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

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

Ontology-based data access is concerned with querying incomplete data sources in the presence of domain-specific knowledge provided by an ontology. A central notion in this setting is that of an *ontology-mediated query*, which is a database query coupled with an ontology. In this 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 show that popular ontology-mediated query languages have the same expressive power as natural fragments of disjunctive datalog, and we study the relative succinctness of ontology-mediated queries and disjunctive datalog queries. Second, we establish intimate connections between ontology-mediated queries and constraint satisfaction problems (CSPs) and their logical generalization, MMSNP formulas. Third, we exploit these connections to obtain new results regarding (i) first-order rewritability and datalog-rewritability of ontology-mediated queries, (ii) P/NP dichotomies for ontology-mediated queries, and (iii) the query containment problem for ontology-mediated queries.

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

General Terms: Algorithms, Languages, Theory

Additional Key Words and Phrases: Ontology-Based Data Access; Query Answering; Query Rewriting

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 [Calvanese et al. 1998; Calvanese et al. 2007; Calì et al. 2012]. We refer to this as *ontology-based data access (OBDA)*.

There are several important use cases for OBDA. A classical one is to enrich an incomplete data source with background knowledge, in order to obtain a more complete set of answers to a query. For example, if a medical patient database contains the facts

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”.

Author’s addresses: M. Bienvenu, Laboratoire de Recherche en Informatique, Université Paris-Sud; B. ten Cate, University of California, Santa Cruz; Carsten Lutz, Fachbereich Informatik, Universität Bremen; F. Wolter, Department of Computer Science, University of Liverpool.

DOI:http://dx.doi.org/10.1145/0000000.0000000

that patient1 has finding Erythema Migrans and patient2 has finding Lyme disease, and the ontology provides the background knowledge that a finding of Erythema Migrans is sufficient for diagnosing Lyme disease, then both patient1 and patient2 can be returned when querying for patients that have the diagnosis Lyme disease. This use of ontologies is central to query answering in the semantic web. OBDA can also be used to enrich the data schema (that is, the relation symbols allowed in the presentation of the data) with additional symbols to be used in a query. For example, a patient database may contain facts such as patient1 has diagnosis Lyme disease and patient2 has diagnosis Listeriosis, and an ontology could add the knowledge that Lyme disease and Listeriosis are both bacterial infections, thus enabling queries such as "return all patients with a bacterial infection" despite the fact that the data schema does not include a relation or attribute explicitly referring to bacterial infections. Especially in the 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 [Poggi et al. 2008]. This typically involves mappings from the source schemas to the schema of the ontology, which we will not explicitly consider here.

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

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, (ii) to make progress towards complete and decidable classifications of ontology-mediated queries with respect to their data complexity, and (iii) to establish decidability and tight complexity bounds for relevant reasoning problems such as query containment and deciding whether a given ontology-mediated query is FO-rewritable or datalog-rewritable.

We take an ontology-mediated query to be a triple $(\mathbf{S}, \mathcal{O}, q)$ where $\mathbf{S}$ is a *data schema*, $\mathcal{O}$ an ontology, and q a query. Here, the data schema $\mathbf{S}$ fixes the set of relation symbols allowed in the data and the ontology $\mathcal{O}$ is a logical theory that may use the relation symbols from $\mathbf{S}$ as well as additional symbols. The query q can use any relation symbol that occurs in $\mathbf{S}$ or in $\mathcal{O}$. As ontology languages, we consider a range of standard description logics (DLs) and several fragments of first-order logic that embed ontology languages such as Datalog$^{\pm}$ [Calì et al. 2009], namely the guarded fragment (GFO), the unary negation fragment (UNFO), and the guarded negation fragment (GNFO).

As query languages for q, we focus on unions of conjunctive queries (UCQs) and unary atomic queries (AQs). The latter are of the form $A(x)$, with A a unary relation symbol, and correspond to what are traditionally called *instance queries* in the OBDA literature (note that A may be a relation symbol from $\mathcal{O}$ that is not part of the data schema). These two query languages are among the most used 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, (GFO, UCQ) refers to ontology-mediated queries in which $\mathcal{O}$ is a GFO-ontology and q is a UCQ. We refer to such query languages $(\mathcal{L}, \mathcal{Q})$ as *ontology-mediated query languages (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 $(\mathcal{ALC}, \text{UCQ})$ has the same expressive power as monadic disjunctive datalog (abbreviated MDDlog) and that $(\mathcal{ALC}, \text{AQ})$ has the same expressive power as unary queries defined in a syntactic fragment of MDDlog that we call connected simple MDDlog. Similar results hold for various description logics that extend $\mathcal{ALC}$ with, for example, inverse roles, role hierarchies, and the universal role, all of which are standard operators included in the W3C-standardized ontology language OWL2 DL. Turning to other fragments of first-order logic, we then show that $(\text{UNFO}, \text{UCQ})$ also has the same expressive power as MDDlog, while (GFO, UCQ) and $(\text{GNFO}, \text{UCQ})$ 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.

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

In Section 4, we study ontology-mediated queries from a *descriptive complexity* perspective. In particular, we establish an intimate connection between OBDA query languages, constraint satisfaction problems, and MMSNP. Recall that constraint satisfaction problems (CSPs) form a subclass of the complexity class NP that, although it contains NP-hard problems, is in certain ways more computationally well-behaved. The widely known Feder-Vardi conjecture [Feder and Vardi 1998] states 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. The conjecture thus asserts that there are no CSPs that are NP-intermediate in the sense of Ladner's theorem. Monotone monadic strict

NP without inequality (abbreviated MMSNP) was introduced by Feder and Vardi as a logical generalization of CSP that enjoys similar computational properties [Feder and Vardi 1998]. In particular, it was shown in [Feder and Vardi 1998; Kun 2007] that there is a dichotomy between PTIME and NP for MMSNP sentences if and only if the Feder-Vardi conjecture holds.

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

To establish a counterpart of (GFO, UCQ) and $(\text{GNFO}, \text{UCQ})$ in the MMSNP world, we introduce guarded monotone strict NP (abbreviated GMSNP) as a generalization of MMSNP; specifically, GMSNP is obtained from MMSNP by allowing guarded second-order quantification in the place of monadic second-order quantification, similarly as in the transition from MDDlog to frontier-guarded disjunctive datalog. The resulting query language coGMSNP has the same expressive power as frontier-guarded disjunctive datalog, and therefore, in particular, (GFO, UCQ) and $(\text{GNFO}, \text{UCQ})$ capture coGMSNP. We observe that GMSNP has the same expressive power as the extension MMSNP_2 of MMSNP proposed in [Madelaine 2009]. 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 [Madelaine 2009].

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

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

We first consider the data complexity of query evaluation for OBDA languages. Ideally, one would like to classify the data complexity of every ontology-mediated query within a given OBDA language such as $(\mathcal{ALC}, \text{UCQ})$. Our aforementioned results tie this task to proving the Feder-Vardi conjecture. We obtain that there is a dichotomy between PTIME and CONP for ontology-mediated queries from $(\mathcal{ALC}, \text{UCQ})$ if and only if the Feder-Vardi conjecture holds, and similarly for many other OBDA languages based on UCQs. Additionally, we obtain the same result for ontology-mediated query languages based on atomic queries such as $(\mathcal{ALC}, \text{BAQ})$ and $(\mathcal{ALC}, \text{AQ})$. This even holds for ontology-mediated query languages based on very expressive descriptions logics such as $(\mathcal{SHIU}, \text{BAQ})$ and $(\mathcal{SHIU}, \text{AQ})$. We also consider the standard extension $\mathcal{ALCF}$ of $\mathcal{ALC}$ with functional roles and note that, for query evaluation in $(\mathcal{ALCF}, \text{AQ})$, there is no dichotomy between PTIME and CONP unless PTIME = NP.

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

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

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

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

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

2. PRELIMINARIES

Schemas, Instances, and Queries. A *schema* is a finite collection $\mathbf{S} = (S_1, \dots, S_k)$ of relation symbols with associated arity. A *fact* over $\mathbf{S}$ is an expression of the form $S(a_1, \dots, a_n)$ where $S \in \mathbf{S}$ is an n-ary relation symbol, and $a_1, \dots, a_n$ 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 in $\mathfrak{D}$. We will frequently use boldface notation for tuples, such as in $\mathbf{a} = (a_1, \dots, a_n)$, and we denote the empty tuple by $()$.

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

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

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

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

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

Ontology-Mediated Queries. We introduce the fundamentals of ontology-based data access. An *ontology language* $\mathcal{L}$ is a fragment of first-order logic (i.e., a set of FO sentences), and an *$\mathcal{L}$-ontology* $\mathcal{O}$ is a finite set of sentences from $\mathcal{L}$.[1] We introduce various concrete ontology languages throughout the 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 is a query over $\mathbf{S} \cup \mathsf{sig}(\mathcal{O})$, with $\mathsf{sig}(\mathcal{O})$ the set of relation symbols used in $\mathcal{O}$. Here, we call $\mathbf{S}$ the *data schema*. Note that the ontology can introduce symbols that are not in the data schema, which allows it to enrich the schema of the query q. Of course, we do not require that every relation symbol of the data schema actually occurs 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 a query over $\mathbf{S}$-instances, even though $\mathcal{O}$ and q might use additional relation symbols.

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

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

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

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

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

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

All ontology languages considered in this 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 [Baader et al. 2003; Bárány et al. 2010; Bárány et al. 2012].

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

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

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

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

$$\begin{array}{ll} \mathsf{HasFinding}(\mathsf{patient1}, \mathsf{jan12find1}) & \mathsf{ErythemaMigrans}(\mathsf{jan12find1}) \\ \mathsf{HasDiagnosis}(\mathsf{patient2}, \mathsf{may7diag2}) & \mathsf{Listeriosis}(\mathsf{may7diag2}) \end{array}$$

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

Description Logics for Specifying Ontologies. In description logic, schemas are generally restricted to relation symbols of arity one and two, called *concept names* and

Table II. First-order translation of $\mathcal{ALC}$ concepts

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

role names, respectively. Despite the potential presence of unary relations, we speak of *binary schemas*. We briefly review the basic description logic $\mathcal{ALC}$. Relevant extensions of $\mathcal{ALC}$ will be introduced later on in the 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 II. An $\mathcal{ALC}$-ontology $\mathcal{O}$ then translates into the set of FO-sentences

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

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

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

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

$$\begin{array}{lll} P(x) \leftarrow \mathsf{HereditaryPredisposition}(x) & & \mathsf{goal}(x) \leftarrow P(x) \\ P(x) \leftarrow \mathsf{HasParent}(x,y) \wedge P(y) & & \end{array}$$

but not to any first-order query.

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

3. OBDA AND DISJUNCTIVE DATALOG

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

A *disjunctive datalog rule* 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 relation* 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) relations*, and all remaining relation symbols in Π are *extensional (EDB) relations*. An $\mathbf{S}'$-instance, with $\mathbf{S}'$ the set of all (IDB and EDB) relation symbols in Π, is a *model* of Π if it satisfies all rules in Π. We use $\mathsf{Mod}(\Pi)$ to denote the set of all models of Π.

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 relations of Π: for every instance $\mathfrak{D}$ over $\mathbf{S}$, we have

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

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 [Eiter et al. 1997].

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

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

A *monadic disjunctive datalog (MDDlog) program* is a DDlog program in which all IDB relations 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.

