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

Meghyn Bienvenu
CNRS & Université Paris Sud
Orsay, France

Balder ten Cate
UC Santa Cruz
Santa Cruz, CA, USA

Carsten Lutz
University of Bremen
Bremen, Germany

Frank Wolter
University of Liverpool
Liverpool, UK

ABSTRACT

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 paper, 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 paper are three-fold. First, we characterize the expressive power of ontology-mediated queries in terms of fragments of disjunctive datalog. 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*; H.2.5 [**Database Management**]: Heterogeneous Databases

Keywords

Ontology-Based Data Access; Query Answering; Query Rewriting

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 semantics, taking into account knowledge provided by an ontology [19, 18, 16]. 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 also central to query answering in the semantic web. OBDA can also be used to enrich the data schema (that is, the relation symbols used 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 bio-medical 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 [40]. 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, which we call an *ontology-mediated query*. OBDA can then be described as the problem of answering ontology-mediated queries. The database queries used in OBDA are typically unions of conjunctive queries, while the ontologies are typically 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 [17, 29, 32]. In particular, relevant 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-based queries using traditional database management systems. This is considered one of the most promising approaches for OBDA, and is currently the subject of significant research activity, see for example [18, 28, 30, 31, 42].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PODS'13, June 22–27, 2013, New York, New York, USA.
Copyright 2013 ACM 978-1-4503-2066-5/13/06 ...$15.00.

The main aims of this paper 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 and (ii) to make progress towards complete and decidable classifications of ontology-mediated queries, with respect to their data complexity, as well as with respect to FO-rewritability and datalog-rewritability.

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 than can occur 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 $\mathcal{O}$. 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$^{\pm}$ [15], namely the guarded fragment (GF), 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 query languages 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, (GF,UCQ) refers to ontology-mediated queries in which $\mathcal{O}$ is a GF-ontology and q is a UCQ. We refer to such query languages $(\mathcal{L}, \mathcal{Q})$ as *ontology-mediated query languages (or, 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 ($\mathcal{ALC}$,UCQ) has the same expressive power as monadic disjunctive datalog (abbreviated MDDlog) and that ($\mathcal{ALC}$,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 (GF,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 α in the head of a rule, there is an atom β in the rule body that contains all variables from α.

In Sections 4 and 5, 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 [24] states that 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. In other words, the conjecture asserts that there are no CSPs which 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 [24]. In particular, it was shown in [24, 33] 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 observe that ($\mathcal{ALC}$,UCQ) and many other OBDA languages based on UCQs have the same expressive power as the query language coMMSNP, consisting of all queries whose complement is definable by an MMSNP formula with free variables. In the spirit of descriptive complexity theory, we say that ($\mathcal{ALC}$,UCQ) *captures* coMMSNP. In fact, this result is a consequence of the results in Section 3 and the observation that MDDlog has the same expressive power as coMMSNP. It has fundamental consequences regarding the data complexity of ontology-mediated queries and the containment problem for such queries, which we describe next.

First, 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. To appreciate this result, recall that considerable effort has been directed towards identifying tractable classes of ontology-mediated queries. 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 result ties this task to proving the Feder-Vardi conjecture. Significant progress has been made in understanding the complexity of CSPs and MMSNPs [14, 12, 34], and the connection established in this paper facilitates the transfer of techniques and results from CSP and MMSNP in order to analyze the data complexity of query evaluation in ($\mathcal{ALC}$,UCQ). We also consider the standard extension $\mathcal{ALCF}$ of $\mathcal{ALC}$ with functional roles and note that, for query evaluation in ($\mathcal{ALCF}$,AQ), there is no dichotomy between PTIME and CONP unless PTIME = NP.

To establish a counterpart of (GF,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, (GF,UCQ) and (GNFO,UCQ) capture coGMSNP. We observe that GMSNP has the same expressive power as the extension MMSNP$_2$ of MMSNP proposed in [37]. It follows from our results in Section 3 that GMSNP (and thus MMSNP$_2$) is strictly more expressive than MMSNP, closing an open problem from [37]. We leave it as an open problem whether GMSNP is computationally as well-behaved as MMSNP, that is, whether there is a dichotomy between PTIME and NP if the Feder-Vardi conjecture holds.

The second application of the connection between OBDA and MMSNP concerns query containment. It was shown in [24] 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 ($\mathcal{ALC}$,UCQ) and (GF,UCQ). Note that this refers to a very general form of query containment in OBDA, as recently introduced and studied in [10]. For ($\mathcal{ALCF}$,AQ), this problem (and every other decision problem discussed below) turns out to be undecidable.

In Section 5, 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): we prove that ($\mathcal{ALC}$,BAQ) captures the query language that consists of all Boolean queries definable as the complement of a CSP. Similarly ($\mathcal{ALC}$,AQ) extended with the universal role 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 enriched with a constant symbol. We then proceed to transfer results from the CSP literature to the ontology-mediated query languages ($\mathcal{ALC}$, BAQ) and ($\mathcal{ALC}$, AQ). First we immediately obtain that the existence of a PTIME/CONP dichotomy for these ontology-mediated query languages is equivalent to the Feder-Vardi conjecture. Then we show that query containment is not only decidable (as we could already conclude from the connection with coMMSNP described in Section 4), but, in fact, NEXPTIME-complete. Finally, taking advantage of recent results for CSPs [35, 26, 13], we are able to show that FO-rewritability and datalog-rewritability, as properties of ontology-mediated queries, are decidable and NEXPTIME-complete for ($\mathcal{ALC}$, AQ) and ($\mathcal{ALC}$,BAQ).

The results in Sections 4 and 5 just summarized are actually proved not only for $\mathcal{ALC}$, but also for several of its extensions. This relies on the equivalences between DL-based OBDA-languages established in Section 3.

Related Work A connection between query answering in DLs and the negation-free fragment of disjunctive datalog was first discovered and utilized in the influential [39, 29], see also [44]. This research is concerned with answer-preserving translations of ontology-mediated queries into disjunctive datalog. In contrast to the current paper, it does not consider the expressive power of ontology-mediated queries, nor their descriptive complexity. A connection between DL-based OBDA and CSPs was first found and exploited in [36], in a setup that is different from the one studied in this paper. In particular, instead of focusing on ontology-mediated queries that consist of a data schema, an ontology, and a database query, [36] 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 paper, and does not consider the expressive power and descriptive complexity of queries used in OBDA.

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 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 of $\mathfrak{D}$. We will frequently use boldface notation for tuples, such as in $\mathbf{a} = (a_1, \ldots, a_n)$, and we denote by $()$ the empty tuple.

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 names from $\mathbf{S}$ (and, possibly, equality) give rise to the n-ary query $q_{\varphi,\mathbf{S}}$, defined by setting for all $\mathbf{S}$-instances $\mathfrak{D}$,

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

To simplify exposition, we assume that FO-queries do not contain constants. We use FOQ to denote the set of all first-order queries, as defined above. Similarly, we use CQ and UCQ to refer to the class of conjunctive queries and unions of conjunctive queries, defined as usual and allowing the use of equality. AQ denotes the set of *atomic queries*, which are of the form $A(x)$ with A a unary relation symbol. Each of these is called a *query language*, which is defined abstractly as a set of queries. Besides FOQ, CQ, UCQ, and AQ, we consider various other query languages 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 $\mathbf{S}$-instance $\mathfrak{D}$, 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 $\mathcal{L}$. We introduce various ontology languages throughout the paper, 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 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. As explained in the introduction, this allows the ontology to enrich the schema of the query q. Of course, we do not require that every relation of the data schema needs to 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 interpreted as a query over $\mathbf{S}$.

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 dom is a non-empty 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 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 $\mathbf{a} \in q(\mathfrak{D}')$.

Note that all ontology languages considered in this paper enjoy finite controllability, meaning that finite relational structures can be replaced with unrestricted ones without changing the certain answers to unions of conjunctive queries [6, 7].

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 $\mathfrak{D}$. 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*).

