investigate whether the resulting OBDA languages correspond to fragments of disjunctive datalog with negation [Eiter et al. 1997] in the same way as monotonic OBDA languages correspond to fragments of disjunctive datalog without negation.

## ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

## ACKNOWLEDGMENTS

We thank Benoit Larose and Libor Barto for discussions on datalog definability of CSPs, Florent Madeleine and Manuel Bodirsky for discussions on MMSNP, and Thomas Eiter for discussions on disjunctive datalog.

## REFERENCES

Bogdan Alexe, Balder Ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. 2011. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.* 36, 4.

Albert Atserias. 2005. On digraph coloring problems and treewidth duality. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*. 106–115.

Franz Baader, Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. 2010. Query and predicate emptiness in description logics. In *Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR'10)*.

Franz Baader, Sebastian Brandt, and Carsten Lutz. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. 364–369.

Franz Baader, Deborah, Diego Calvanese, Deborah L. McGuiness, Daniele Nardi, and Peter F. Patel-Schneider, Eds. 2003. *The Description Logic Handbook*. Cambridge University Press.

Jean-Francois Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michael Thomazo. 2011. Walking the complexity lines for generalized guarded existential rules. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*.

Vince Bárány, Michael Benedikt, and Balder Ten Cate. 2013. Rewriting guarded negation queries. In *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS'13)*. Springer, 98–110.

Vince Barany, Georg Gottlob, and Martin Otto. 2010. Querying the guarded fragment. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS'110)*. 1–10.

Vince Barany, Balder Ten Cate, and Martin Otto. 2012. Queries with guarded negation. *Proc. VLDB Endow.* 5, 11, 1328–1339.

Vince Barany, Balder Ten Cate, and Luc Segoufin. 2011. Guarded negation. In *Proceedings of the 38th International Colloquium on Automata, Languages, and Programming (ICALP'11)*. 356–367.

Libor Barto and Marcin Kozik. 2009. Constraint satisfaction problems of bounded width. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09)*. 595–603.

Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. 2012. Query containment in description logics reconsidered. In *Proceedings of the 13th International Conference on the Principles of Knowledge Representation and Reasoning (KR'12)*.

Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. 2013a. First-order rewritability of atomic queries in horn description logics. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. 754–760.

Meghyn Bienvenu, Balder Ten Cate, Carsten Lutz, and Frank Wolter. 2013b. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. In *Proceedings of the Symposium on Principles of Database Systems (PODS'13)*. ACM Press, New York, 213–224.

Manuel Bodirsky, Hubie Chen, and Tomás Feder. 2012. On the complexity of MMSNP. *SIAM J. Discr. Math.* 26, 1, 404–414.

Andrei A. Bulatov. 2009. Bounded relational width. http://www.cs.sfu.ca/∼abulatov/papers/relwidth.pdf.

Andrei A. Bulatov. 2011. On the CSP dichotomy conjecture. In *Proceedings of the 6th International Conference on Computer Science: Theory and Applications (CSR'11)*. 331–344.

Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2009. A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the Symposium on Principles of Database Systems (PODS'09)*. 77–86.

Andrea Calì, Georg Gottlob, and Andreas Pieris. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193, 87–128.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2006. Data complexity of query answering in description logics. In *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning (KR'06)*. 260–270.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable reasoning and efficient query answering in description logics: The DL-lite family. *J. Autom. Reason.* 39, 3, 385–429.

Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 1998. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*. 149–158.

Balder Ten Cate and Luc Segoufin. 2011. Unary negation. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS'11)*. 344–355.

Alexandros Chortaras, Despoina Trivela, and Giorgos B. Stamou. 2011. Optimized query rewriting for OWL 2 QL. In *Proceedings of the 23rd International Conference on Automated Deduction (CADE'11)*. 192–206.

Giuseppe De Giacomo and Maurizio Lenzerini. 1994. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*. 205–212.

Thomas Eiter, Wolfgang Faber, Michael Fink, and Stefan Woltran. 2007. Complexity results for answer set programming with bounded predicate arities and implications. *Ann. Math. Artif. Intell.* 51, 2–4, 123–165.

Thomas Eiter, Georg Gottlob, and Heikki Mannila. 1997. Disjunctive datalog. *ACM Trans. Database Syst.* 22, 3, 364–418.

Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. 2012. Query rewriting for horn-shiq plus rules. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'12)*.

Tomás Feder, Florent R. Madelaine, and Iain A. Stewart. 2004. Dichotomies for classes of homomorphism problems involving unary functions. *Theor. Comput. Sci.* 314, 1–2, 1–43.

Tomás Feder and Moshe Y. Vardi. 1998. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.* 28, 1, 57–104.

Jan Foniok, Jaroslav Nesetril, and Claude Tardif. 2008. Generalised dualities and maximal finite antichains in the homomorphism order of relational structures. *Euro. J. Combinator.* 29, 4, 881–899.