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

THEOREM 3.1. *Query evaluation in MDDlog is Π^p_2-complete regarding combined complexity. The lower bound holds already over binary schemas.*

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

For the lower bound, we give a reduction from 2QBF validity. Consider a 2QBF $\forall x_1 \ldots x_m \exists y_1 \ldots y_n \, \varphi$, where φ is a 3CNF over clauses $c_1, \ldots, c_k$. We create an MDDlog program Π whose set of EDB relations consists of unary relations $C_1, \ldots, C_k$ and binary relations V_1, V_2, V_3, and start, and whose (monadic) IDB relations are $X_1, \ldots, X_m$. For

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

each clause c_i, we denote by v_i^j $(1 \leq j \leq 3)$ the variable appearing in the jth literal of c_i, and we let S_i denote the set of tuples in $\{0,1\}^3$ representing the seven truth assignments for (v_i^1, v_i^2, v_i^3) which satisfy c_i. We encode φ using the instance $\mathcal{D}_\varphi$ defined as follows:

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

The program Π consists of a set of rules which select a truth assignment for the universally quantified variables:

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

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

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

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

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

Let S be a schema and $\mathcal{C}$ a set of unary relation symbols (colors) $\{C_1, \dots, C_n\}$ that is disjoint from S. A *$\mathcal{C}$-colored* S*-instance* is an $\mathbf{S} \cup \mathcal{C}$-instance $\mathfrak{D}$ such that for every $d \in \mathsf{adom}(\mathfrak{D})$, there is exactly one fact in $\mathfrak{D}$ of the form $C_i(d)$. An instance $\mathfrak{D}'$ is called a *$\mathcal{C}$-coloring* of an S-instance $\mathfrak{D}$ if $\mathfrak{D}$ is the restriction of $\mathfrak{D}'$ to the schema S. Given a set $\mathcal{F}$ of $\mathcal{C}$-colored S-instances (called *forbidden patterns*), we define $\mathsf{Forb}(\mathcal{F})$ as the set of all S-instances $\mathfrak{D}$ for which there exists a $\mathcal{C}$-coloring $\mathfrak{D}'$ of $\mathfrak{D}$ with $\mathfrak{F} \not\to \mathfrak{D}'$ for every $\mathfrak{F} \in \mathcal{F}$. The forbidden patterns problem defined by $\mathcal{F}$ is to decide whether a given S-instance belongs to $\mathsf{Forb}(\mathcal{F})$. We let FPP denote the set of all forbidden patterns problems and use coFPP to refer to the query language that consists of all Boolean queries $q_{\mathcal{F},\mathbf{S}}$ defined by setting

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

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

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

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

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

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

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

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

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

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

3.1. Ontologies Specified in Description Logics

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

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

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

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

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

The definition of certain answers to Q on an instance $\mathfrak{D}$ involves a quantification over all models of $\mathcal{O}$ that extend $\mathfrak{D}$. It turns out that in the case of $(\mathcal{ALC}, \text{UCQ})$ queries, it suffices to consider a particular type of extensions of $\mathfrak{D}$ that we term *forest extension*. Intuitively, such an extension of $\mathfrak{D}$ corresponds to attaching tree-shaped structures to the elements of $\mathfrak{D}$. Formally, a relational structure $(\mathsf{dom}, \mathfrak{B})$ over a binary schema is *tree-shaped* if the directed graph $(\mathsf{dom}, \{(a,b) \mid R(a,b) \in \mathfrak{B} \text{ for some } R\})$ is a tree and there do not exist facts $R(a,b), S(a,b) \in \mathfrak{B}$ with $R \neq S$. A relational structure $\mathfrak{D}'$ is a forest extension of an instance $\mathfrak{D}$ if $\mathfrak{D} \subseteq \mathfrak{D}'$ and $\mathfrak{D}' \setminus \mathfrak{D}$ is a union of tree-shaped instances $\{\mathfrak{D}'_a \mid a \in \mathsf{adom}(\mathfrak{D})\}$ such that

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

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

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

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

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

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

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

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

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

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

It can be shown that the number of queries in $\mathsf{tree}(q)$ is bounded by $|q|$, see [Lutz 2008] for full details. Moreover, we clearly have $|p| \leq |q|$ for all $p \in \mathsf{tree}(q)$. Set $\mathsf{cl}(\mathcal{O}, q) = \mathsf{sub}(\mathcal{O}) \cup \mathsf{tree}(q)$. A *type* (for $\mathcal{O}$ and q) is a subset of $\mathsf{cl}(\mathcal{O}, q)$. The CQs in $\mathsf{tree}(q)$ include all potential 'non-core components' from the intuitive explanation above. The answer variable of such a CQ (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 are used as IDB relations, and the symbols from $\mathbf{S}$ are the EDB relations.

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

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

Set k equal to the maximum of 2 and the width of q, that is, the number of variables that occur in q. By a *diagram*, we mean a conjunction $\delta(x_1, \dots, x_n)$ 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:

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

Intuitively, these rules 'guess' a (representation of a) forest 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 P_τ determine those non-core component queries that homomorphically map into the $\mathfrak{D}'_a$ parts. The second line ensures coherence of the guesses and the last line guarantees that q has the required homomorphism into $\mathfrak{D}'$. It is proved in Appendix A.1 that Π is indeed equivalent to Q.

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

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

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

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

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

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

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

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

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

Each disjunctive datalog rule can be associated with an undirected graph whose nodes are the variables that occur in the rule and whose edges reflect co-occurrence of two variables in an atom in the rule body. We say 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. An MDDlog program is *unary* if its goal relation is unary.

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

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

(2) *for every MDDlog program Π, there is an equivalent query $(\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALC}, \text{AQ})$ such that $|\mathcal{O}| \in O(|\Pi|)$.*

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

$$\bigwedge_{C \in \mathsf{sub}(\mathcal{O})} (\, P_C(x) \vee P_{\neg C}(x) \,) \leftarrow \mathsf{adom}(x)$$

$$\bot \leftarrow \delta(\mathbf{x}) \qquad \text{for all non-realizable diagrams } \delta(\mathbf{x})$$

$$\mathsf{goal}(x) \leftarrow P_{A_0}(x)$$

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

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 $\exists R.\top \sqsubseteq \mathsf{goal}$. Similarly, $\mathsf{goal}(x) \leftarrow R(y, z)$ is rewritten into $\top \sqsubseteq \forall R.\mathsf{goal}\}$, and $P_1(x) \vee P_2(y) \leftarrow R(x, y) \wedge P_3(x) \wedge P_4(y)$ is rewritten into $P_3 \sqcap \exists R.(P_4 \sqcap \neg P_2) \sqcap \neg P_1 \sqsubseteq \bot$. Let $\mathcal{O}$ be the resulting ontology and let $q = \mathsf{goal}(x)$. Then Π is equivalent to the query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S}$ consists of the EDB relations in Π. $\square$

Note that the connectedness condition is required since one cannot express MDDlog rules such as $\mathsf{goal}(x) \leftarrow A(x) \wedge B(y)$ in $(\mathcal{ALC}, \text{AQ})$. Multiple variable occurrences in EDB relations have to be excluded because programs such as $\mathsf{goal}(x) \leftarrow A(x),\ \bot \leftarrow R(x,x)$ (return all elements in A if the instance contains no reflexive R-edge, and return the active domain otherwise) also cannot be expressed in $(\mathcal{ALC}, \text{AQ})$.

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

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

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

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

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

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

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

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

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

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

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

[4] S stands for 'start computation'.

in the ontology, which require the role name R to be interpreted as a transitive relation. $\mathcal{ALCF}$ is the extension of $\mathcal{ALC}$ with *functional role* statements $\mathsf{func}(R)$, which require the role name R to be interpreted as a partial function. Finally, $\mathcal{ALCU}$ is the extension with the *universal role* U which is interpreted as the total relation $\mathsf{dom} \times \mathsf{dom}$ 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.

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

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

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

PROOF. (sketch) To establish Point 1, we use a folklore technique for eliminating inverse roles [De Giacomo and Lenzerini 1994]. Let $Q = (\mathbf{S}, \mathcal{O}, q)$ be a query from $(\mathcal{ALCHIU}, \text{UCQ})$ and assume w.l.o.g. that

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

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

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

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

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

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

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

PROOF. W.l.o.g., we can restrict our attention to a restricted class of instances. Fix the schema $\mathbf{S} = \{R, Y_0, Y_1, Y_2\}$ with R a role name and Y_0, Y_1, Y_2 concept names. We use the composition $R^-; R$ of the inverse role R^- with its base role R to simulate a symmetric role, which is not available in the DLs considered here. The *counting instance of length k* is the S-instance $\mathfrak{C}_k$ that consists of an $R^-; R$-path of length k, that

$$\begin{tikzcd}
Y_0 & & Y_1 & & Y_2 & & Y_0 \\
a_0 & a_1 \arrow[l, "R"'] \arrow[r, "R"] & a_2 & a_3 \arrow[l, "R"'] \arrow[r, "R"] & a_4 & a_5 \arrow[l, "R"'] \arrow[r, "R"] & a_6
\end{tikzcd}$$

Fig. 1. Counting instance of length 3.

is, a sequence of elements $a_0, \ldots, a_{2k}$ such that, for $0 < i < 2k$ with i odd, we have $R(a_i, a_{i-1}), R(a_i, a_{i+1}) \in \mathfrak{D}$. We additionally require that, for $0 \leq i \leq 2k$ with i even, $\mathfrak{D}$ contains the fact $Y_j(a_i)$ where $j = i/2 \bmod 3$. An example can be found in Figure 1. In Appendix A.1, we show the following, which clearly establishes Theorem 3.7: there is a polynomial p such that for every $k > 0$,

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

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

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

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

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

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

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

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

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

PROOF. Assume for a contradiction that the S-instances $\mathfrak{D}_0$ and $\mathfrak{D}_1$ described in the lemma exist for all $k, n > 0$, but that Q is equivalent to some query in MDDlog. Then, by Proposition 3.2, there is a set $\mathcal{F}$ of $\mathcal{C}$-colored S-structures such that for all S-instances $\mathfrak{D}$, we have $Q(\mathfrak{D}) = 1$ if and only if $\mathfrak{D} \notin \mathsf{Forb}(\mathcal{F})$. Let $k_0 = |\mathcal{C}|$, and let n_0 be the maximal number of elements in the domain of any $\mathfrak{F} \in \mathcal{F}$. We can assume w.l.o.g. that $\mathcal{C} = \mathcal{C}_{k_0}$.

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

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

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

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

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

— $\mathfrak{D}_1$ has elements e, f and $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$ and the atoms $R(e, a_1), R(a_m, f)$ and $R(a_i, a_{i+1})$ for $1 \leq i < m$, and $S(e, b_1), S(b_m, f)$ and $S(b_i, b_{i+1})$ for $1 \leq i < m$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2. Ontologies Specified in First-Order Logic

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

The Unary Negation Fragment. The *unary negation fragment of first-order logic* (UNFO) [ten Cate and Segoufin 2011] is the fragment of first-order logic that consists

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

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. Note that the translation of $\mathcal{ALC}$-ontologies into FO formulas given in Table II actually produces UNFO formulas.

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

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

PROOF. (sketch) Point (2) is a consequence of Theorem 3.3 and the fact that $(\mathcal{ALC},\text{UCQ})$ is a fragment of (UNFO,UCQ). Here, we provide the translation from (UNFO,UCQ) to MDDlog. Let $Q = (\mathbf{S},\mathcal{O},q) \in (\text{UNFO},\text{UCQ})$ be given. Every UNFO-formula with at most one free variable is equivalent to a disjunction of formulas generated by the 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 $\psi_i(x,\mathbf{y})$ is either a relational atom or a formula with at most one free variable generated by the same grammar. Note that all generated formulas have at most one free variable and that no equality is used although we allow it in the original definition of UNFO. For the ontology $\mathcal{O}$, we assume w.l.o.g. that it is a *single* sentence generated by the above grammar, rather than a disjunction of such sentences, because $\mathsf{cert}_{q,\mathcal{O}_1\vee\mathcal{O}_2}(\mathfrak{D})$ is the intersection of $\mathsf{cert}_{q,\mathcal{O}_1}(\mathfrak{D})$ and $\mathsf{cert}_{q,\mathcal{O}_2}(\mathfrak{D})$, and MDDlog is closed under taking intersections of queries. Let k be the maximum of the number of variables in $\mathcal{O}$ and the number of variables in q. We denote by $\mathsf{cl}_k(\mathcal{O},q)$ the set of all formulas $\varphi(x)$ of the form

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

with $\mathbf{y} = (y_1,\ldots,y_m)$, $m \leq k$, $n \leq |\mathcal{O}| + |q|$, where each ψ_i is either a relational atom that uses a symbol from q or is of the form $\chi(x)$ or $\chi(y_i)$, for $\chi(z) \in \mathsf{sub}(\mathcal{O})$. Note that $\mathsf{cl}_k(\mathcal{O},q)$ contains all CQs that use only symbols from q and whose size is bounded by the number of atoms in q. Also the length of each formula in $\mathsf{cl}_k(\mathcal{O},q)$ is polynomial in $|\mathcal{O}| + |q|$, and consequently, the cardinality of $\mathsf{cl}_k(\mathcal{O},q)$ is single exponential in $|\mathcal{O}| + |q|$. A *type* τ is a subset of $\mathsf{cl}_k(\mathcal{O},q)$. 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 symbols. As in the proof of Theorem 3.3, we call a structure $\mathfrak{B}$ over $\mathbf{S}' \cup \mathsf{sig}(\mathcal{O})$ *type-coherent* if for all types 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 3.3. It follows from [ten Cate and Segoufin 2011] that it is decidable whether a diagram implies a query, and whether a diagram is realizable. The MDDlog program Π is defined as in the proof of Theorem 3.3, repeated here for the sake of completeness:

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

In Appendix A.2, we prove that the resulting MDDlog query q_Π is equivalent to Q. $\square$

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

The Guarded Fragment and the Guarded Negation Fragment. The *guarded fragment of first-order logic* (GFO) comprises all formulas built up from atomic formulas using the Boolean connectives and guarded quantification of the form $\exists \mathbf{x}(\alpha \wedge \varphi)$ and $\forall \mathbf{x}(\alpha \rightarrow \varphi)$, where, in both cases, 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 $\mathcal{ALC}$-ontology into an equivalent sentence of GFO.