Example 1 *The left-hand side of Table 1 shows an ontology $\mathcal{O}$ that is formulated in the guarded fragment of FO. Consider the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q)$ with data schema and query*

$$\mathbf{S} = \{\mathsf{ErythemaMigrans}, \mathsf{LymeDisease}, \mathsf{HereditaryPredisposition}, \mathsf{finding}, \mathsf{diagnosis}, \mathsf{parent}\}$$
$$q(x) = \exists y(\ \mathsf{diagnosis}(x, y) \wedge \mathsf{BacterialInfection}(y)\ ).$$

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

finding(pat1, jan12find1)	ErythemaMigrans(jan12find1)
diagnosis(pat2, may7diag2)	Listeriosis(may7diag2)

$\forall x(\ \exists y(\mathsf{finding}(x,y) \wedge \mathsf{ErythemaMigrans}(y)) \rightarrow \exists y(\mathsf{diagnosis}(x,y) \wedge \mathsf{LymeDisease}(y))\ )$	$\exists\mathsf{finding}.\mathsf{ErythemaMigrans} \sqsubseteq \exists\mathsf{diagnosis}.\mathsf{LymeDisease}$
$\forall x(\ (\mathsf{LymeDisease}(x) \vee \mathsf{Listeriosis}(x)) \rightarrow \mathsf{BacterialInfection}(x)\ )$	$\mathsf{LymeDisease} \sqcup \mathsf{Listeriosis} \sqsubseteq \mathsf{BacterialInfection}$
$\forall x(\ \exists y.(\mathsf{HereditaryDisposition}(y) \wedge \mathsf{parent}(x,y)) \rightarrow \mathsf{HereditaryDisposition}(y))\ )$	$\exists\mathsf{parent}.\mathsf{HereditaryDisposition} \sqsubseteq \mathsf{HereditaryDisposition}$

Table 1: Example ontology, presented in (the guarded fragment of) first-order logic and the DL $\mathcal{ALC}$

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

Table 2: First-order translation of $\mathcal{ALC}$-concepts

we have $\mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D}) = \{\mathsf{pat1}, \mathsf{pat2}\}$.

Description Logics for Specifying Ontologies. In description logic, schemas are generally restricted to relations of arity one and two, called *concept names* and *role names*, respectively. For brevity, 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 paper.

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 2. An $\mathcal{ALC}$-ontology $\mathcal{O}$ then translates into the set of FO-sentences $\mathcal{O}^* = \{\forall x.(C^*(x) \rightarrow D^*(x)) \mid C \sqsubseteq D \in \mathcal{O}\}$. On the right-hand side of Table 1, we show the $\mathcal{ALC}$-version of the guarded fragment ontology displayed on the left-hand side. Note that, although the translation is equivalence-preserving in this case, in general, the guarded fragment is a more expressive ontology language than $\mathcal{ALC}$. Throughout the paper, we do not explicitly distinguish between a DL ontology and its translation into FO.

We remark that, from a DL perspective, the above 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 paper.

Example 2 *Let $\mathcal{O}$ and $\mathbf{S}$ be as in Example 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 $q_2(x) = \mathsf{HereditaryDisposition}(x)$, the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q_2)$ is equivalent to the query defined by the datalog program*

$$\begin{array}{lll} P(x) & \leftarrow & \mathsf{HereditaryDisposition}(x) \qquad \mathsf{goal}(x) \leftarrow P(x) \\ P(x) & \leftarrow & \mathsf{parent}(y,x) \wedge P(y) \end{array}$$

but not to any first-order query.

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* ρ 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 ρ, and to $R_1(\mathbf{y}_1) \wedge \ldots \wedge R_n(\mathbf{y}_n)$ as the *body* of ρ. Every variable that occurs in the head of a rule ρ is required to also occur in the body of ρ. Empty rule heads are denoted $\bot$. A *disjunctive datalog (DDlog) program* Π is a finite set of disjunctive datalog rules with a selected *goal predicate* 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* Π is the arity of the goal relation. Relation symbols that occur in the head of at least one rule of Π are *intensional (IDB) predicates* of Π, and all remaining relation symbols in Π are *extensional (EDB) predicates*.

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

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

Here, $\mathsf{Mod}(\Pi)$ denotes the set of all instances over $\mathbf{S}'$ that satisfy all rules in Π, with $\mathbf{S}'$ the set of all IDB and EDB predicates in Π. Note that the DDlog programs considered in this paper are negation-free. Restricted to this fragment, there is no difference between the different semantics of DDlog studied e.g. in [21].

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

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

3.1 Ontologies Specified in Description Logics

We show that ($\mathcal{ALC}$,UCQ) has the same expressive power as MDDlog and identify a fragment of MDDlog that has the same expressive power as ($\mathcal{ALC}$,AQ). In addition, we 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.[1]

($\mathcal{ALC}$,UCQ) and MDDlog. The first main result of this section is Theorem 1 below, which relates ($\mathcal{ALC}$,UCQ) and MDDlog.

Theorem 1 *($\mathcal{ALC}$,UCQ) and MDDlog have the same expressive power.*

[1]In fact, this assumption is inessential for Theorems 1 and 3 (which speak about UCQs), but required for Theorems 2, 4, and 5 (which speak about AQs) to hold.

Proof. (sketch) We start with giving some intuitions about answering ($\mathcal{ALC}$,UCQ) queries which guide our translation of such queries into MDDlog programs. Recall that the definition of certain answers to an ontology-mediated query on an instance $\mathfrak{D}$ involves a quantification over all models of $\mathcal{O}$ which extend $\mathfrak{D}$. It turns out that in the case of ($\mathcal{ALC}$,UCQ) queries (and, as we will see later, more generally for (UNFO,UCQ) queries), it suffices to consider a particular type of extensions of $\mathfrak{D}$ that we term *pointwise extensions*. Intuitively, such an extension of $\mathfrak{D}$ corresponds to attaching domain-disjoint structures to the elements of $\mathfrak{D}$. Formally, for instances $\mathfrak{D} \subseteq \mathfrak{D}'$, we call $\mathfrak{D}'$ a pointwise extension of $\mathfrak{D}$ if $\mathfrak{D}' \setminus \mathfrak{D}$ is the union of instances $\{\mathfrak{D}'_a \mid a \in \mathsf{adom}(\mathfrak{D})\}$ such that $\mathsf{adom}(\mathfrak{D}'_a) \cap \mathsf{adom}(\mathfrak{D}) \subseteq \{a\}$ 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 $\mathcal{O}$ which are pointwise 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 query q' obtained from q by identifying all variables that h sends to the same element, and to a decomposition of q' into a collection of components $q'_0, \dots, q'_k$ where the 'core component' q'_0 comprises all atoms of q' whose variables h sends to elements of $\mathfrak{D}$ and for each $\mathfrak{D}'_a$ in the image of h, there is a 'non-core component' q'_i, $1 \leq i \leq k$, such that q'_i comprises all atoms of q' whose variables h sends to elements of $\mathfrak{D}'_a$. Note that the non-core components are pairwise variable-disjoint and share at most one variable with the core component.

We now detail the translation from an ontology-mediated query $(\mathbf{S}, \mathcal{O}, q) \in (\mathcal{ALC},\text{UCQ})$ into an equivalent MDDlog program. Let $\mathsf{sub}(\mathcal{O})$ be the set of subconcepts (that is, syntactic subexpressions) of concepts that occur in $\mathcal{O}$, and let $\mathsf{cl}(\mathcal{O}, q)$ denote the union of $\mathsf{sub}(\mathcal{O})$ and the set of all CQ that have at most one free variable, use only symbols from q, and whose number of atoms is bounded by the number of atoms of q. A *type* (for $\mathcal{O}$ and q) is a subset of $\mathsf{cl}(\mathcal{O}, q)$. The CQs present in $\mathsf{cl}(\mathcal{O}, q)$ include all potential 'non-core components' from the intuitive explanation above. The free variable of a CQ in $\mathsf{cl}(\mathcal{O}, q)$ (if any) represents the overlap between the core component and the non-core component.

We introduce a fresh unary relation symbol P_τ for every type τ, 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_τ will be used as IDB relations, and the symbols from $\mathbf{S}$ will be the EBD relations.