Ralph Freese, Marcin Kozik, Andrei Krokhin, Miklós Maróti, Ralph Mckenzie, and Ross Willard. 2009. On maltsev conditions associated with omitting certain types of local structures. http://www.math.hawaii.edu/~ralph/Classes/619/OmittingTypesMaltsev.pdf.

Georg Gottlob, Erich Grädel, and Helmut Veith. 2002. Datalog lite: A deductive query language with linear time model checking. *ACM Trans. Comput. Logics* 3, 1, 42–79.

Georg Gottlob and Thomas Schwentick. 2012. Rewriting ontological queries into small nonrecursive datalog programs. In *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning (KR'12)*.

Bernardo Cuenca Grau, Boris Motik, Giorgos Stoilos, and Ian Horrocks. 2013. Computing datalog rewritings beyond horn ontologies. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. 832–838.

Pavol Hell, Jaroslav Nešetřil, and X. Zhu. 1996. Duality and polynomial testing of tree homomorphisms. *Trans. Amer. Math. Soc.* 348, 4, 1281–1297.

André Hernich, Clemens Kupke, Thomas Lukasiewicz, and Georg Gottlob. 2013. Well-founded semantics for extended datalog and ontological reasoning. In *Proceedings of the 32nd Symposium on Principles of Database Systems (PODS'13)*. Richard Hull and Wenfei Fan, Eds., ACM Press, New York, 225–236.

Ian Horrocks and Ulrike Sattler. 1999. A description logic with transitive and inverse roles and role hierarchies. *J. Logic Comput.* 9, 3, 385–410.

Ullrich Hustadt, Boris Motik, and Ulrike Sattler. 2007. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reason.* 39, 3, 351–384.

David S. Johnson. 1990. A catalog of complexity classes. In *Handbook of Theoretical Computer Science*, MIT Press, 67–161.

Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev. 2012a. Conjunctive query answering with OWL 2 QL. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*.

Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, and Michael Zakharyaschev. 2012b. Exponential lower bounds and separation for query rewriting. In *Proceedings of the 39th International Colloquium of Automata, Languages, and Programming (ICALP'12)*. 263–274.

Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. 2010. The combined approach to query answering in DL-lite. In *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*.

Adila Krisnadhi and Carsten Lutz. 2007. Data complexity in the EL family of DLS. In *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*. 333–347.

Gábor Kun. 2007. Constraints, MMSNP, and expander structures. http://arxiv.org/abs/0706.1701v1.

Gábor Kun and Jaroslav Nesetril. 2008. Forbidden lifts (NP and CSP for combinatorialists). *Euro. J. Comb.* 29, 4, 930–945.

Benoit Larose, Cynthia Loten, and Claude Tardif. 2007. A characterisation of first-order constraint satisfaction problems. *Logical Methods Comput. Sci.* 3, 4.

Carsten Lutz. 2007. Inverse roles make conjunctive queries hard. In *Proceedings of the International Workshop on Description Logics (DL'07)*.

Carsten Lutz. 2008. The complexity of conjunctive query answering in expressive description logics. In *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR'08)*. 179–193.

Carsten Lutz and Frank Wolter. 2012. Non-uniform data complexity of query answering in description logics. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*.

Florent R. Madelaine. 2009. Universal structures and the logic of forbidden patterns. *Logical Methods Comput. Sci.* 5, 2.

Florent R. Madelaine and Iain A. Stewart. 2007. Constraint satisfaction, logic and forbidden patterns. *SIAM J. Comput.* 37, 1, 132–163.

Boris Motik. 2006. Reasoning in description logics using resolution and deductive databases. https://www.cs.ox.ac.uk/boris.motik/pubs/motik06PhD.pdf.

Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. 2010. Tractable query answering and rewriting under description logic constraints. *J. Appl. Logic* 8, 2,186–209.

Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking data to ontologies. *J. Data Semant.* 10, 133–173.

Mariano Rodriguez-Muro and Diego Calvanese. 2012. High performance query answering over DL-lite ontologies. In *Proceedings of the 13$^{th}$ International Conference on Principles of Knowledge Representation and Reasoning (KR'12).* 308–318.

Riccardo Rosati and Alessandro Almatelli. 2010. Improving query answering over DL-lite ontologies. In *Proceedings of the 12$^{th}$ International Conference on thePrinciples of Knowledge Representation and Reasoning (KR'10).* 290–300.

Benjamin Rossman. 2008. Homomorphism preservation theorems. *J. ACM* 55, 3.

Sebastian Rudolph, Markus Krotzsch, and Pascal Hitzler. 2012. Type-elimination-based reasoning for the description logic shiqbs using decision diagrams and disjunctive datalog. *Logical Methods Comput. Sci.* 8, 1.

Frantisek Simancik. 2012. Elimination of complex RIAS without automata. In *Proceedings of the 25$^{th}$ International Workshop on Description Logics (DL'12).*

W3C. 2009. OWL 2 web ontology language: Document overview. W3C recommendation. http://www.w3.org/TR/owl2-overview/.

Received October 2013; revised June 2014; accepted August 2014