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

PROPOSITION 3.15. *The Boolean query*

(†) *there are* $a_1, \dots, a_n, b$*, 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 $(\mathit{GFO}, \mathit{UCQ})$ *and not in MDDlog.*

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

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

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

— $\mathfrak{D}_1$ has elements $d_1, \dots, d_m, e$ and the atoms $A(d_1)$, $B(d_m)$, and $P(d_i, e, d_{i+1})$ for $1 \leq i < m$.

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

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

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

— If $e \in \mathsf{adom}(\mathfrak{B}_1')$, then $h(e) = e_i$.
— If $d_1 \in \mathsf{adom}(\mathfrak{B}_1')$, then $h(d_1) = d_1$.
— If $d_m \in \mathsf{adom}(\mathfrak{B}_1')$, then $h(d_m) = d_m$.
— If $d_{\ell+1}, \dots, d_{\ell+p}$ is a maximal sequence of elements from $\mathsf{adom}(\mathfrak{B}_1')$ with $\ell + p < m$, then let ℓ' be such that the coloring of $d_{\ell+1}, \dots, d_{\ell+p}$ coincides with the coloring of

$d_{\ell'+1}, \dots, d_{\ell'+p}$ and $i \notin \{\ell'+1, \ell'+2, \dots, \ell'+n-1\}$. Set $h(d_{\ell+j}) = d_{\ell'+j}$ for every $1 \leq j \leq p$.

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

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

As fragments of first-order logic, the unary negation fragment and the guarded fragment are incomparable in expressive power. They have a common generalization, which is known as the guarded-negation fragment (GNFO) [Bárány et al. 2011]. This fragment is defined in the same way as UNFO, except that, besides unary negation, we allow *guarded negation* of the form $\alpha \wedge \neg\varphi$, where the guard 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 GFO. Specifically, (GFO, UCQ) and $(\text{GNFO}, \text{UCQ})$ are expressively equivalent to a natural generalization of MDDlog, namely *frontier-guarded DDlog.* Recall that a datalog rule is *guarded* if its body includes an atom that contains all variables which occur in the rule [Gottlob et al. 2002]. A weaker notion of guardedness, which we call here *frontier-guardedness*, inspired by [Baget et al. 2011; Bárány et al. 2012], 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 because the head of a rule in MDDlog contains at most one variable which has to occur in the body of the rule. We now show that both (GFO, UCQ) and $(\text{GNFO}, \text{UCQ})$ have the same expressive power as frontier-guarded DDlog. For understandng the following theorem, it is useful to recall that every sentence of GFO can be translated into an equivalent sentence of GNFO with only a polynomial blowup [Bárány et al. 2011].

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

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

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

For item (3), we make use of a result from [Bárány et al. 2011], which states that, for every GNFO sentence ϕ over a schema $\mathbf{S}' \supseteq \mathbf{S}$, we can construct in polynomial time a GFO sentence ψ and a positive-existential first-order sentence χ, both of which are over a possibly larger schema $\mathbf{T} = \mathbf{S}' \cup \{T_1, \dots, T_n\}$, such that ϕ is logically equivalent to the existential second-order sentence $\exists T_1 \dots T_n(\psi \wedge \neg\chi)$. By applying this translation to $\mathcal{O}$, and using the fact that a positive-existential FO sentence can be translated to a UCQ in exponential time, we obtain a GFO sentence $\mathcal{O}'$ and a Boolean UCQ q'', both over schema $\mathbf{T} = \mathbf{S}' \cup \{T_1, \dots, T_n\}$, such that $\mathcal{O}$ is logically equivalent to $\exists T_1 \dots T_n(\mathcal{O}' \wedge \neg q'')$. Note that, even though the size of q'' may be exponential in the size of $\mathcal{O}$, this is only because q'' may consist of exponentially many CQs, and each CQ is itself of polynomial size. If q is Boolean, it now follows that the $(\mathrm{GNFO}, \mathrm{UCQ})$ query $(\mathbf{S}, \mathcal{O}, q)$ is equivalent to the $(\mathrm{GFO}, \mathrm{UCQ})$ query $(\mathbf{S}, \mathcal{O}', q')$ where $q' = q \vee q''$. If, on the other hand, q is an n-ary UCQ, with $n > 0$, then one final step is needed: we turn the Boolean UCQ q'' into an n-ary UCQ by adding, for each free variable of q, an atom to the body of each CQ in q'', in all possible ways, thereby expressing that the variable takes a value from the active domain. Note that modification of q'' may involve an exponential blowup, but the resulting UCQ is still only single exponential in the size of $\mathcal{O}$, since each of its CQs is only of polynomial size.

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

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

4. CORRESPONDENCES TO MMSNP AND CSP

We first show that MDDlog captures coMMSNP and thus, by the results obtained in the previous section, the same is true for many OBDA languages based on UCQs. We also propose GMSNP, an extension of MMSNP inspired by frontier guarded DDlog, and show that $(\mathrm{GFO}, \mathrm{UCQ})$ and $(\mathrm{GNFO}, \mathrm{UCQ})$ capture coGMSNP, and that GMSNP has the same expressive power as a previously proposed extension of MMSNP called MMSNP_2. Then we turn to fragments of MDDlog that correspond to OBDA languages based on AQs and show that they capture CSPs (and generalizations thereof).

4.1. Correspondences to MMSNP

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

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

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

where we set $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi$ to true when $\mathfrak{D}$ is the empty instance (that is, $\mathsf{adom}(\mathfrak{D}) = \emptyset$) and Φ is a sentence. We call the resulting query language *coMMSNP*. Note that the equality atoms in MMSNP allow us to express queries that require some of the answer variables to be bound to the same domain element. This is needed for the following observation that coMMSNP has the same expressive power as MDDlog. We remark that equality atoms are not present in the original definition of MMSNP in [Feder and Vardi 1998], but they can easily be eliminated in MMSNP formulas without free variables by identifying variables that co-occur in an equality atom.

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

PROOF. We start with the translation from coMMSNP to MDDlog. Let $\Phi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ be an MMSNP formula over schema $\mathbf{S}$ with free variables $y_1, \dots, y_k$, and let $q_\Phi \in$ 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) every free variable y_j appears in some atom α_i or β_i, and (ii) if α_i is an equality atom, then it takes the form $y_j = y_\ell$. In fact, we can achieve (i) by replacing violating implications ψ with the set of implications ψ' that can be obtained from ψ by adding, for each variable y_j that is not present in ψ, a body atom $S(\mathbf{x})$ where S is a relation symbol that occurs in Φ and $\mathbf{x}$ is a tuple of variables that contains y_j once and otherwise only fresh variables that do not occur in Φ. To enforce condition (ii), for every equality atom $z_1 = z_2$ in which z_2 is not a free variable, we replace all occurrences of z_2 by z_1 and delete the atom.

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

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

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

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

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

Conversely, let Π be a k-ary MDDlog program whose set of EDB relations is $\mathbf{S}$. 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 relations in Π and $x_1, \dots, x_m$ the FO variables in Π that do not occur in the goal relation. Set $\Phi_\Pi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ where φ is the conjunction of all non-goal rules in Π plus the implication $\vartheta' \rightarrow \bot$ 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 ith position with y_i, and then conjunctively adding $y_i = y_j$ whenever the ith and jth positions in the rule head have the same variable. It can be verified that for all $\mathbf{S}$-instances $\mathfrak{D}$, we have $q_\Pi(\mathfrak{D}) = q_{\Phi_\Pi}(\mathfrak{D})$. $\Box$

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

We now consider OBDA languages based on the guarded fragment and GNFO. By Proposition 3.15, (GFO, UCQ) and $(\text{GNFO}, \text{UCQ})$ are strictly more expressive than MDDlog and we cannot use Proposition 4.1 to relate these query languages to the Feder-Vardi conjecture. Theorem 3.17 suggests 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 $X_1, \ldots, X_n$ SO variables of any arity, $x_1, \ldots, x_n$ 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 3.15 and Theorem 3.17, this yields the second statement of the following lemma; the first statement can be proved similarly to Proposition 4.1.

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

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

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

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

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

Although defined in a different way, GMSNP is essentially the same logic as MMSNP$_2$, which is studied in [Madelaine 2009]. 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 $\mathfrak{D} \models_\pi X(R(x_1, \ldots, x_n))$ if $R(\pi(x_1), \ldots, \pi(x_n)) \in \pi(X)$. We observe the following.

THEOREM 4.3. *GMSNP and MMSNP$_2$ have the same expressive power.*

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

COROLLARY 4.4. *MMSNP$_2$ is strictly more expressive than MMSNP.*

4.2. Correspondences to CSPs

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

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

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

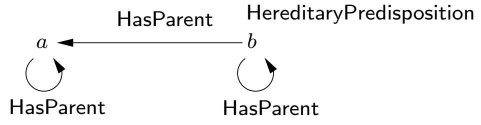
$$\{\mathbf{d} \in \mathsf{adom}(\mathfrak{D})^n \mid \forall (\mathfrak{B}, \mathbf{b}) \in \mathcal{F} : (\mathfrak{D}, \mathbf{d}) \not\to (\mathfrak{B}, \mathbf{b})\}.$$

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

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

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

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



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

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

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

(1) *($\mathcal{ALCU}$,**AQ**), ($\mathcal{SHIU}$,**AQ**), unary simple MDDlog, and generalized coCSP with one marked element;*
(2) *($\mathcal{ALC}$,**AQ**), ($\mathcal{SHI}$,**AQ**), unary connected simple MDDlog, and generalized coCSPs with one marked element such that all templates have the same instance;*
(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 corresponding CSP template(s) can be constructed in exponential time.

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

We treat Points 1-4 in reverse order, starting with Point 4 and proving that Boolean connected simple MDDlog and coCSP are equally expressive. Let Π be a Boolean connected simple MDDlog program. A *type for* Π is a set τ of IDBs and unary EDBs from Π. By $\mathsf{tp}(\Pi)$ we denote the set of all types for Π. We say that $\tau \in \mathsf{tp}(\Pi)$ is *realizable* if there is a model of Π in which some element d satisfies exactly the unary relation symbols from τ. Note that this is equivalent to the singleton instance $\{A(d) \mid A \in \tau\}$ being a model of Π. For binary $R \in \mathbf{S}$, we call a pair (τ_1, τ_2) of types *R-coherent* if there is a model $\mathfrak{D}$ of Π and two elements d_1, d_2 from $\mathfrak{D}$ such that $R(d_1, d_2) \in \mathfrak{D}$ and d_i satisfies exactly the unary relation symbols from τ_i, $i \in \{1, 2\}$. Note that this is equivalent to the two-element instance $\{R(d_1, d_2)\} \cup \bigcup_{i\in\{1,2\}}\{A(d_i) \mid A \in \tau_i\}$ being a model of Π. Given a set T of realizable types, we define the *canonical model* $\mathfrak{B}_T$ for T and Π by setting

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5. APPLICATIONS

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

5.1. Query Evaluation and Dichotomies

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

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

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

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

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

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

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

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

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

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

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

5.2. Query Containment

We apply the correspondence results from earlier to obtain results about the query containment problem for OBDA languages. Specifically, the following general containment problem was proposed in [Bienvenu et al. 2012] as a powerful tool for OBDA: given ontology-mediated queries $(\mathbf{S}, \mathcal{O}_i, q_i)$, $i \in \{1, 2\}$, decide whether for all S-instances $\mathfrak{D}$, we have $\mathsf{cert}_{q_1,\mathcal{O}_1}(\mathfrak{D}) \subseteq \mathsf{cert}_{q_2,\mathcal{O}_2}(\mathfrak{D})$.[7] Applications include the optimization of ontology-mediated queries and managing the effects on query answering of replacing an ontology with a new, updated version. In terms of OBDA languages such as ($\mathcal{ALC}$, UCQ), the

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

above problem corresponds to query containment in the standard sense: an S-query q_1 is *contained in* an S-query q_2, written $q_1 \subseteq q_2$, if for every S-instance $\mathfrak{D}$, we have $q_1(\mathfrak{D}) \subseteq q_2(\mathfrak{D})$. Note that there are also less general (and computationally simpler) notions of query containment in OBDA that do not fix the data schema [Calvanese et al. 1998].

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

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

We thus obtain the following result for OBDA languages.