We will 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

$$\begin{aligned} \tau \quad = \quad & \{q' \in \mathsf{cl}(\mathcal{O}, q) \mid q' \text{ Boolean }, \mathfrak{B} \models q'\} \cup \\ & \{C \in \mathsf{cl}(\mathcal{O}, q) \mid C \text{ unary}, \mathfrak{B} \models C[d]\}. \end{aligned}$$

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 $\delta(x_1, \dots, x_n)$ 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 Π consists of the following collections of rules:

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

Intuitively, these rules 'guess' a pointwise extension $\mathfrak{D}'$ of $\mathfrak{D}$. Specifically, the types P_τ 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 determine those non-core component queries that matched in the $\mathfrak{D}'_a$ parts. The second line ensures coherence of the guesses and the last line guarantees that q has the required match in $\mathfrak{D}'$. It is proved in the full version of this paper that the MDDlog query q_Π is indeed equivalent to $(\mathbf{S}, \mathcal{O}, q)$.

For the converse direction, let Π be an MDDlog program. For each unary IDB relation A of Π, we introduce two fresh unary relations, 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

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

Let q be the union of (i) all conjunctive queries that constitute the body of a goal rule, as well as (ii) all 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

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

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 Π, is equivalent to the query defined by Π. ❑

$\mathcal{ALC}$ with Atomic Queries. We characterize ($\mathcal{ALC}$,AQ) by a fragment of MDDlog. This query language has the same expressive power as the OBDA language ($\mathcal{ALC}$,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, ($\mathcal{ALC}$,AQ) also has the same expressive power as ($\mathcal{ALC}$, 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 that 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.

Theorem 2 *($\mathcal{ALC}$,AQ) has the same expressive power as unary connected simple MDDlog.*

Proof. (sketch) The translation from ($\mathcal{ALC}$,AQ) to unary connected simple MDDlog queries is a modified version of the translation given in the proof of Theorem 1. Assume that $(\mathbf{S}, \mathcal{O}, q)$ with $q = A(x)$ is given. We now take types to be subsets of $\mathsf{sub}(\mathcal{O})$ and then define diagrams exactly as before (with $k = 2$). The MDDlog program Π consists of the following rules:

$$\begin{array}{rl} \bigvee\limits_{\tau \subseteq \mathsf{sub}(\mathcal{O})} P_\tau(x) \leftarrow \mathsf{adom}(x) & \\ \bot \leftarrow \delta(\mathbf{x}) & \text{for all non-realizable diagrams } \delta(\mathbf{x}) \\ & \text{of the form } P_{\tau_1}(x) \wedge P_{\tau_2}(x), \\ & P_\tau(x) \wedge A(x), \text{ or} \\ & P_{\tau_1}(x_1) \wedge S(x_1, x_2) \wedge P_{\tau_2}(x_2) \\ \mathsf{goal}(x) \leftarrow P_\tau(x) & \text{for all } P_\tau \text{ with } A \in P_\tau \end{array}$$

Clearly, Π is unary, connected, and simple. Equivalence of the queries $(\mathbf{S}, \mathcal{O}, q)$ and q_Π is proved in the full version of this paper.

Conversely, let Π be a unary connected simple MDDlog program. It is easy to rewrite each rule of Π 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 $\exists R.\top \sqsubseteq \mathsf{goal}$ and $P_1(x) \vee P_2(y) \leftarrow R(x,y) \wedge A(x) \wedge B(y)$ is rewritten into $A \sqcap \exists R.(B \sqcap \neg P_2) \sqsubseteq P_1$. Let $\mathcal{O}$ be the resulting ontology and let $q = \mathsf{goal}(x)$. Then the query q_Π is equivalent to the query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S}$ consists of the EDB relations in Π. ❑

Note that the connectedness condition is required since one cannot express MDDlog rules such as $\mathsf{goal}(x) \leftarrow \mathsf{adom}(x) \wedge A(y)$ with $y \neq x$ in ($\mathcal{ALC}$,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 ($\mathcal{ALC}$,AQ).

Extensions of $\mathcal{ALC}$. We identify several standard extensions of ($\mathcal{ALC}$,UCQ) and ($\mathcal{ALC}$,AQ) that have the same expressive power, and some that do not. We introduce the relevant extensions only briefly and refer to [4] for more details.

$\mathcal{ALCI}$ is the extension of $\mathcal{ALC}$ in which one can state that a role name R is the *inverse* of a role name S, that is, $\forall xy(R(x,y) \leftrightarrow S(y,x))$; $\mathcal{ALCH}$ is the extension in which one can state that a role name R is *included* in a role name S, that is, $\forall xy(R(x,y) \rightarrow S(x,y))$; $\mathcal{S}$ is the extension of $\mathcal{ALC}$ in which one can require some roles names to be interpreted as *transitive relations*; $\mathcal{ALCF}$ is the extension in which one can state that some role names are interpreted as *partial functions*; and $\mathcal{ALCU}$ is the extension with the *universal role* U, interpreted as $\mathsf{dom} \times \mathsf{dom}$ in any relational structure $\mathfrak{B}$ with domain dom. Note that U should be regarded as a logical symbol and is not a member of any schema. All these means of expressivity are included in the OWL2 DL profile of the W3C-standardized ontology language OWL2 [47].

We use the usual naming scheme to denote combinations of these extensions, for example $\mathcal{ALCHI}$ for the union of $\mathcal{ALCH}$ and $\mathcal{ALCI}$ and $\mathcal{SHI}$ for the union of $\mathcal{S}$ and $\mathcal{ALCHI}$. The following result summarizes the expressive power of extensions of $\mathcal{ALC}$.

Theorem 3

1. *($\mathcal{ALCHIU}$,UCQ) has the same expressive power as MDDlog and as ($\mathcal{ALC}$,UCQ).*
2. *($\mathcal{S}$,UCQ) and ($\mathcal{ALCF}$,UCQ) are strictly more expressive than ($\mathcal{ALC}$,UCQ).*

Proof. (sketch) In Point 1, we start with ($\mathcal{ALCIU}$,UCQ), for which the result follows from Theorem 6 in Section 3.2 since $\mathcal{ALCIU}$ is a fragment of UNFO. Role inclusions $\forall xy(R(x,y) \rightarrow S(x,y))$ do not add expressive power since they can be simulated by adding to the ontology the inclusions $\exists R.C \sqsubseteq \exists S.C$ for all $C \in \mathsf{sub}(\mathcal{O})$, and replacing every atom $S(x,y)$ in the UCQ by $R(x,y) \vee S(x,y)$.

For Point 2, we separate ($\mathcal{S}$,UCQ) from ($\mathcal{ALC}$,UCQ) by showing that the following ontology-mediated query $(\mathbf{S}_1, \mathcal{O}_1, q_1)$ cannot be expressed in ($\mathcal{ALC}$,UCQ): $\mathbf{S}_1$ consists of two role names R and S, $\mathcal{O}_1$ states that these role names are both transitive, and $q_1 = \exists xy(R(x,y) \wedge S(x,y))$. For ($\mathcal{ALCF}$,UCQ), we show that $(\mathbf{S}_2, \mathcal{O}_2, q_2)$ cannot be expressed in ($\mathcal{ALC}$,UCQ), where $\mathbf{S}_2$ consists of role name R and concept name A, $\mathcal{O}_2$ states that R is functional, and $q_2 = A(x)$. Detailed proofs are provided in the full version of this paper. They rely on a characterization of ($\mathcal{ALC}$,UCQ) in terms of colored forbidden patterns [38], which is a by-product of the connection between ($\mathcal{ALC}$,UCQ) and MMSNP that will be established in Section 4. ❑

The next result is interesting when contrasted with Point 2 of Theorem 3: when ($\mathcal{ALC}$,UCQ) is replaced with ($\mathcal{ALC}$,AQ), then the addition of transitive roles no longer increases the expressive power.