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

Note that this result is considerably stronger than those in [Bienvenu et al. 2012], 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 5.6.

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

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

PROOF. We provide the proof of the lower bound, which is by reduction from the exponential grid tiling problem. An instance of this problem is given by a natural number $n > 0$ and a triple $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ with $\mathbb{T}$ a non-empty, finite set of *tile types* including a sequence of *initial tile types* $T_{0,0}, \ldots, T_{n,0}$ to be placed in the lower left corner, $H \subseteq \mathbb{T} \times \mathbb{T}$ a *horizontal matching relation*, and $V \subseteq \mathbb{T} \times \mathbb{T}$ a *vertical matching relation*. A *solution* for n and $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ is a map $f : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, 2^n - 1\} \to \mathbb{T}$ such that $f(0, 0) = T_{0,0}, \ldots, f(n, 0) = T_{n,0}$, $(f(i, j), f(i + 1, j)) \in \mathbb{H}$ for all $i < 2^n - 1$, and $(f(i, j), f(i, j + 1)) \in \mathbb{V}$ for all $j < 2^n - 1$. It is NEXPTIME-complete to decide whether an instance of the exponential grid tiling problem has a solution [Johnson 1990].

For the reduction, consider a problem instance given by the initial tile types $T_{0,0}, \ldots, T_{n,0}$ and $(\mathbb{T}, \mathbb{H}, \mathbb{V})$, where $\mathbb{T} = \{T_1, \ldots, T_p\}$. We construct a schema $\mathbf{S}_{\mathsf{grid}}$, two $\mathcal{ALC}$ ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, and a query $E(x)$ with E a concept name such that a solution for n and $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ exists if and only if $q_{\mathbf{S}_{\mathsf{grid}}, \mathcal{O}_1, E(x)} \subseteq q_{\mathbf{S}_{\mathsf{grid}}, \mathcal{O}_2, E(x)}$ if and only if $q_{\mathbf{S}_{\mathsf{grid}}, \mathcal{O}_1, \exists x. E(x)} \subseteq q_{\mathbf{S}_{\mathsf{grid}}, \mathcal{O}_2, \exists x. E(x)}$.

We first define the ontology $\mathcal{O}_2$ which encodes the $2^n \times 2^n$-grid. We use role names H and V to represent the horizontal and vertical axes of the grid and two binary counters $\mathbf{X}$ and $\mathbf{Y}$ for counting from 0 to $2^n - 1$. The counters use concept names $X_0, \ldots, X_{n-1}, \overline{X}_0, \ldots, \overline{X}_{n-1}$ and $Y_0, \ldots, Y_{n-1}, \overline{Y}_0, \ldots, \overline{Y}_{n-1}$ as their bits, respectively.

The ontology $\mathcal{O}_2$ contains the inclusions

$$\overline{X}_i \sqsubseteq \neg X_i \qquad \overline{Y}_i \sqsubseteq \neg Y_i$$

for $i = 0, \dots, n-1$. Counters are relevant only if the concept

$$\mathsf{Def} = (\bigsqcap_{0 \leq i \leq n-1} (X_i \sqcup \overline{X}_i)) \sqcap (\bigsqcap_{0 \leq i \leq n-1} (Y_i \sqcup \overline{Y}_i))$$

is true. $\mathcal{O}_2$ contains the following well-known inclusions stating that the value of the counter $\mathbf{X}$ is incremented when going to H-successors (and Def is true) and the value of the counter $\mathbf{Y}$ is incremented when going to V-successors (and Def is true). For $k = 0, \dots, n-1$:

$$\mathsf{Def} \sqcap \bigsqcap_{0 \leq j \leq k-1} X_j \sqsubseteq P_k \qquad \mathsf{Def} \sqcap \bigsqcup_{0 \leq j \leq k-1} \overline{X}_j \sqsubseteq Q_k$$

where

$$P_k = (X_k \rightarrow \forall H.(\mathsf{Def} \rightarrow \overline{X}_k)) \sqcap (\overline{X}_k \rightarrow \forall H.(\mathsf{Def} \rightarrow X_k))$$

and

$$Q_k = (X_k \rightarrow \forall H.(\mathsf{Def} \rightarrow X_k)) \sqcap (\overline{X}_k \rightarrow \forall H.(\mathsf{Def} \rightarrow \overline{X}_k))$$

and similarly for $\mathbf{Y}$ and V. $\mathcal{O}_2$ also states that the value of the counter $\mathbf{X}$ does not change when going to V-successors and the value of the counter $\mathbf{Y}$ does not change when going to H-successors. For $i = 0, \dots, n-1$:

$$\mathsf{Def} \sqcap X_i \sqsubseteq \forall V.(\mathsf{Def} \rightarrow X_i) \qquad \mathsf{Def} \sqcap \overline{X}_i \sqsubseteq \forall V.(\mathsf{Def} \rightarrow \overline{X}_i)$$

and similarly for $\mathbf{Y}$ and H. In addition, $\mathcal{O}_2$ states that when the counter $\mathbf{X}$ is $2^n - 1$, there is no H-successor (satisfying Def) and if the counter $\mathbf{Y}$ is $2^n - 1$, there is no V-successor (satisfying Def):

$$\mathsf{Def} \sqcap X_0 \sqcap \dots \sqcap X_{n-1} \sqsubseteq \forall H.(\mathsf{Def} \rightarrow \bot) \qquad \mathsf{Def} \sqcap Y_0 \sqcap \dots \sqcap Y_{n-1} \sqsubseteq \forall V.(\mathsf{Def} \rightarrow \bot)$$

This finishes the definition of $\mathcal{O}_2$. The schema $\mathbf{S}_{\mathsf{grid}}$ consists of all concept and role names in the signature of $\mathcal{O}_2$:

$$\mathbf{S}_{\mathsf{grid}} = \{H, V\} \cup \{X_0, \dots, X_{n-1}, \overline{X}_0, \dots, \overline{X}_{n-1}\} \cup \{Y_0, \dots, Y_{n-1}, \overline{Y}_0, \dots, \overline{Y}_{n-1}\}.$$

We now extend $\mathcal{O}_2$ to define the ontology $\mathcal{O}_1$. The signature of $\mathcal{O}_1$ extends that of $\mathcal{O}_2$ by using the tile types T_i from $\mathbb{T}$ as concept names and introducing a fresh concept name E. In addition to the inclusions from $\mathcal{O}_2$, the ontology $\mathcal{O}_1$ contains $n+1$ inclusions stating that tile type $T_{i,0}$ holds at $(i, 0)$ for $0 \leq i \leq n$. For example, for $i = 0$:

$$\neg X_0 \sqcap \dots \sqcap \neg X_{n-1} \sqcap \neg Y_0 \sqcap \dots \sqcap \neg Y_{n-1} \sqsubseteq T_{0,0}.$$

$\mathcal{O}_1$ further states that the tiling is complete on Def:

$$\mathsf{Def} \sqsubseteq \bigsqcup_{1 \leq i \leq p} T_i$$

Next, $\mathcal{O}_1$ states that if a tiling condition is violated, then the concept name E is true:

$$\begin{aligned} T_i \sqcap T_j &\sqsubseteq E && \text{for all } i \neq j \\ T_i \sqcap \exists H.T_j &\sqsubseteq E && \text{for all } (i,j) \notin \mathbb{H} \\ T_i \sqcap \exists V.T_j &\sqsubseteq E && \text{for all } (i,j) \notin \mathbb{V} \end{aligned}$$

Finally, E is propagated along H and V:

$$\exists H.E \sqsubseteq E \qquad \exists V.E \sqsubseteq E$$

This concludes the definitions of $\mathcal{O}_1$, $\mathcal{O}_2$, and $\mathbf{S}_{\mathsf{grid}}$. We show:

Claim. The following statements are equivalent:

(1) the exponential grid tiling problem given by n and $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ has no solution;
(2) $q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_1,E(x)}$ is not contained in $q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_2,E(x)}$;
(3) $q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_1,\exists x.E(x)}$ is not contained in $q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_2,\exists x.E(x)}$.

Assume first that $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ admits no $2^n \times 2^n$-tiling. We regard the pairs (i, j) with $0 \leq i \leq 2^n - 1$ and $0 \leq j \leq 2^n - 1$ as domain elements, and let $\mathfrak{D}_{\mathsf{grid}}$ be the $\mathbf{S}_{\mathsf{grid}}$-instance that contains the following facts:

— $H((i, j), (i + 1, j)) \in \mathfrak{D}_{\mathsf{grid}}$ for $0 \leq i < 2^n - 1$,
— $V((i, j), (i, j + 1)) \in \mathfrak{D}_{\mathsf{grid}}$ for $0 \leq j < 2^n - 1$.
— $X_k(i, j) \in \mathfrak{D}_{\mathsf{grid}}$ if the kth bit of i is 1,
— $\overline{X}_k(i, j) \in \mathfrak{D}_{\mathsf{grid}}$ if the kth bit of i is 0,
— $Y_k(i, j) \in \mathfrak{D}_{\mathsf{grid}}$ if the kth bit of j is 1, and
— $\overline{Y}_k(i, j) \in \mathfrak{D}_{\mathsf{grid}}$ if the kth bit of j is 0.

Since $\mathfrak{D}_{\mathsf{grid}}$ counts correctly, it is consistent with $\mathcal{O}_2$. Consequently, we have $q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_2,E(x)}(\mathfrak{D}_{\mathsf{grid}}) = \emptyset$ and $q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_2,\exists x.E(x)}(\mathfrak{D}_{\mathsf{grid}}) = 0$. However, since $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ admits no $2^n \times 2^n$-tiling, it follows that $(0, 0) \in q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_1,E(x)}(\mathfrak{D}_{\mathsf{grid}})$ and $q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_1,\exists x.E(x)}(\mathfrak{D}_{\mathsf{grid}}) = 1$. This establishes statements 2 and 3.

Conversely, assume that there is a solution given by $f : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, 2^n - 1\} \rightarrow \mathbb{T}$. We show that $q_{\mathbf{S}_{\mathsf{grid}},\mathcal{O}_1,\exists x.E(x)}(\mathfrak{D}) = 0$ for all $\mathbf{S}_{\mathsf{grid}}$-instances $\mathfrak{D}$ which are consistent with $\mathcal{O}_2$. Then statements 2 and 3 are refuted, as required.

Suppose that $\mathfrak{D}$ is consistent with $\mathcal{O}_2$. We define a model $(\mathsf{dom}, \mathfrak{D}')$ of $\mathcal{O}_1$ with $\mathfrak{D} \subseteq \mathfrak{D}'$ as follows. The domain of $\mathfrak{D}'$ coincides with $\mathsf{adom}(\mathfrak{D})$, and symbols from $\mathbf{S}_{\mathsf{grid}}$ are defined in $\mathfrak{D}'$ in exactly the same way as in $\mathfrak{D}$. It thus remains to add facts about the tile types T_k. Since $\mathfrak{D}$ is consistent with $\mathcal{O}_2$, every element $d \in \mathsf{adom}(\mathfrak{D})$ such that $\mathsf{Def}(d) \in \mathfrak{D}$ is associated with a unique value i for the counter $\mathbf{X}$ and j for the counter $\mathbf{Y}$. Include the fact $T_k(d)$ in $\mathfrak{D}'$ iff $f(i, j) = T_k$. Note that $\mathfrak{D}'$ contains no facts that involve E. It can be checked that the resulting structure is a model of $\mathcal{O}_1$. $\Box$

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

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

5.3. FO- and Datalog-Rewritability

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

S.[8] We observe that for ontology-mediated queries, FO-rewritability implies datalog-rewritability.

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

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

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

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

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

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

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

[8]By datalog query, we mean a DDLog query defined by a disjunction-free DDLog program, i.e., a DDLog program in which the head of each rule is a single atom.
[9]An NP algorithm for datalog-definability is implicit in [Freese et al. 2009], based on results from [Barto and Kozik 2009], see also [Bulatov 2009]. We thank Benoit Larose and Libor Barto for pointing this out.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

— Fix a sequence of distinct fresh variables $\mathbf{y} = y_1, \dots, y_n$ and replace each IDB atom $R(\mathbf{x})$ by $R(\mathbf{x}, \mathbf{y})$.
— Let $\sim$ be the least equivalence relation over the rule variables satisfying the following property: if $P_i(z)$ appears in the rule body, then $z \sim y_i$. Drop all P_i atoms and merge all variables appearing in the same equivalence class under $\sim$.
— For each variable y_i which appears only in the head, add $\mathsf{adom}(y_i)$ to the rule body.