Theorem 4 *($\mathcal{ALC}$,AQ) has the same expressive power as ($\mathcal{SHI}$,AQ).*

Proof. (sketch) The proof of Theorem 2 given above actually shows that unary connected simple MDDlog is at least as expressive as ($\mathcal{ALCI}$,AQ). Thus, ($\mathcal{ALC}$,AQ) has the same expressive power as ($\mathcal{ALCI}$,AQ). Now it is folklore that in $\mathcal{ALCI}$ transitive roles can be replaced by certain concept inclusions without changing the certain answers to atomic queries. This can be done similarly to the elimination of role inclusions in the proof above, see [39, 45]. Thus ($\mathcal{ALCI}$,AQ) has the same expressive power as ($\mathcal{SHI}$,AQ), and the result follows. ❑

It follows from [45] that this observation can be extended to all complex role inclusions that are admitted in the description logic $\mathcal{SROIQ}$. In contrast, the addition of the universal role on the side of the OBDA query language extends the expressive power of ($\mathcal{ALC}$,AQ). Namely, it corresponds, on the MDDlog side, to dropping the requirement that rule bodies must be connected. For example, the MDDlog query $\mathsf{goal}(x) \leftarrow \mathsf{adom}(x) \wedge A(y)$ can then be expressed using the ontology $\mathcal{O} = \{\exists U.A \sqsubseteq \mathsf{goal}\}$ and the AQ $\mathsf{goal}(x)$.

Theorem 5 *($\mathcal{ALCU}$,AQ) and ($\mathcal{SHIU}$,AQ) both have the same expressive power as unary simple MDDlog.*

We close this section with a brief remark about *Boolean atomic queries* (BAQs), that is, queries of the form $\exists x.A(x)$, where A is a unary relation symbol. Such queries will be considered in Section 5. It is possible to establish modified versions of Theorems 2 to Theorem 5 above in which AQs are replaced by BAQs and unary goal predicates by 0-ary goal-predicate, respectively.

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.[2] To overcome this restriction, we consider the guarded fragment of first-order logic and the unary-negation fragment of first-order logic [6, 46]. Both generalize the description logic $\mathcal{ALC}$ in different ways. We also consider their natural common generalization, the guarded negation fragment of first-order logic [7]. Our results from the previous subsection turn out to generalize to all these fragments. We start by considering the unary negation fragment.

The *unary-negation fragment of first-order logic* (UNFO) [46] is the fragment of first-order logic that consists of those formulas that are 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 UNFO, whereas $\exists xy\neg R(x,y)$ does not. It is easy to show that every $\mathcal{ALC}$-TBox is equivalent to a UNFO sentence.

Theorem 6 *(UNFO,UCQ) has the same expressive power as MDDlog.*

[2]There are actually a few DLs that can handle relations of unrestricted arity, such as those presented in [19]. We do not consider such DLs in this paper, but remark that large fragments of them can be translated into UNFO.

Proof. (sketch) The translation from MDDlog to (UNFO,UCQ) is given by Theorem 1. Here, we provide the translation from (UNFO,UCQ) to MDDlog. Let $Q = (\mathbf{S}, \mathcal{O}, q) \in$ (UNFO,UCQ) be given. We assume that $\mathcal{O}$ is a single UNFO sentence that is in the normal form generated by the following 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 ψ_i is either a relational atom or a formula with at most one free variable generated by the same grammar, and the free variables in ψ_i are among $x, \mathbf{y}$. Note that no equality is used and that all generated formulas have at most one free variable. Easy syntactic manipulations show that every UNFO-formula with at most one free variable is equivalent to a disjunction of formulas generated by the above grammar. In the case of $\mathcal{O}$, we may furthermore assume that it is a *single* such sentence, rather than a disjunction, 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 $\mathsf{sub}(\mathcal{O})$ be the set of all subformulas of $\mathcal{O}$ with at most one free variable z (we apply a one-to-one renaming of variables as needed to ensure that each formula in $\mathsf{sub}(\mathcal{O})$ with a free variable has the same free variable z). Let k be the maximum of the number of variables in $\mathcal{O}$ and the number of variables in q. We denote by $\mathsf{cl}_k(\mathcal{O})$ 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 $\mathbf{y} = (y_1, \dots, y_m)$, $m \leq k$, where each ψ_i is either a relational atom or is of the form $\chi(x)$ or $\chi(y_i)$, for $\chi(z) \in \mathsf{sub}(\mathcal{O})$. A *type* τ is a subset of $\mathsf{cl}_k(\mathcal{O})$; the set of all types is denoted $\mathsf{type}(\mathcal{O})$.

We introduce a fresh unary relation symbol P_τ for each type τ, and we denote by $\mathbf{S}'$ the schema that extends $\mathbf{S}$ with these additional relations. As before, we call a structure $\mathfrak{B}$ over $\mathbf{S}' \cup \mathsf{sig}(\mathcal{O})$ type-coherent if for all types τ and elements d in the domain of $\mathfrak{B}$, we have $P_\tau(d) \in \mathfrak{B}$ just in the case that τ is the (unique) type realized at d in $\mathfrak{B}$. Diagrams, realizability, and "implying q" are defined as in the proof of Theorem 1. It follows from [46] that it is decidable whether a diagram implies a query, and whether a diagram is realizable. The MDDlog program Π is defined as in the proof of Theorem 1, except that now in the first rule, τ ranges over types in $\mathsf{type}(\mathcal{O})$. In the full version of this paper, we prove that the resulting MDDlog query q_Π is equivalent to Q. ❏

Next, we consider the *guarded fragment of first-order logic* (GF). It 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, α is an atomic formula (a "guard") that contains all free variables of φ. 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 $\mathcal{ALC}$ TBox into an equivalent sentence of GF.

It turns out that the OBDA language (GF, UCQ) is strictly more expressive than MDDlog.

Proposition 1 *The Boolean query*

($\dagger$) *there are $a_1, \dots, a_n, b$, 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 (GF,UCQ) and not in MDDlog.

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

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

We show in the full version of this paper that Q is not expressible in MDDlog using the colored forbidden patterns characterization mentioned in the proof sketch of Theorem 3. ❏

As fragments of first-order logic, the unary-negation fragment and the guarded fragment are incomparable in expressive power. They have a common generalization, which is known as the guarded-negation fragment (GNFO) [8]. 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 α is an atomic formula that contains all the variables of φ. 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 GF. Specifically, (GF, 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 [27]. A weaker notion of guardedness, which we call here *frontier-guardedness*, inspired by [5, 7], requires that, for each atom α in the head of the rule, there is an atom β in the rule body such that all variables that occur in α occur also in β. 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.

Theorem 7 *(GF,UCQ) and (GNFO,UCQ) have the same expressive power as frontier-guarded DDlog.*

Theorem 7 is proved in the full version of this paper via translations from (GNFO,UCQ) to frontier-guarded DDlog and back that are along the same lines as the translations from (UNFO,UCQ) to MDDlog and back. In addition, we use a result from [8] to obtain a translation from (GNFO,UCQ) to (GF,UCQ).

4. OBDA AND MMSNP

We 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 then use this connection to transfer results from MMSNP to OBDA languages with UCQs, linking the data complexity of these languages to the Feder-Vardi conjecture and establishing decidability of query containment. We also propose GMSNP, an extension of MMSNP inspired by frontier guarded DDlog, and show that (GF,UCQ) and (GNFO,UCQ) capture coGMSNP, and that GMSNP has the same expressive power as a previously proposed extension of MMSNP called MMSNP$_2$.

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 $X_1, \dots, X_n$ monadic second-order (SO) variables, $x_1, \dots, x_m$ FO-variables, and φ 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 α_i is of the form $X_i(\mathbf{x})$, $R(\mathbf{x})$ (with $R \in \mathbf{S}$), or $x = y$, and each β_i is of the form $X_i(\mathbf{x})$. 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 Φ over schema $\mathbf{S}$ with n free variables gives rise to a query

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

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 Φ is a sentence. We observe that the resulting query language *coMMSNP* has the same expressive power as MDDlog.