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

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

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

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

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

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

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

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

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

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

PROOF. To decide whether a generalized CSP with marked elements given as a set of templates $\mathcal{F} = \{(\mathfrak{B}_1, \mathbf{b}_1), \dots, (\mathfrak{B}_n, \mathbf{b}_n)\}$ is FO-rewritable, it suffices to first guess a subset $\mathcal{F}' \subseteq \mathcal{F}$ and then to verify that (i) coCSP($(\mathfrak{B}, \mathbf{b})^c$) is FO-rewritable for each $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}'$, and (ii) for each $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}$ there is a $(\mathfrak{B}', \mathbf{b}') \in \mathcal{F}'$ such that $(\mathfrak{B}, \mathbf{b}) \rightarrow (\mathfrak{B}', \mathbf{b}')$. By Theorem 5.10, this can be done in NP. To see why this procedure is correct, suppose first that there is a subset $\mathcal{F}' \subseteq \mathcal{F}$ that satisfies conditions (i) and (ii). By the above observation, there exists a subset $\mathcal{F}'' \subseteq \mathcal{F}'$ of homomorphically incomparable instances such that coCSP($\mathcal{F}''$) is equivalent to coCSP($\mathcal{F}'$), which by (ii) is also equivalent to coCSP($\mathcal{F}$). Because of condition (i), we know that for every instance $(\mathfrak{B}, \mathbf{b}) \in \mathcal{F}''$, coCSP($(\mathfrak{B}, \mathbf{b})^c$) is FO-rewritable. We can thus apply Proposition 5.11 to conclude that coCSP($\mathcal{F}''$), or equivalently coCSP($\mathcal{F}$), is FO-rewritable. Conversely, if coCSP($\mathcal{F}$) is FO-rewritable, then we may guess a subset $\mathcal{F}' \subseteq \mathcal{F}$ of homomorphically incomparable instances such that coCSP($\mathcal{F}'$) is equivalent to coCSP($\mathcal{F}$). Condition (i) must be satisfied because of Proposition 5.11, and condition (ii) holds because coCSP($\mathcal{F}'$) and coCSP($\mathcal{F}$) are equivalent. Datalog-rewritability can be decided analogously. □

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

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

PROOF. We begin by showing how the exponential grid tiling problem can be reduced to FO-rewritability of queries from ($\mathcal{ALC}$, AQ) and ($\mathcal{ALC}$, BAQ). Fix $n > 0$, $T_{0,0}, \dots, T_{n,0}$, and $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ with $\mathbb{T} = \{T_1, \dots, T_p\}$. We construct a schema $\mathbf{S}$, an $\mathcal{ALC}$-ontology $\mathcal{O}$ and a query $A(x)$ such that there is a solution for n and $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ if and only if $q_{\mathbf{S},\mathcal{O},A(x)}$ is FO-rewritable if and only if $q_{\mathbf{S},\mathcal{O},\exists x.A(x)}$ is FO-rewritable.

We consider the ontology $\mathcal{O}_2$, its extension $\mathcal{O}_1$, and the schema $\mathbf{S}_{\mathsf{grid}}$ from the proof of Theorem 5.7. To define $\mathcal{O}$, we take a fresh role name S and two fresh concept names A and F and set

$$\mathcal{O} = \mathcal{O}_1 \cup \{\exists S.E \sqsubseteq E, E \sqcap F \sqsubseteq A\}$$

and $\mathbf{S} = \mathbf{S}_{\mathsf{grid}} \cup \{S, F\}$.

Claim. The following conditions are equivalent:

— $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ admits no $2^n \times 2^n$-tiling;
— $q_{\mathbf{S},\mathcal{O},A(x)}$ is not FO-rewritable;
— $q_{\mathbf{S},\mathcal{O},\exists x.A(x)}$ is not FO-rewritable.

First assume that $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ admits no $2^n \times 2^n$-tiling. To show that the query $q_{\mathbf{S},\mathcal{O},A(x)}$ is not FO-rewritable, it suffices to show that there is no finite set $\mathcal{D}$ of unary marked S-instances (an obstruction set) such that for every S-instance $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$: $d \in q_{\mathbf{S},\mathcal{O},A(x)}(\mathfrak{D})$ if and only if there exists $(\mathfrak{A}, a) \in \mathcal{D}$ such that $(\mathfrak{A}, a) \rightarrow (\mathfrak{D}, d)$. To this end, we define S-instances $\mathfrak{D}_m$ as the union of the instance $\mathfrak{D}_{\mathsf{grid}}$ from the proof of Theorem 5.7 and the facts

$$F(a_0), S(a_0, a_1), \dots, S(a_m, (0,0))$$

where $a_0, \dots, a_m$ are fresh elements. It can be checked that

— $a_0 \in q_{\mathbf{S},\mathcal{O},A(x)}(\mathfrak{D}_m)$ for all $m > 0$;

— $a_0 \notin q_{\mathbf{S},\mathcal{O},A(x)}(\mathfrak{D}'_m)$, where $\mathfrak{D}'_m$ results from $\mathfrak{D}_m$ by removing some fact (a_k, a_{k+1}) from $\mathfrak{D}_m$.

It follows immediately that no finite obstruction set exists. The argument for $q_{\mathbf{S},\mathcal{O},\exists x.A(x)}$ is similar.

Conversely, assume that $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ has a $2^n \times 2^n$-tiling given by $f : \{0, \dots, 2^n - 1\} \times \{0, \dots, 2^n - 1\} \to \mathbb{T}$. We have to show that there exists an FO-formula $\varphi(x)$ over $\mathbf{S}$ such that for all $\mathbf{S}$-instances $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$, $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \varphi[d]$ just in the case that $d \in q_{\mathbf{S},\mathcal{O},A(x)}(\mathfrak{D})$.

Note that one can easily construct a first-order sentence φ_{grid} over $\mathbf{S}_{\mathsf{grid}}$ such that for all $\mathbf{S}_{\mathsf{grid}}$-instances $\mathfrak{D}$,

$$(\mathsf{adom}, \mathfrak{D}) \models \varphi_{\mathsf{grid}} \quad \Leftrightarrow \quad \mathfrak{D} \text{ is not consistent with } \mathcal{O}_2.$$

We fix such a sentence φ_{grid} and show that the following statements are equivalent for every $\mathbf{S}$-instance $\mathfrak{D}$:

(1) $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \varphi_{\mathsf{grid}}$;
(2) $d \in q_{\mathbf{S},\mathcal{O},A(x)}(\mathfrak{D})$.

The direction from Point 1 to Point 2 is trivial. Conversely, assume that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \not\models \varphi_{\mathsf{grid}}$. Then $\mathfrak{D}$ is consistent with $\mathcal{O}_2$. We define a model $(\mathsf{dom}, \mathfrak{D}')$ of $\mathcal{O}$ with $\mathfrak{D}' \supseteq \mathfrak{D}$ as follows. The domain of $\mathfrak{D}'$ coincides with $\mathsf{adom}(\mathfrak{D})$. Symbols from $\mathbf{S}$ are defined in $\mathfrak{D}'$ in exactly the same way as in $\mathfrak{D}$. It thus remains to add the facts about the tile types T_k. Since $\mathcal{D}$ is consistent with $\mathcal{O}_2$, every element $d \in \mathsf{adom}(\mathfrak{D})$ with $\mathsf{Def}(d) \in \mathfrak{D}$ is associated with a unique value i for the counter $\mathbf{X}$ and j for the counter $\mathbf{Y}$. Include the fact $T_k(d)$ in $\mathfrak{D}'$ iff $f(i,j) = T_k$. Note that $\mathfrak{D}'$ contains no facts that involve E or A. It is readily checked that the resulting structure is a model of $\mathcal{O}$, as required.

We now give the proof for datalog-rewritability. For the reduction, let $n > 0$, $T_{0,0}, \dots, T_{n,0}$, and $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ be an instance of the exponential grid tiling problem with $\mathbb{T} = \{T_1, \dots, T_p\}$. We construct a schema $\mathbf{S}$, an $\mathcal{ALC}$-ontology $\mathcal{O}'$ and a query $A(x)$ such that $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ has a solution if and only if $q_{\mathbf{S},\mathcal{O}',A(x)}$ is datalog-rewritable if and only if $q_{\mathbf{S},\mathcal{O}',\exists x.A(x)}$ is datalog-rewritable.

We consider again the ontology $\mathcal{O}_2$, its extension $\mathcal{O}_1$, and the schema $\mathbf{S}_{\mathsf{grid}}$ from the proof of Theorem 5.7. To define $\mathcal{O}'$ we take fresh role names S and S' and fresh concept names P_1, P_2, P_3 and encode the 3-colorability problem as follows:

$$\begin{aligned}\mathcal{O}' = {} & \mathcal{O}_1 \cup \{\exists S.E \sqsubseteq E, \exists S'.A \sqsubseteq A\} \cup \{E \sqsubseteq P_1 \sqcup P_2 \sqcup P_3\} \cup \\ & \{P_i \sqcap P_j \sqsubseteq A \mid 1 \leq i < j \leq 3\} \cup \{P_i \sqcap \exists S'.P_i \sqsubseteq A \mid 1 \leq i \leq 3\}.\end{aligned}$$

We consider the schema $\mathbf{S} = \mathbf{S}_{\mathsf{grid}} \cup \{S, S'\}$.

Claim. The following conditions are equivalent:

— $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ admits no $2^n \times 2^n$-tiling;
— $q_{\mathbf{S},\mathcal{O}',A(x)}$ is not datalog-rewritable;
— $q_{\mathbf{S},\mathcal{O}',\exists x.A(x)}$ is not datalog-rewritable.

First assume that $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ admits no $2^n \times 2^n$-tiling. For any connected undirected graph G, we identify some v in G with $(0,0)$ and define a $\mathbf{S}$-instance $\mathfrak{D}'$ as the union of $\mathfrak{D}_{\mathsf{grid}}$ and the facts $S(d,d')$ for all d, d' in G and $S'(d,d')$ for every edge $\{d,d'\}$ in G. It can be checked that

— $(0,0) \in q_{\mathbf{S},\mathcal{O}',A(x)}(\mathfrak{D})$ if and only if G is not 3-colorable;
— $q_{\mathbf{S},\mathcal{O}',\exists x.A(x)}(\mathfrak{D}) = 1$ if and only if G is not 3-colorable.

It follows immediately that neither $q_{\mathbf{S},\mathcal{O}',A(x)}$ nor $q_{\mathbf{S},\mathcal{O}',\exists x.A(x)}$ are datalog-rewritable.

Conversely, if $(\mathbb{T},\mathbb{H},\mathbb{V})$ admits a $2^n \times 2^n$-tiling then one can show datalog-rewritability using exactly the same argument as was used for FO-rewritability. $\square$

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

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

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

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

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

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

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

6. SCHEMA-FREE ONTOLOGY-MEDIATED QUERIES

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

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

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

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

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

PROOF. As observed above, if $(\mathcal{ALC}, \text{BAQ})$ has no dichotomy between PTIME and CONP for schema-free queries, then it does not have such a dichotomy for fixed-schema queries. Therefore, by Theorem 5.1, it remains to prove that every query in coCSP is polynomially equivalent to some schema-free query $(\mathbf{S}^\infty, \mathcal{O}, \exists x.A(x))$. Assume a CSP template $\mathfrak{B}$ over schema $\mathbf{S}$ is given. To construct a polynomially equivalent schema-free query, we first construct an equivalent fixed-schema query $(\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALC}, \text{BAQ})$

and then modify the construction to obtain a polynomially equivalent schema-free query. The construction of $(\mathbf{S}, \mathcal{O}, q)$ is virtually identical to the translation of a CSP template into an MDDlog program in the proof of Theorem 4.6. Take for every $d \in \mathsf{adom}(\mathfrak{B})$ a fresh concept name $A_d \notin \mathbf{S}$ and a fresh concept name $A \notin \mathbf{S}$, set $q = \exists x.A(x)$, and let

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

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

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

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

Next we consider query containment for schema-free ontology-mediated queries. Recall that complexity upper bounds and decidability results transfer from fixed-schema query languages to schema-free query languages since $(\mathbf{S}^\infty, \mathcal{O}, q)$ behaves in exactly the same way as the fixed-schema query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S} = \mathsf{sig}(\mathcal{O}) \cup \mathsf{sig}(q)$. For the converse direction, we prove a general polynomial reduction of query containment for fixed-schema OBDA languages to query containment for the corresponding schema-free languages. We say that an ontology language $\mathcal{L}$ *can express emptiness* if for every relation symbol R, there exists a sentence $\varphi_{R=\emptyset}$ in $\mathcal{L}$ which states that R is empty. Clearly, all DLs introduced in this paper, UNFO, GFO, and GNFO can express emptiness.