Proposition 2 *coMMSNP and MDDlog have the same expressive power.*

Proof. Let $\Phi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ be an MMSNP formula with free variables $y_1, \dots, y_k$, and let $q_{\Phi,\mathbf{S}} \in$ coMMSNP be the corresponding query. We can assume w.l.o.g. that all implications $\psi = \alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta_1 \vee \cdots \vee \beta_m$ in φ satisfy the following properties: (i) $n > 0$ and, (ii) each variable that occurs in a β_i atom also occurs in an α_i atom. In fact, we can achieve both (i) and (ii) by replacing violating implications ψ with the set of implications ψ' that can be obtained from ψ by adding, for each variable x that occurs only in the head of ψ, an atom $S(\mathbf{x})$ where S is a predicate that occurs in Φ and $\mathbf{x}$ is a tuple of variables that contains x once and otherwise only fresh variables that do not occur in Φ. Define an MDDlog program Π_Φ that consists of all implications in φ whose head is not $\perp$ plus a rule

$$\mathsf{goal}(y_1, \dots, y_k) \leftarrow \vartheta \wedge \mathsf{adom}(y_1) \wedge \cdots \wedge \mathsf{adom}(y_k)$$

for each implication $\vartheta \rightarrow \perp$ in φ. It can be proved that $q_{\Phi,\mathbf{S}} = q_{\Pi_\Phi,\mathbf{S}}$ for all schemas $\mathbf{S}$. Finally, it is straightforward to remove the equalities from the rule bodies in Π_Φ.

Conversely, let Π be a k-ary MDDlog program and assume w.l.o.g. that each rule uses a disjoint set of variables. Reserve fresh variables $y_1, \dots, y_k$ as free variables for the desired MMSNP formula, and let $X_1, \dots, X_n$ be the IDB predicates in Π and $x_1, \dots, x_m$ the FO-variables in Π that do not occur in the goal predicate. Set $\Phi_\Pi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ where φ is the conjunction of all non-goal rules in Π plus the implication $\vartheta' \rightarrow \perp$ for each rule $\mathsf{goal}(\mathbf{x}) \leftarrow \vartheta$ in Π. Here, ϑ' is obtained from ϑ by replacing each variable $x \in \mathbf{x}$ whose left-most 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 position in the rule head have the same variable. It can be proved that $q_{\Pi,\mathbf{S}} = q_{\Phi_\Pi,\mathbf{S}}$ for all schemas $\mathbf{S}$. ❏

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. Furthermore, Proposition 2 allow us to transfer results from MMSNP to OBDA. We start by considering the data complexity of the query evaluation problem: for a query q, the *evaluation problem* is to decide, given an instance $\mathfrak{D}$ and a 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.

The proof of the following theorem relies on Proposition 2 and Theorems 1, 3, and 6. It also exploits the fact that the Feder-Vardi dichotomy conjecture can equivalently be stated for MMSNP sentences [24, 33]. Some technical development is needed to deal with the presence of free variables. Details are in the full version of this paper.

Theorem 8 *($\mathcal{ALC}$,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) and ($\mathcal{S}$,UCQ) are two extensions of ($\mathcal{ALC}$,UCQ) that were identified in Section 3 to be more expressive than ($\mathcal{ALC}$,UCQ) itself. It was already proved in [36] (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 9 ([36]) *For every* NP*-Turing machine M, there is a query q in ($\mathcal{ALCF}$,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*).*

We leave it as an open problem to analyze the computational power of ($\mathcal{S}$,UCQ).

There are other interesting results that can be transferred from MMSNP to OBDA. Here, we consider query containment. Specifically, the following general containment problem was proposed in [10] 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 $\mathsf{cert}_{q_1,\mathcal{O}_1}(\mathfrak{D}) \subseteq \mathsf{cert}_{q_2,\mathcal{O}_2}(\mathfrak{D})$.[3] 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 ($\mathcal{ALC}$,UCQ), the above 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 [19].

It was proved in [24] that containment of MMSNP sentences is decidable. We thus obtain the following result for OBDA languages.

Theorem 10 *Query containment is decidable for the OBDA languages ($\mathcal{ALC}$,UCQ), ($\mathcal{ALCHIU}$,UCQ), and (UNFO,UCQ).*

Note that this result is considerably stronger than those in [10], which 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 10.

We now consider OBDA languages based on the guarded fragment and GNFO. By Proposition 1, (GF,UCQ) and (GNFO,UCQ) are strictly more expressive than MDDlog and we cannot use Proposition 2 to relate these query languages to the Feder-Vardi conjecture. Theorem 7 suggests that 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 (abbreviated GMSNP)* has the form $\exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ with $X_1, \dots, X_n$

[3]In fact, this definition is slightly different from the one used in [10]. There, containment is defined only over instances $\mathfrak{D}$ that are consistent w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$, i.e., where there is at least one finite $\mathbf{S}$-structure $(\mathsf{dom}, \mathfrak{D}')$ such that $\mathfrak{D} \subseteq \mathfrak{D}'$ and $\mathfrak{D}' \in \mathsf{Mod}(\mathcal{O}_i)$.

SO variables of any arity, $x_1, \dots, x_n$ FO-variables, and φ 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 α_i is of the form $X_i(\mathbf{x})$, $R(\mathbf{x})$ (with $R \in \mathbf{S}$), or $x = y$, and each β_i is of the form $X_i(\mathbf{x})$. Additionally, we require that for every head atom β_i, there is a body atom α_j such that α_j contains all variables from β_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 1 and Theorem 7, this yields the second statement of the following lemma; the first statement can be proved similarly to Proposition 2.

Theorem 11 *coGMSNP has the same expressive power as frontier-guarded DDlog and is strictly more expressive than coMMSNP.*

Although defined in a different way, GMSNP is essentially the same logic as MMSNP$_2$, which is studied in [37]. Specifically, 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 $\mathfrak{D} \models_\pi X(R(x_1, \dots, x_n))$ if $R(\pi(x_1), \dots, \pi(x_n)) \in \pi(X)$. We observe the following.

Proposition 3 *GMSNP and MMSNP$_2$ have the same expressive power.*

Details for the proofs of both Theorem 11 and Lemma 3 are in the full version of this paper. In [37], it was left as an open question whether MMSNP$_2$ is more expressive than MMSNP, which is resolved by the results above.

We leave it as an interesting open question whether Theorem 8 can be extended to (GF,UCQ) and (GNFO,UCQ), that is, whether GMSNP (equivalently: MMSNP$_2$) has a dichotomy between PTIME and NP if the Feder-Vardi conjecture holds. While this question is implicit already in [37], the results established in this paper underline its significance from a different perspective.

5. OBDA AND CSP

We show that OBDA languages based on AQs capture CSPs (and generalizations thereof), and we transfer results from CSPs to OBDA languages. In comparison to the previous section, we obtain a richer set of results, and often even worst-case optimal decision procedures. Recall that each finite relational structure $\mathfrak{B}$ over a schema $\mathbf{S}$ gives rise to a *constraint satisfaction problem* which is to decide, given a finite relational structure $\mathfrak{A}$ over $\mathbf{S}$, whether there is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ (written $\mathfrak{A} \rightarrow \mathfrak{B}$). In this context, the relational structure $\mathfrak{B}$ 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 ($\mathcal{ALC}$,BAQ), where BAQ is the class of *Boolean* atomic queries. To also cover non-Boolean AQs, we consider two natural generalizations of CSPs. First, a *generalized CSP* is defined by a finite *set $\mathcal{F}$ of templates*, rather than only a single one [25]. The problem then consists in deciding, given an input structure $\mathfrak{A}$, whether there is a template $\mathfrak{B} \in \mathcal{F}$ such that $\mathfrak{A} \rightarrow \mathfrak{B}$. Second, in a *(generalized) CSP with constant symbols*, both the template(s) and the input structure are endowed with constant symbols [23, 1]. To be more precise, let $\mathbf{S}$ be a schema and $\mathbf{c} = c_1, \dots, c_m$ a finite sequence of distinct constant symbols. A *finite relational structure over* $\mathbf{S} \cup \mathbf{c}$ has the form $(\mathfrak{A}, d_1, \dots, d_m)$ with $\mathfrak{A}$ a finite relational structure over $\mathfrak{A}$ that, in addition, interprets the constant symbols c_i by elements d_i of the domain dom of $\mathfrak{A}$, for $1 \leq i \leq m$. Let $(\mathfrak{A}, \mathbf{a})$ and $(\mathfrak{B}, \mathbf{b})$ be finite relational structures over $\mathbf{S} \cup \mathbf{c}$. A mapping h is a *homomorphism* from $(\mathfrak{A}, \mathbf{a})$ to $(\mathfrak{B}, \mathbf{b})$, written $(\mathfrak{A}, \mathbf{a}) \rightarrow (\mathfrak{B}, \mathbf{b})$, if it is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ and $h(a_i) = b_i$ for $1 \leq i \leq m$. A (generalized) CSP with constant symbols 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 constant symbols constitute the most general case. Specifically, each finite set of templates $\mathcal{F}$ over $\mathbf{S} \cup \mathbf{c}$ with $\mathbf{c} = c_1, \dots, c_m$ gives rise to an m-ary query coCSP($\mathcal{F}$) that maps every $\mathbf{S}$-instance $\mathfrak{D}$ to

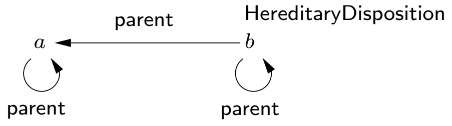
$$\{\mathbf{d} \in \mathsf{adom}(\mathfrak{D})^m \mid \forall (\mathfrak{B}, \mathbf{b}) \in \mathcal{F} : (\mathfrak{D}, \mathbf{d}) \not\rightarrow (\mathfrak{B}, \mathbf{b})\},$$

where we view $(\mathfrak{D}, \mathbf{d})$ as a finite relational structure whose domain is $\mathsf{adom}(\mathfrak{D})$. The query language that consists of all such queries is called *generalized coCSP with constant symbols*. The fragment of this query language that is obtained by admitting only sets of templates $\mathcal{F}$ without constant symbols is called *generalized coCSP*, and the fragment induced by singleton sets $\mathcal{F}$ without constant symbols is called *coCSP*.

Example 3 *Selecting an illustrative fragment of Examples 1 and 2, let*

$$\mathcal{O} = \{\exists \mathsf{parent}.\mathsf{HereditaryDisposition} \sqsubseteq \mathsf{HereditaryDisposition}\}$$
$$\mathbf{S} = \{\mathsf{HereditaryDisposition}, \mathsf{parent}\}$$

Moreover, let $q_2(x) = \mathsf{HereditaryDisposition}(x)$ be the query from Example 2. To identify a query in coCSP with constant symbols that is equivalent to the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q_2)$, let $\mathcal{B}$ be the following template:



It can be shown 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 $(\mathfrak{D}, d) \not\rightarrow (\mathfrak{B}, a)$ and thus the query coCSP$(\mathfrak{B})$ is as required.

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

Theorem 12 *The following are lists of query languages that have the same expressive power:*

1. *($\mathcal{ALCU}$,AQ), ($\mathcal{SHIU}$,AQ), unary simple MDDlog, and generalized coCSP with one constant symbol;*
2. *($\mathcal{ALC}$,AQ), ($\mathcal{SHI}$,AQ), unary connected simple MDDlog, and generalized coCSPs with one constant symbol such that all templates are identical except for the interpretation of the constant symbol;*
3. *($\mathcal{ALCU}$,BAQ), ($\mathcal{SHIU}$,BAQ), Boolean simple MDDlog, and generalized coCSP;*

4. *($\mathcal{ALC}$,BAQ), ($\mathcal{SHI}$,BAQ), Boolean connected simple MDDlog, and coCSP.*

Moreover, given the ontology-mediated query or monadic datalog program, the correponding CSP template is of at most exponential size and can be constructed in time polynomial in the size of the template.

Proof. The equivalences between OBDA languages and fragments of MDDlog have been proved in Section 3. We give a proof of the remaining claim of Point 1, namely that ($\mathcal{ALCU}$,AQ) and generalized coCSP with one constant symbol are equally expressive. We extend the notation used in the proof of Theorem 1. For simplicity, throughout this proof we regard $\forall R.C$ as an abbreviation for $\neg\exists R.\neg C$.

Let $Q = (\mathbf{S}, \mathcal{O}, A(x))$ be an ontology-mediated query formulated in ($\mathcal{ALCU}$,AQ). A *type for* $\mathcal{O}$ is a set $\tau \subseteq \mathsf{sub}(\mathcal{O})$ and $\mathsf{tp}(\mathcal{O})$ denotes the set of all types for $\mathcal{O}$. We say that $\tau \in \mathsf{tp}(\mathcal{O})$ is *realizable* if there is an $\mathfrak{A} = (\mathsf{dom}, \mathfrak{D}) \in \mathsf{Mod}(\mathcal{O})$ and a $d \in \mathsf{dom}$ such that $C \in \tau$ iff $\mathfrak{A} \models C^*[d]$ for all $C \in \mathsf{sub}(\mathcal{O})$. A set of types $T \subseteq \mathsf{tp}(\mathcal{O})$ is *realizable in a Q-countermodel* if there is an $\mathfrak{A} \in \mathsf{Mod}(\mathcal{O})$ that realizes exactly the types in T and such that $A \notin \tau$ for at least one $\tau \in T$.

Let $\mathcal{C}$ be the set of all $T \subseteq \mathsf{tp}(\mathcal{O})$ that are realizable in a Q-countermodel and maximal with this property. Note that the number of elements of $\mathcal{C}$ is bounded by the size of $\mathcal{O}$ since for any two distinct $T_1, T_2 \in \mathcal{C}$, there must be a concept $\exists U.D \in \mathsf{sub}(\mathcal{O})$ such that $\exists U.D \in \tau$ for all $\tau \in T_1$ and $\exists U.D \notin \tau$ for all $\tau \in T_2$ or vice versa; otherwise, we can take the disjoint union of any structures $\mathfrak{A}_1, \mathfrak{A}_2$ which show that T_1, T_2 are realizable in a Q-countermodel to obtain Q-countermodel that realizes $T_1 \cup T_2$. For $R \in \mathbf{S}$, we call a pair (τ_1, τ_2) of types *R-coherent* if $\exists R.C \in \tau_1$ for every $\exists R.C \in \mathsf{sub}(\mathcal{O})$ such that $C \in \tau_2$.

With each $T \in \mathcal{C}$, we associate the *canonical* $\mathbf{S}$*-structure* $\mathfrak{B}_T$ with domain T and the following facts:

- $B(\tau)$ for all $\tau \in T$ and $B \in \mathbf{S}$ such that $B \in \tau$;
- $R(\tau_1, \tau_2)$ for all $\tau_1, \tau_2 \in T$ and $R \in \mathbf{S}$ such that (τ_1, τ_2) is R-coherent.

Note that the construction of $\mathfrak{B}_T$ is well-known from the literature on modal and description logic. For example, $\mathfrak{B}_T$ can be viewed as a finite fragment of a canonical model of a modal logic that is constructed from maximal consistent sets of formulas [11]. Alternatively, $\mathfrak{B}_T$ can be viewed as the result of a type elimination procedure [41].

We obtain the desired set $\mathcal{F}$ of CSP templates by setting

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

One can show that for every $\mathbf{S}$-instance $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$, there exists $(\mathfrak{B}_T, \tau) \in \mathcal{F}$ with $(\mathfrak{D}, d) \to (\mathfrak{B}_T, \tau)$ iff $d \notin q_{\mathbf{S},\mathcal{O},A(x)}(\mathfrak{D})$. Thus, the ontology-mediated query Q is equivalent to the query defined by $\mathcal{F}$.