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

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

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

The theorem follows if we can show that Q_1 is contained in Q_2 iff $(\mathbf{S}', \mathcal{O}_1, q_1)$ is contained in $(\mathbf{S}', \mathcal{O}_2', q_2)$, where $\mathbf{S}' = \mathbf{S} \cup \mathsf{sig}(\mathcal{O}_1) \cup \mathsf{sig}(q_1) \cup \mathsf{sig}(\mathcal{O}_2) \cup \mathsf{sig}(q_2)$. First assume $(\mathbf{S}', \mathcal{O}_1, q_1)$ is not contained in $(\mathbf{S}', \mathcal{O}_2', q_2)$ and $\mathfrak{D}$ is a $\mathbf{S}'$-instance with $\mathbf{a} \in \mathsf{cert}_{q_1,\mathcal{O}_1}(\mathfrak{D})$

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

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

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

PROOF. The NEXPTIME upper bound follows directly from the corresponding upper bound for fixed-schema queries (Theorem 5.16). To obtain the NEXPTIME lower bound, we modify the hardness proofs in Theorem 5.16 by replacing certain concept names by compound $\mathcal{ALC}$-concepts, as in the proof of Theorem 6.1 above.

For FO-rewritability, consider the ontology $\mathcal{O}$ from the NEXPTIME-hardness proof of Theorem 5.16. Take for any concept name $G \in \{T_1, \dots, T_p, E\}$ a fresh role name R_G and replace in $\mathcal{O}$ every occurrence of G by $\forall R_G.G$. Denote the resulting ontology by $\mathcal{O}^*$. Note that A is the only concept name not in S that has not been replaced by a compound concept. In fact, we cannot replace A by a compound concept since it is used in the atomic queries $A(x)$ and $\exists x.A(x)$. Nevertheless, using Fact 1 from the proof of Theorem 6.1 and the proof of Theorem 5.16, one can show that the following conditions are equivalent:

— $(\mathfrak{T}, H, V)$ admits no $2^n \times 2^n$-tiling;
— $q_{\mathbf{S}^\infty,\mathcal{O}^*,A(x)}$ is not FO-rewritable;
— $q_{\mathbf{S}^\infty,\mathcal{O}^*,\exists x.A(x)}$ is not FO-rewritable.

The only modification required in the proof is that now A can occur in the data instance and so $\varphi_{\mathsf{grid}} \vee A(x)$ and, respectively, $\varphi_{\mathsf{grid}} \vee \exists x.A(x)$ instead of just φ_{grid} are the required FO-rewritings if a tiling exists.

For datalog-rewritability, consider the ontology $\mathcal{O}'$ from the proof of Theorem 5.16. We take for any concept name $G \in \{T_1, \dots, T_p, E, P_1, P_2, P_3\}$ a fresh role name R_G and replace in $\mathcal{O}'$ every occurrence of G by $\forall R_G.G$. Denote the resulting ontology by $\mathcal{O}^{**}$. Again A is the only concept name not in S that has not been replaced by a compound concept. Using again Fact 1 from the proof of Theorem 6.1 and the proof of Theorem 5.16, one can show that the following conditions are equivalent:

— $(\mathfrak{T}, H, V)$ admits no $2^n \times 2^n$-tiling;
— $q_{\mathbf{S}^\infty,\mathcal{O}^{**},A(x)}$ is not datalog-rewritable;
— $q_{\mathbf{S}^\infty,\mathcal{O}^{**},\exists x.A(x)}$ is not datalog-rewritable.

In the proof, since now A can occur in the input instance and $\mathcal{O}^{**}$ contains $\exists H.A \sqsubseteq A$, one has to add the rules $P_A(x) \leftarrow A(x)$, $P_A(x) \leftarrow P_A(y) \wedge H(x,y)$, and $\mathsf{goal}(x) \leftarrow P_A(x)$ to the datalog program to obtain a rewriting if a tiling exists. □

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

[Bienvenu et al. 2013a]. Note that the reduction given above cannot be applied to $\mathcal{EL}$ since the concepts $\forall R.G$ are not $\mathcal{EL}$ concepts.

7. CONCLUSION

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

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

— Are FO-rewritability and datalog-rewritability decidable for standard OBDA languages based on UCQs such as $(\mathcal{ALC}, \text{UCQ})$? It follows from the results in Sections 3 and 4 that this problem is equivalent to the question whether FO-rewritability and datalog-rewritability are decidable for monadic disjunctive datalog and, equivalently, MMSNP.
— What is the computational complexity of deciding query containment for OBDA languages based on UCQs? For $(\mathcal{ALC}, \text{UCQ})$, decidability follows from the decidability of query containment for MMSNP, but tight complexity bounds do not appear to be known.
— Is a PTIME/NP-dichotomy for query evaluation in (GFO, UCQ) equivalent to the Feder-Vardi conjecture or is it possible to prove a non-dichotomy result? Is query containment decidable for (GFO, UCQ)? Are FO-rewritability and datalog-rewritability decidable for (GFO, UCQ)? As explained in Section 4, resolving these questions is equivalent to answering them for GMSNP and MMSNP_2.
— What is the status of OBDA languages such as $(\mathcal{S}, \text{UCQ})$ and of OBDA languages that are based on ontology languages with nominals? Do they have the same expressive power as natural fragments of disjunctive datalog?
— There are several open questions regarding the succinctness of OBDA languages. For example, is there really a double exponential succinctness gap beween $(\mathcal{ALCI}, \text{UCQ})$ (resp. $(\text{UNFO}, \text{UCQ})$) and MDDlog as well as between $(\text{GNFO}, \text{UCQ})$ and frontier-guarded DDlog or can the translations be improved by one exponential? Is the exponential blowup in the translation of frontier-guarded DDlog to (GFO, UCQ) avoidable?
— In this paper, we have focused on ontology-mediated queries based on atomic queries and UCQs. Other query languages frequently used in OBDA are conjunctive queries (CQs) and positive existential queries (PEQs). Since every PEQ is equivalent to a UCQ, all expressivity, decidability, and data complexity results trivially transfer from UCQs to PEQs. However, PEQs could well behave differently regarding succinctness. For CQs, the results in this paper imply that there is a dichotomy between PTIME and CONP for $(\mathcal{ALC}, \text{CQ})$ if and only if the Feder-Vardi conjecture holds. It is an interesting open problem whether there is a natural characterization of $(\mathcal{ALC}, \text{CQ})$ in terms of disjunctive datalog.

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

ACKNOWLEDGMENTS

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

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”.

REFERENCES

Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. 2011. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.* 36, 4 (2011), 23.

Albert Atserias. 2005. On Digraph Coloring Problems and Treewidth Duality. In *Proc. of LICS*. 106–115.

Franz Baader, Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. 2010. Query and Predicate Emptiness in Description Logics. In *Proc. of KR*.

Franz Baader, Sebastian Brandt, and Carsten Lutz. 2005. Pushing the $\mathcal{EL}$ Envelope. In *Proc. of IJCAI*. 364–369.

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

Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. 2011. Walking the Complexity Lines for Generalized Guarded Existential Rules. In *Proc. of IJCAI*.

Vince Bárány, Michael Benedikt, and Balder ten Cate. 2013. Rewriting Guarded Negation Queries. In *Proc. of MFCS*. Springer, 98–110.

Vince Bárány, Georg Gottlob, and Martin Otto. 2010. Querying the Guarded Fragment. In *Proc. of LICS*. 1–10.

Vince Bárány, Balder ten Cate, and Martin Otto. 2012. Queries with Guarded Negation. *PVLDB* 5, 11 (2012), 1328–1339.

Vince Bárány, Balder ten Cate, and Luc Segoufin. 2011. Guarded Negation. In *Proc. of ICALP*. 356–367.

Libor Barto and Marcin Kozik. 2009. Constraint Satisfaction Problems of Bounded Width. In *Proc. of FOCS*. 595–603.

Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. 2012. Query Containment in Description Logics Reconsidered. In *Proc. of KR*.

Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. 2013a. First-Order Rewritability of Atomic Queries in Horn Description Logics. In *Proc. of IJCAI*.

Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. 2013b. Ontology-based data access: a study through disjunctive datalog, CSP, and MMSNP. In *Proc. of PODS*. ACM, 213–224.

Manuel Bodirsky, Hubie Chen, and Tomás Feder. 2012. On the Complexity of MMSNP. *SIAM J. Discrete Math.* 26, 1 (2012), 404–414.

Andrei A. Bulatov. 2009. Bounded relational width. (2009). In preparation. Manuscript available from http://www.cs.sfu.ca/~abulatov/mpapers.html.

Andrei A. Bulatov. 2011. On the CSP Dichotomy Conjecture. In *Proc. of CSR*. 331–344.

Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2009. A general datalog-based framework for tractable query answering over ontologies. In *Proc. of PODS*. 77–86.

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

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2006. Data Complexity of Query Answering in Description Logics. In *Proc. of KR*. 260–270.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *J. Autom. Reasoning* 39, 3 (2007), 385–429.

Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 1998. On the Decidability of Query Containment under Constraints. In *Proc. of PODS*. 149–158.

Alexandros Chortaras, Despoina Trivela, and Giorgos B. Stamou. 2011. Optimized Query Rewriting for OWL 2 QL. In *Proc. of CADE*. 192–206.

Giuseppe De Giacomo and Maurizio Lenzerini. 1994. Boosting the Correspondence between Description Logics and Propositional Dynamic Logics. In *Proc. of AAAI*. 205–212.

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

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

Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. 2012. Query Rewriting for Horn-SHIQ Plus Rules. In *Proc. of AAAI*.

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

Tomás Feder and Moshe Y. Vardi. 1998. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM J. Comput.* 28, 1 (1998), 57–104.

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

Ralph Freese, Marcin Kozik, Andrei Krokhin, Miklós Maróti, Ralph KcKenzie, and Ross Willard. 2009. On Maltsev conditions associated with omitting certain types of local structures. (2009). In preparation. Manuscript available from http://www.math.hawaii.edu/~ralph/Classes/619/OmittingTypesMaltsev.pdf.

Georg Gottlob, Erich Grädel, and Helmut Veith. 2002. Datalog LITE: a deductive query language with linear time model checking. *ACM Trans. Comput. Log.* 3, 1 (2002), 42–79.

Georg Gottlob and Thomas Schwentick. 2012. Rewriting Ontological Queries into Small Nonrecursive Datalog Programs. In *Proc. of KR*.

Bernardo Cuenca Grau, Boris Motik, Giorgos Stoilos, and Ian Horrocks. 2013. Computing Datalog Rewritings Beyond Horn Ontologies. In *Proc. of IJCAI*.

Pavol Hell, Jaroslav Nešetřil, and X. Zhu. 1996. Duality and Polynomial Testing of Tree Homomorphisms. *Trans. of the AMS* 348, 4 (1996), 1281–1297.

André Hernich, Clemens Kupke, Thomas Lukasiewicz, and Georg Gottlob. 2013. Well-founded semantics for extended datalog and ontological reasoning. In *Proc. of PODS*, Richard Hull and Wenfei Fan (Eds.). ACM, 225–236.

Ian Horrocks and Ulrike Sattler. 1999. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *J. Log. Comput.* 9, 3 (1999), 385–410.

Ullrich Hustadt, Boris Motik, and Ulrike Sattler. 2007. Reasoning in Description Logics by a Reduction to Disjunctive Datalog. *J. Autom. Reasoning* 39, 3 (2007), 351–384.

David S. Johnson. 1990. *Handbook of Theoretical Computer Science*. MIT Press, Chapter A Catalog of Complexity Classes, 67–161.

Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, and Michael Zakharyaschev. 2012b. Exponential Lower Bounds and Separation for Query Rewriting. In *Proc. of ICALP*. 263–274.

Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev. 2012a. Conjunctive Query Answering with OWL 2 QL. In *Proc. of KR*.

Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. 2010. The Combined Approach to Query Answering in DL-Lite. In *Proc. of KR*.

Adila Krisnadhi and Carsten Lutz. 2007. Data Complexity in the EL family of DLs. In *Proc. of DL*.

Gábor Kun. 2007. Constraints, MMSNP, and Expander Structures. (2007). Available at http://arxiv.org/abs/0706.1701v1.

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

Benoit Larose, Cynthia Loten, and Claude Tardif. 2007. A Characterisation of First-Order Constraint Satisfaction Problems. *Logical Methods in Computer Science* 3, 4 (2007).

Carsten Lutz. 2007. Inverse Roles Make Conjunctive Queries Hard. In *Proc. of DL*.

Carsten Lutz. 2008. The Complexity of Conjunctive Query Answering in Expressive Description Logics. In *Proc. of IJCAR*. 179–193.