Conversely, assume that $\mathcal{F}$ is a finite set of $\mathbf{S}$-structures with one constant. Take some $(\mathfrak{B}, b) \in \mathcal{F}$, and for every d in the domain $\mathsf{dom}(\mathfrak{B})$ of $\mathfrak{B}$, create some fresh concept name A_d. Let A be another fresh concept name, and set

$$\begin{aligned}
\mathcal{O}_{\mathfrak{B},b} \quad = \quad & \{A_d \sqsubseteq \neg A_{d'} \mid d \neq d'\} \cup \\
& \{A_d \sqcap \exists R.A_{d'} \sqsubseteq \bot \mid R(d, d') \notin \mathfrak{B}, R \in \mathbf{S}\} \cup \\
& \{A_d \sqcap B \sqsubseteq \bot \mid B(d) \notin \mathfrak{B}, B \in \mathbf{S}\} \cup \\
& \{\top \sqsubseteq \bigsqcup_{d \in \mathsf{dom}(\mathfrak{B})} A_d, \ \neg A_b \sqsubseteq A\}
\end{aligned}$$

Consider the ontology-mediated query $Q_{\mathfrak{B},b} = (\mathbf{S}, \mathcal{O}_{\mathfrak{B},b}, A(x))$. One can show that for every $\mathbf{S}$-instance $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$, $(\mathfrak{D}, d) \to (\mathfrak{B}, b)$ iff $d \notin q_{Q_{\mathfrak{B},b}}(\mathfrak{D})$. Thus, $Q_{\mathfrak{B},b}$ is the desired query if $\mathcal{F}$ is a singleton. For the general case, let $\mathcal{O}$ be the disjunction over all $\mathcal{O}_{\mathfrak{B},b}$ with $(\mathfrak{B}, b) \in \mathcal{F}$. Note that $\mathcal{O}$ can be expressed in $\mathcal{ALCU}$: first, rewrite each $\mathcal{O}_{\mathfrak{B},b}$ into a single inclusion of the form $\top \sqsubseteq C_{\mathfrak{B},b}$ and then set

$$\mathcal{O} = \{\top \sqsubseteq \bigsqcup_{(\mathfrak{B},b) \in \mathcal{F}} \forall U.C_{\mathfrak{B},b}\}.$$

Using the above observation about the queries $Q_{\mathfrak{B},b}$, it is not hard to show that the ($\mathcal{ALCU}$,AQ)-query $Q = (\mathbf{S}, \mathcal{O}, A(x))$ is equivalent to the query coCSP($\mathcal{F}$).

This completes the proof of Point 1. The proofs of Points 2 to 4 are similar and given in the full version of this paper. ❑

Theorem 12 allows us to transfer results from the CSP world to OBDA, which, in light of recent progress on CSPs, turns out to be very fruitful. We start with data complexity.

Theorem 13 *($\mathcal{ALC}$,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).*

Since $\mathcal{SHIU}$-ontologies can be replaced by $\mathcal{ALCU}$-ontologies in ontology-mediated queries due to Theorem 5, the "if" direction of (all cases mentioned in) Theorem 13 actually follows from Theorem 8. The "only if" direction is a consequence of Theorem 12. We now consider further interesting applications of Theorem 12, in particular to deciding query containment, FO-rewritability, and datalog rewritability.

5.1 Query Containment

In Section 4, we have 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. Consequently, it is straightforward to show that query containment for generalized coCSP with constant symbols is NP-complete. Thus, Theorem 12 yields the following NEXPTIME upper bound for query containment in OBDA languages. The corresponding lower bound is proved in the full version of this paper by a non-trivial reduction of a NEXPTIME-complete tiling problem.

Theorem 14 *Query containment in ($\mathcal{SHIU}$,AQ$\cup$BQ) is in* NEXPTIME. *It is* NEXPTIME*-hard already for ($\mathcal{ALC}$,AQ) and for ($\mathcal{ALC}$,BAQ).*

It is a consequence of a result in [10] that query containment is undecidable for $\mathcal{ALCF}$. We show in the full version of this paper how the slight gap pointed out in Footnote 3 can be bridged.

5.2 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 [18, 22, 20]. Specifically, an ontology-mediated query $(\mathbf{S}, \mathcal{O}, q)$ is *FO-rewritable* if there exists an FO-query over $\mathbf{S}$ that is equivalent to it and *datalog-rewritable* if there exists a datalog program over $\mathbf{S}$ that defines it. We observe that every ontology-mediated query that is FO-rewritable is also datalog-rewritable.

Proposition 4 *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 [43] that if q_Q is FO-rewritable, then q_Q is rewritable into a UCQ (thus into datalog).*

Example 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. 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. Specifically, an $\mathbf{S}$-query coCSP($\mathcal{F}$) is *FO-definable* if there is an FO-sentence φ over $\mathbf{S}$ such that for all finite relational structures $\mathfrak{A}$ over $\mathbf{S}$, we have $\mathfrak{A} \models \varphi$ iff $\mathfrak{A} \not\to \mathfrak{B}$ for all $\mathfrak{B}$ in $\mathcal{F}$. Similarly, coCSP($\mathcal{F}$) is *datalog-definable* if there exists a datalog program Π that defines it. FO-definability and datalog-definability have been studied extensively for CSPs, culminating in the following results.

Theorem 15 *Deciding, for a given finite relational structure $\mathfrak{B}$ without constant symbols, whether coCSP($\mathfrak{B}$) is FO-definable is* NP*-complete [35]. The same is true for datalog-definability [26].*[4]

Combining the preceding theorem with Theorem 12, we obtain NEXPTIME upper bounds for deciding FO-rewritability and datalog-rewritability of queries from ($\mathcal{SHI}$,BAQ).

To capture the more important AQs rather than only BAQs, we show that Theorem 15 can be lifted, in a natural way, to generalized CSPs with constant symbols. The central step is provided by Proposition 5 below. For each finite relational structure $\mathfrak{B}$ with constant symbols $c_1, \ldots, c_n$, let us denote by $\mathfrak{B}^c$ the corresponding relational structure without constant symbols over the schema that contains additional unary relations $P_1, \ldots, P_n$, where each P_i denotes the singleton set that consists of the element denoted by c_i.

Proposition 5 *For every set of homomorphically incomparable structures $\mathfrak{B}_1, \ldots, \mathfrak{B}_n$ with constant symbols,*

1. *coCSP($\mathfrak{B}_1, \ldots, \mathfrak{B}_n$) is FO-definable iff coCSP($\mathfrak{B}_i^c$) is FO-definable for $1 \leq i \leq n$.*
2. *coCSP($\mathfrak{B}_1, \ldots, \mathfrak{B}_n$) is datalog-definable iff coCSP($\mathfrak{B}_i^c$) is datalog-definable for $1 \leq i \leq n$.*

A proof of Proposition 5 is provided in the full version of this paper. It relies on the characterization of FO-definable CSPs as those CSPs that have *finite obstruction sets*; this characterization was given in [2] for structures without constant symbols and follows from results in [43] for the case of structures with constant symbols.

Note that every set of structures $\mathfrak{B}_1, \ldots, \mathfrak{B}_n$ has a subset $\mathfrak{B}'_1, \ldots, \mathfrak{B}'_m$ which consists of homomorphically incomparable structures such that coCSP($\mathfrak{B}_1, \ldots, \mathfrak{B}_n$) is equivalent to coCSP($\mathfrak{B}'_1, \ldots, \mathfrak{B}'_m$). We use this observation to establish the announced lifting of Theorem 15.

Theorem 16 *FO-definability and datalog-definability of generalized CSP with constant symbols is* NP*-complete.*

Proof. To decide whether a generalized CSP with constant symbols given as a set of templates $\mathcal{F} = \{\mathfrak{B}_1, \ldots, \mathfrak{B}_n\}$ is FO-definable, it suffices to first guess a subset $\mathcal{F}' \subseteq \mathcal{F}$ and then to verify that (i) coCSP($\mathfrak{B}^c$) is FO-definable for each $\mathfrak{B} \in \mathcal{F}'$, and (ii) for each $\mathfrak{B} \in \mathcal{F}$ there is a $\mathfrak{B}' \in \mathcal{F}'$ such that $\mathfrak{B} \to \mathfrak{B}'$. By Theorem 15, this can be done in NP. Correctness follows from Proposition 5 and the fact that whenever there is a subset $\mathcal{F}'$ satisfying (i) and (ii), then by the observation above there must be a subset $\mathcal{F}'' \subseteq \mathcal{F}'$ of homomorphically incomparable structures such that coCSP($\mathcal{F}''$) is equivalent to coCSP($\mathcal{F}'$), which by (ii) is equivalent to coCSP($\mathcal{F}$). Datalog-definability can be decided analogously. ❏

From Theorems 12 and 16, 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 proved in the full version of this paper using a reduction from a NEXPTIME-hard tiling problem (in fact, the same problem as in the lower bound for query containment).