Carsten Lutz and Frank Wolter. 2012. Non-Uniform Data Complexity of Query Answering in Description Logics. In *Proc. of KR*.

Florent R. Madelaine. 2009. Universal Structures and the Logic of Forbidden Patterns. *Logical Methods in Computer Science* 5, 2 (2009).

Florent R. Madelaine and Iain A. Stewart. 2007. Constraint Satisfaction, Logic and Forbidden Patterns. *SIAM J. Comput.* 37, 1 (2007), 132–163.

Boris Motik. 2006. *Reasoning in description logics using resolution and deductive databases*. Ph.D. Dissertation.

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

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

Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. *J. Data Semantics* 10 (2008), 133–173.

Mariano Rodriguez-Muro and Diego Calvanese. 2012. High Performance Query Answering over DL-Lite Ontologies. In *Proc. of KR*. 308–318.

Riccardo Rosati and Alessandro Almatelli. 2010. Improving Query Answering over DL-Lite Ontologies. In *Proc. of KR*. 290–300.

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

Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. 2012. Type-elimination-based reasoning for the description logic SHIQbs using decision diagrams and disjunctive datalog. *Logical Methods in Computer Science* 8, 1 (2012).

Frantisek Simancik. 2012. Elimination of Complex RIAs without Automata. In *Proc. of DL*.

Balder ten Cate and Luc Segoufin. 2011. Unary negation. In *Proc. of STACS*. 344–355.

A. PROOFS FOR SECTION 3

A.1. Proofs for Section 3.1

For brevity, we say that an assignment π of elements of an instance $\mathfrak{D}$ to the (answer and quantified) variables of a CQ q is a *match of* q *in* $\mathfrak{D}$ if $\mathfrak{D}$ satisfies q under π.

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

PROOF. (continued) We establish here the correctness of the translation from $(\mathcal{ALC}, \text{UCQ})$ to MDDlog. Let m be the arity of $(\mathbf{S}, \mathcal{O}, q)$. We have to show the following.

Claim. For all $\mathbf{S}$-instances $\mathfrak{D}$ and $\mathbf{a} \in \mathsf{adom}(\mathfrak{D})^m$, we have $\mathbf{a} \in \mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$ iff $\mathbf{a} \in q_\Pi(\mathfrak{D})$.

"if". Assume that $\mathbf{a} \notin \mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$. Then there is a $(\mathsf{dom}', \mathfrak{D}') \in \mathsf{Mod}(\mathcal{O})$ such that $\mathfrak{D} \subseteq \mathfrak{D}'$ and $\mathbf{a} \notin q(\mathfrak{D}')$. For each $b \in \mathsf{adom}(\mathfrak{D})$, let $\mu(b)$ be the unique type realized at b in $\mathfrak{D}'$, that is,

$$\begin{aligned}\mu(b) = \{q' \in \mathsf{cl}(\mathcal{O}, q) \mid q' \text{ is Boolean and } \mathfrak{D}' \models q'\} \cup \\ \{\varphi \in \mathsf{cl}(\mathcal{O}, q) \mid \varphi \text{ has one free variable and } \mathfrak{D}' \models \varphi[b]\}.\end{aligned}$$

Let $\mathfrak{D}''$ be the instance that consists of the atoms in $\mathfrak{D}$ and the atom $P_{\mu(b)}(b)$ for each $b \in \mathsf{adom}(\mathfrak{D})$. It can be verified that $\mathfrak{D}''$ is a model of Π. In particular, it follows from the construction of $\mathfrak{D}''$ and the fact that $\mathbf{a} \notin q(\mathfrak{D}')$ that whenever a diagram $\delta(\mathbf{x})$ has a match π in $\mathfrak{D}''$ and $\delta(\mathbf{x})$ implies $q(\mathbf{x}')$, then $\pi(\mathbf{x}') \neq \mathbf{a}$. Since $\mathfrak{D}''$ is a model of Π and $\mathsf{goal}(\mathbf{a}) \notin \mathfrak{D}''$, we have $\mathbf{a} \notin q_\Pi(\mathfrak{D})$.

"only if". Assume that $\mathbf{a} \notin q_\Pi(\mathfrak{D})$, and let $\mathfrak{D}' \in \mathsf{Mod}(\Pi)$ be such that $\mathfrak{D} \subseteq \mathfrak{D}'$ and $\mathfrak{D}'$ does not contain $\mathsf{goal}(\mathbf{a})$. We assume w.l.o.g. that $\mathsf{adom}(\mathfrak{D}) = \mathsf{adom}(\mathfrak{D}')$. Note that the first two rules of Π ensure that for each $a \in \mathsf{adom}(\mathfrak{D})$, there is a unique type $\mu(a)$ such that $P_{\mu(a)}(a) \in \mathfrak{D}'$. The second rule further ensures that for each $a \in \mathsf{adom}(\mathfrak{D})$, there is a tree-shaped model $(\mathsf{dom}_a, \mathfrak{D}_a)$ of $\mathcal{O}$ in which $\mu(a)$ is realized at root a. We may assume that these models have disjoint domains. Let $(\mathsf{dom}'', \mathfrak{D}'')$ be the relational structure obtained by first taking the union of $(\mathsf{dom}_a, \mathfrak{D}_a)_{a \in \mathsf{adom}(\mathfrak{D})}$, and then adding all facts from $\mathfrak{D}$. To prove that $\mathbf{a} \notin \mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$, it suffices to show that

(i) $(\mathsf{dom}'', \mathfrak{D}'')$ is a model of $\mathcal{O}$, and
(ii) $\mathbf{a} \notin q(\mathfrak{D}'')$.

For Point (i), let $\mu(d)$ be the unique type realized by d in $(\mathsf{dom}_a, \mathfrak{D}_a)$, for all $d \in \mathsf{dom}_a$ and $a \in \mathsf{dom}(\mathfrak{D})$. It can be shown by induction on the structure of C that for all concepts $C \in \mathsf{sub}(\mathcal{O}) \cup \{A \in \mathbf{S} \mid A \text{ unary}\}$ and all $d \in \mathsf{dom}''$, we have

$$(\mathsf{dom}'', \mathfrak{D}'') \models C(d) \quad \text{iff} \quad C \in \mu(d) \tag{1}$$

Since each $(\mathsf{dom}_a, \mathfrak{D}_a)$ is a model of $\mathcal{O}$ and each $\mu(d)$ is realized in some such model, this implies Point (i) as desired.

It thus remains to establish Point (ii). Assume to the contrary that there is a disjunct $q'(\mathbf{x}')$ of q such that $\mathbf{a} \in q'(\mathfrak{D}'')$, that is, there is a match π of $q'(\mathbf{x}')$ in $\mathfrak{D}''$ such that $\pi(\mathbf{x}') = \mathbf{a}$. We define a diagram $\delta(\mathbf{x})$ that contains

(a) all atoms $A(x) \in q'$ such that $\pi(x) \in \mathsf{adom}(\mathfrak{D})$,
(b) all atoms $R(x, y) \in q'$ such that $\pi(x), \pi(y) \in \mathsf{adom}(\mathfrak{D})$,
(c) all atoms $P_{\mu(a)}(x)$ such that x is a variable in q' and $\pi(x) = a \in \mathsf{adom}(\mathfrak{D})$;
(d) all atoms $P_{\mu(a)}(z_a)$ (with z_a a fresh variable) such that $P_{\mu(a)}(a) \in \mathfrak{D}'$ and there is some variable x in q' such that $\pi(x) \in \mathsf{dom}_a$.

Atoms of type (d) are used to handle the case in which a Boolean subquery q'' of q' is mapped to $\mathfrak{D}_a$, but the element a does not itself belong to the image of π. We remark that the mapping π can be straightforwardly extended to a match for $\delta(\mathbf{x})$ in $\mathfrak{D}'$ by setting $\pi(z_a) = a$ for each variable z_a. Since $\delta(\mathbf{x})$ is satisfied in $\mathfrak{D}'$ under π, $\pi(\mathbf{x}') = \mathbf{a}$, and $\mathsf{goal}(\mathbf{a}) \notin \mathfrak{D}'$, by the last rule of Π, we can obtain the desired contradiction by showing that $\delta(\mathbf{x})$ implies $q'(\mathbf{x}')$.

Thus, let $(\mathsf{dom}, \mathfrak{B}) \in \mathsf{Mod}(\mathcal{O})$ be a type-coherent structure, and let τ be a match of $\delta(\mathbf{x})$ in $\mathfrak{B}$. We have to show that q' has a match in $\mathfrak{B}$ that agrees with τ on $\mathbf{x}'$. To this end, consider the following CQs:

— q_0 is the restriction of q' to those variables that π maps to elements of $\mathfrak{D}$;
— for each connected component p of q' that π maps completely inside some $\mathfrak{D}_a$ the query q_p that is obtained from p by identifying all variables that π maps to the same element; it is not hard to see that $q_p \in \mathsf{tree}(q)$;
— for each atom $R(x, y) \in q'$ with $\pi(x) = a \in \mathsf{adom}(\mathfrak{D})$ and $\pi(y) \in \mathsf{dom}_a \setminus \mathsf{adom}(\mathfrak{D})$, the CQ $q_{R(x,y)}$ is $\{R(x, y)\} \cup \widehat{q}'|_y$ where $\widehat{q}'$ is obtained from q' by identifying all variables that π maps to the same element and $\widehat{q}'|_y$ is the restriction of $\widehat{q}'$ to those variables reachable from y (in $\widehat{q}'$ viewed as a directed graph). Note that x is the only variable in $q_{R(x,y)}$ that can be an answer variable. Again, it is not hard to see that $q_{R(x,y)} \in \mathsf{tree}(q)$.

The queries of the form q_p do not contain any variables from $\mathbf{x}'$, and if a query $q_{R(x,y)}$ contains a variable from $\mathbf{x}'$, then it must be x (which is then an answer variable). To show that q' has a match in $\mathfrak{B}$ that agrees with τ on $\mathbf{x}'$, it thus suffices to show that (i) q_0 is satisfied in $\mathfrak{B}$ under τ, (ii) each q_p has a match in $\mathfrak{B}$, and (iii) each $q_{R(x,y)}$ has a match τ' in $\mathfrak{B}$ such that $\tau'(x) = \tau(x)$. In fact, these matches can then easily be assembled into the required match of q' in $\mathfrak{B}$.

(i) Immediate since all atoms in q_0 also belong to $\delta(\mathbf{x})$.

(ii) Take a query q_p. Then p is a connected component of q' that π maps completely inside some $\mathfrak{D}_a$. Since $\mathfrak{D}_a$ realizes the type $\mu(a)$ at a, we must have $q_p \in \mu(a)$. By construction, $\delta(\mathbf{x})$ contains an atom $P_{\mu(a)}(x)$. Since τ is a match for $\delta(\mathbf{x})$ in $\mathfrak{B}$, we must have $P_{\mu(a)}(\tau(x)) \in \mathfrak{B}$. As $\mathfrak{B}$ is type-coherent, we can find a match of q_p in $\mathfrak{B}$.

(iii) Take a query $q_{R(x,y)}$. First assume that x is an answer variable. Then $q_{R(x,y)}$ has a match in $\mathfrak{D}_a$ that maps x to a. Since $\mathfrak{D}_a$ realizes the type $\mu(a)$ at a, we must have $q_{R(x,y)} \in \mu(a)$. By construction, $\delta(\mathbf{x})$ contains the atom $P_{\mu(a)}(x)$. Since τ is a match for $\delta(\mathbf{x})$ in $\mathfrak{B}$, we must have $P_{\mu(a)}(\tau(x)) \in \mathfrak{B}$. Then, using the fact that $\mathfrak{B}$ is type-coherent, we can find a match τ' of $q_{R(x,y)}$ in $\mathfrak{B}$ such that τ' maps x to $\tau(x)$. The case where x is not an answer variables is similar, we then have $q_{R(x,y)} \in \mu(a)$ and $P_{\mu(a)}(z_a) \in \delta(\mathbf{x})$. □

The next proposition completes the proof of Theorem 3.7.

PROPOSITION A.1. *There is a polynomial* p *such that for every* $k > 0$,

(1) *there is a query* $(Q_k)_{k \geq 1}$ *from* $(\mathcal{ALCI}, \textit{UCQ})$ *such that* $|Q_k| \leq p(k)$ *and* $Q_k(\mathfrak{C}_\ell) = 1$ *for all* $\ell \geq k$ *and*

(2) *for every query* Q *from* $(\mathcal{ALCHU}, \textit{UCQ})$ *such that* $Q(\mathfrak{C}_\ell) = 1$ *iff* $\ell \geq k$, *we have* $|Q| \geq 2^{k/3}$.