Theorem 17 *It is in* NEXPTIME *to decide FO-rewritability and datalog-rewritability of queries in ($\mathcal{SHIU}$,AQ$\cup$BAQ). Both problems are* NEXPTIME*-hard for ($\mathcal{ALC}$,AQ) and ($\mathcal{ALC}$, BAQ).*

Modulo a minor difference in the treatment of instances that are not consistent (see Footnote 3), it follows from a result in [36] that FO-rewritability is undecidable for ($\mathcal{ALCF}$,AQ). In the full version of this paper, we show how to bridge the difference and how to modify the proof so that the result also applies to datalog-rewritability.

Theorem 18 *FO-rewritability and datalog-rewritability are undecidable for ($\mathcal{ALCF}$,AQ) and ($\mathcal{ALCF}$,BAQ).*

6. CONCLUSION

Another query language frequently used in OBDA with description logics is conjunctive queries. The results in this paper imply that there is a dichotomy between PTIME and CONP for ($\mathcal{ALC}$,CQ) if and only if the Feder-Vardi conjecture holds. We leave it open whether there is a natural characterization of ($\mathcal{ALC}$,CQ) in terms of disjunctive datalog.

We mention two natural lines of future research. First, it would be interesting to understand the data complexity and query containment problem for (GF,UCQ) and (GNFO,UCQ). In particular, we would like to know whether Theorems 8 and 10 extend to (GF,UCQ) and (GNFO,UCQ). As explained in Section 4, resolving this question for Theorem 8 is equivalent to clarifying the computational status of GMSNP and MMSNP$_2$.

Another interesting topic for future work is to analyze FO-rewritability and datalog-rewritability of ontology-mediated queries based on UCQs (instead of AQs) as a decision problem. It follows from our results that this is equivalent to deciding FO-definability and datalog-definability of MMSNP formulas (or even GMSNP formulas).

Acknowledgements. We thank Benoit Larose and Liber Barto for discussions on datalog-definability of CSPs, and Florent Madeleine and Manuel Bodirsky for discussions on MMSNP.

Meghyn Bienvenu was supported by the ANR project PAGODA (ANR-12-JS02-007-01). Balder ten Cate was supported by NSF Grants IIS-0905276 and IIS-1217869. Carsten Lutz was supported by the DFG SFB/TR 8 “Spatial Cognition”.

[4]An NP algorithm for datalog-definability is implicit in [26], based on results from [9], see also [13]. We thank Benoit Larose and Liber Barto for pointing this out.

7. REFERENCES

[1] B. Alexe, B. ten Cate, P. G. Kolaitis, and W. C. Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4), 2011.

[2] A. Atserias. On digraph coloring problems and treewidth duality. In *LICS*, 2005.

[3] F. Baader, M. Bienvenu, C. Lutz, and F. Wolter. Query and predicate emptiness in description logics. In *KR*, 2010.

[4] F. Baader, D. Calvanese, D. L. McGuiness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[5] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI*, 2011.

[6] V. Bárány, G. Gottlob, and M. Otto. Querying the guarded fragment. In *LICS*, 2010.

[7] V. Bárány, B. ten Cate, and M. Otto. Queries with guarded negation. *PVLDB*, 5(11), 2012.

[8] V. Bárány, B. ten Cate, and L. Segoufin. Guarded negation. In *ICALP*, 2011.

[9] L. Barto and M. Kozik. Constraint satisfaction problems of bounded width. In *FOCS*, 2009.

[10] M. Bienvenu, C. Lutz, and F. Wolter. Query containment in description logics reconsidered. In *KR*, 2012.

[11] P. Blackburn, M. de Rijke, and Y. Venema. Modal Logic. *Cambridge University Press*, 2001.

[12] M. Bodirsky, H. Chen, and T. Feder. On the complexity of MMSNP. *SIAM J. Discrete Math.*, 26(1):404–414, 2012.

[13] A. Bulatov. Bounded relational width. In preparation. http://www.cs.sfu.ca/ abulatov/mpapers.html.

[14] A. A. Bulatov. On the CSP dichotomy conjecture. In *CSR*, 2011.

[15] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *PODS*, 2009.

[16] A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193, 2012.

[17] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *KR*, 2006.

[18] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3), 2007.

[19] D. Calvanese, G. D. Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *PODS*, 1998.

[20] B. Cuenca Grau, M. Kaminski, and B. Motik Computing Datalog Rewritings Beyond Horn Ontologies. In *IJCAI*, 2013

[21] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Trans. Database Syst.*, 22(3), 1997.

[22] T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao. Towards practical query answering for Horn-$\mathcal{SHIQ}$. In *DL*, 2012.

[23] T. Feder, F. R. Madelaine, and I. A. Stewart. Dichotomies for classes of homomorphism problems involving unary functions. *Theor. Comput. Sci.*, 314(1-2), 2004.

[24] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1), 1998.

[25] J. Foniok, J. Nesetril, and C. Tardif. Generalised dualities and maximal finite antichains in the homomorphism order of relational structures. *Eur. J. Comb.*, 29(4), 2008.

[26] R. Freese, M. Kozik, A. Krokhin, M. Maróti, R. KcKenzie, and R. Willard. On Maltsev conditions associated with omitting certain types of local structures. In preparation. http://www.math.hawaii.edu/∼ralph/Classes/619/OmittingTypesMaltsev.pdf

[27] G. Gottlob, E. Grädel, and H. Veith. Datalog LITE: a deductive query language with linear time model checking. *ACM Trans. Comput. Log.*, 3(1), 2002.

[28] G. Gottlob and T. Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In *KR*, 2012.

[29] U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3), 2007.

[30] S. Kikot, R. Kontchakov, V. V. Podolskii, and M. Zakharyaschev. Exponential lower bounds and separation for query rewriting. In *ICALP*, 2012.

[31] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in DL-Lite. In *KR*, 2010.

[32] A. Krisnadhi and C. Lutz. Data complexity in the $\mathcal{EL}$ family of DLs. In *LPAR*, 2007.

[33] G. Kun. Constraints, MMSNP, and Expander Structures. http://arxiv.org/abs/0706.1701v1, 2007.

[34] G. Kun and J. Nesetril. Forbidden lifts (NP and CSP for combinatorialists). *Eur. J. Comb.*, 29(4), 2008.

[35] B. Larose, C. Loten, and C. Tardif. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Comp. Sci.*, 3(4), 2007.

[36] C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In *KR*, 2012.

[37] F. R. Madelaine. Universal structures and the logic of forbidden patterns. *Logical Methods in Comp. Sci.*, 5(2), 2009.

[38] F. R. Madelaine and I. A. Stewart. Constraint satisfaction, logic and forbidden patterns. *SIAM J. Comput.*, 37(1), 2007.

[39] B. Motik. *Reasoning in description logics using resolution and deductive databases*. PhD thesis, 2006.

[40] A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10, 2008.

[41] V. R. Pratt. Models of program logics. In *FoCS*, 1979.

[42] R. Rosati and A. Almatelli. Improving Query Answering over DL-Lite Ontologies. In *KR*, 2010.

[43] B. Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3), 2008.

[44] S. Rudolph, M. Krötzsch, and P. Hitzler. Type-elimination-based reasoning for the description logic $\mathcal{SHIQ}b_s$ using decision diagrams and disjunctive datalog. *Logical Methods in Comp. Sci.*, 8(1), 2012.

[45] F. Simancik. Elimination of complex RIAs without automata. In *DL*, 2012.

[46] B. ten Cate and L. Segoufin. Unary negation. In *STACS*, 2011.

[47] W3C OWL Working Group. OWL 2 Web Ontology Language. http://www.w3.org/TR/owl2-overview/, 2012.