PROOF. (sketch) The queries $Q_k = (\mathbf{S}, \mathcal{O}_k, q_k)$ from Point 1 can be constructed by realizing a counter with 2^k bits that assigns a value from the range $0 \ldots 2^{2^k} - 1$ to each node of a counting instance and is incremented along each $R; R^-$-edge of the instance; the query Q_k then returns 'true' if the counter, which from now on we refer to as the *path counter*, reaches maximum value $2^{2^k} - 1$. The definition of the required

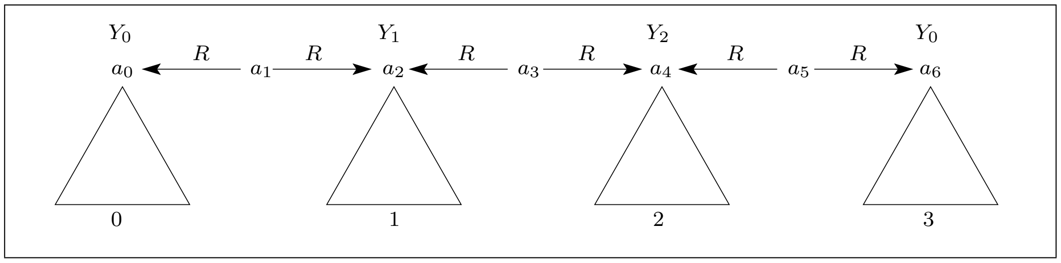


Fig. 2. Counting instance with 'counting trees'.

ontologies and queries is rather technical and almost identical to those introduced in [Lutz 2007; 2008] to prove that query evaluation in $(\mathcal{ALCI}, \text{UCQ})$ is 2EXPTIME-hard regarding combined complexity. For this reason, we only give a high-level sketch of the construction here and refer to [Lutz 2007; 2008] for full details.

The general idea is that the ontology $\mathcal{O}_k$ provides the 'infrastructure' for the path counter while the actual query q_k plays a central role in properly incrementing the counter, and also makes sure that the maximum counter value is reached. The counting infrastructure consists of a binary tree of depth k below each element a_i of a counting instance with i even (that is, the elements which are labeled with Y_i, for some $i \in \{0, 1, 2\}$). Figure 2 shows the counting instance from Figure 1 extended with these 'counting trees', which are enforced by existential quantifiers in $\mathcal{O}_k$. Each counting tree stores one value of the path counter in binary, with one bit stored at every leaf of the tree using concept names B (bit is one) and $\overline{B}$ (bit is zero). When the maximum value $2^{2^k} - 1$ of the counter is reached, then $\mathcal{O}_k$ sets a marker M at the root of the corresponding tree, and one disjunct of the UCQ q_k checks for the existence of that marker. It does not seem possible to enforce that the path counter has value 0 at the beginning of the path, so we simply start with any value selected non-deterministically by disjunctions in $\mathcal{O}_k$. Since certain answer semantics quantifies over *all* possible extensions of the instance that satisfy $\mathcal{O}_k$, this is clearly sufficient.

It remains to ensure that the counter is incremented properly, which is not possible using $\mathcal{O}_k$ alone, but requires an interplay between $\mathcal{O}_k$ and q_k. To start with, we store a second counter value at each counting tree that, just like the path counter, can take values from the range $0 \dots 2^{2^k} - 1$. This is again done by storing one bit at each leaf node, via concept names B^d and $\overline{B^d}$. We call this second counter the *delayed counter* and, using $\mathcal{O}_k$, make sure locally at each tree that the value of the path counter can be obtained from the value of the delayed counter by incrementation. To ensure proper incrementation of the path counter along the path, it thus remains to enforce that the path counter of each counting tree has the same value as the delayed counter of the next counting tree along the path.

This is done by adding disjuncts to q_k that result in Q_k returning 'true' whenever there are two subsequent counting trees whose path counter and delayed counter do *not* agree in the described way. The exact details of how to construct these additional disjuncts of q_k are rather technical [Lutz 2007; 2008]. The involved queries are highly symmetric for being able to transport the values of each one of the exponentially many bits, which is the reason for working with the symmetric $R; R^-$-sequences in counting instances instead of with R-sequences. The labeling of the nodes of a counting instance with Y_0, Y_1, Y_2 is necessary to distinguish the next counting tree along the path from the previous one. In OBDA languages without inverse roles such as $(\mathcal{ALCHU}, \text{UCQ})$, the construction fails because the symmetry of the queries essentially relies on inverse roles being available.

To show Point 2, assume to the contrary that there is a $k > 0$ and a query $Q = (\mathbf{S}, \mathcal{O}, q)$ from $(\mathcal{ALCHU}, \text{UCQ})$ such that $|Q| < 2^{k/3}$ and $Q(\mathfrak{C}_\ell) = 1$ iff $\ell \geq k$. Let $m := 2^{2^k} - 1$ and consider the counting instance $\mathfrak{C}_m$ of length m, which has elements $a_0, \dots, a_{2m}$. The ontology-mediated query Q must evaluate to false on $\mathfrak{C}_m$, and thus there is a relational structure $\mathfrak{B} \in \mathsf{Mod}(\mathcal{O})$ that extends $\mathfrak{C}_m$ and such that $q(\mathfrak{B}) = 0$. Since $\mathcal{ALCHU}$-ontologies and the non-existence of a homomorphism from q are preserved under unraveling, we can assume as in the proof of Theorem 3.3 that $\mathfrak{B}$ is a forest extension of $\mathfrak{C}_m$, that is, $\mathfrak{B}$ is obtained from $\mathfrak{C}_m$ by (essentially) attaching a tree-shaped structure to each element. Since $\mathcal{ALCHU}$ includes role hierarchies, we need a slightly more general notion of 'forest extension', which (i) allows successors in trees to be connected by more than one role name—we might have both $S(a, b)$ and $T(a, b)$ when b is a successor of a and $S \neq T$; and (ii) where additional edges $S(a_i, a_{i+1})$ and $S(a_{i+1}, a_i)$ might be added to $\mathfrak{C}_m$ to satisfy role hierarchy statements in $\mathcal{O}$. Let $\mathsf{cl}(\mathcal{O}, q)$ be defined as in the proof of Theorem 3.3 (where again trees admit multi-role successors) and recall that $|\mathsf{cl}(\mathcal{O}, q)| \leq |\mathcal{O}| + |q|$. For $0 \leq i \leq 2m$, we use $\mathsf{tp}_{\mathfrak{B}}(a_i)$ to denote the *type* of a_i in $\mathfrak{B}$, that is, the set of those $C \in \mathsf{cl}(\mathcal{O}, q)$ such that C is true at a_i in $\mathfrak{B}$. Note that the set $\{\mathsf{tp}_{\mathfrak{B}}(a_i) \mid 0 \leq i \leq 2m\}$ has size at most $2^{|Q|}$. Since $2^{3|Q|} < 2^k = 2m$, we must find positions $i, j \in \{0, \dots, 2m - 1\}$ with $i < j - 1$ and such that the following conditions are satisfied:

(1) $\mathsf{tp}_{\mathfrak{B}}(a_i) = \mathsf{tp}_{\mathfrak{B}}(a_j)$ and $\mathsf{tp}_{\mathfrak{B}}(a_{i+1}) = \mathsf{tp}_{\mathfrak{B}}(a_{j+1})$;
(2) $S(a_i, a_{i+1}) \in \mathfrak{B}$ iff $S(a_j, a_{j+1}) \in \mathfrak{B}$ for all binary relations S from $\mathbf{S}$ and $\mathcal{O}_k$;
(3) $S(a_{i+1}, a_i) \in \mathfrak{B}$ iff $S(a_{j+1}, a_j) \in \mathfrak{B}$ for all binary relations S from $\mathbf{S}$ and $\mathcal{O}_k$.

Now let $\mathfrak{B}'$ be obtained from $\mathfrak{B}$ by duplicating the sequence $a_{i+1}, \dots, a_j$, including the attached trees. The new structure $\mathfrak{B}'$ has 'root elements' $a_0, \dots, a_j, a'_{i+1}, \dots, a'_j, a_{j+1}, \dots, a_{2m}$ where each a'_ℓ roots an isomorphic copy of the tree that can be found at a_ℓ in $\mathfrak{B}$. It is easy to see that $\mathfrak{B}'$ is a forest extension of some counting instance $\mathfrak{C}_\ell$ of length at least 2^{2^k}. Moreover, using the similarity of $\mathfrak{B}$ and $\mathfrak{B}'$ it can be proved that $\mathfrak{B}'$ is from $\mathsf{Mod}(\mathcal{O})$, and that $q(\mathfrak{B}') = 0$. Thus Q evaluates to false on $\mathfrak{C}_\ell$, in contradiction to the assumed properties of the ontology-mediated query Q. □

A.2. Proofs for Section 3.2

Theorem 3.14 (UNFO, UCQ) has the same expressive power as MDDlog.

PROOF. (continued) We show that, for every instance $\mathfrak{D}$ and elements $\mathbf{a} \in \mathsf{adom}(\mathfrak{D})$, we have $\mathbf{a} \in \mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$ if and only if $\mathbf{a} \in q_\Pi(\mathfrak{D})$. The "if" direction proceeds exactly as in the proof of Theorem 3.3, so here we focus on the "only if" direction.

"only if". Assume that $\mathbf{a} \notin q_\Pi(\mathfrak{D})$ and let $\mathfrak{D}'$ be a model of Π with $\mathfrak{D} \subseteq \mathfrak{D}'$ that does not contain $\mathsf{goal}(\mathbf{a})$. For each $a \in \mathsf{adom}(\mathfrak{D})$, let $\mu(a)$ be the unique type such that $P_{\mu(a)}(a) \in \mathfrak{D}'$, and let $(\mathsf{dom}_a, \mathfrak{D}_a)$ be a model of $\mathcal{O}$ in which $\mu(a)$ is realized at a. Note that such a model must exist because otherwise the diagram $P_{\mu(a)}(x)$ would be non-realizable and Π would include a rule $\bot \leftarrow P_{\mu(a)}(x)$. We may assume that these models have disjoint domains. Let $(\mathsf{dom}'', \mathfrak{D}'')$ be obtained by first taking the union of $(\mathsf{dom}_a, \mathfrak{D}_a)_{a \in \mathsf{adom}(\mathfrak{D})}$, and then adding to it all facts of $\mathfrak{D}$. We show that

(i) $(\mathsf{dom}'', \mathfrak{D}'')$ is a model of $\mathcal{O}$, and
(ii) $\mathbf{a} \notin q(\mathfrak{D}'')$.

We start with the first claim. Let $\mu(d)$ be the unique type realized by d in $(\mathsf{dom}_a, \mathfrak{D}_a)$, for all $d \in \mathsf{dom}_a$. We show the following by induction on the structure of φ:

$(*)$ For all $\varphi \in \mathsf{cl}_k(\mathcal{O}, q)$ and $d \in \mathsf{dom}''$, we have that $\varphi \in \mu(d)$ iff $(\mathsf{dom}'', \mathfrak{D}'') \models \varphi[d]$.

Note that φ may be either a sentence or a formula with exactly one free variable, and in the former case, we interpret $\varphi[d]$ as φ. Since all types $\mu(d)$ must include the sentence $\mathcal{O}$, $(*)$ implies (i).

The base case ($\varphi = \top$) and the inductive step for formulas of the form $\neg\psi(x)$ are omitted since they are straightforward. Thus, let φ be a formula from $\mathsf{cl}_k(\mathcal{O}, q)$ of the form $\exists \mathbf{y} \bigwedge_i \psi_i(x, \mathbf{y})$, and let $d \in \mathsf{dom}_a$. We may assume that φ is connected, meaning that the graph whose nodes are the subformulas ψ_i and containing an edge between ψ_i and ψ_j if they share a variable, is connected. This is because, if φ is not connected, then the claim follows immediately from the analogous claims for each of the connected components of φ. We present the proof for the case where φ has answer variable x (the argument for sentences is similar).

First suppose that $\varphi \in \mu(d)$, which means $(\mathsf{dom}_a, \mathfrak{D}_a) \models \varphi[d]$. It follows that there is an assignment π of elements of dom_a to the variables $x, \mathbf{y}$ such that $\pi(x) = d$ and for every i, $(\mathsf{dom}_a, \mathfrak{D}_a) \models \psi_i(\pi(x, \mathbf{y}))$. If ψ_i is an atomic formula, then using the fact that $\mathfrak{D}_a \subseteq \mathfrak{D}''$, we obtain $(\mathsf{dom}'', \mathfrak{D}'') \models \psi_i(\pi(x, \mathbf{y}))$. If ψ_i is not atomic, then it must have at most one free variable u. We thus have that $(\mathsf{dom}_a, \mathfrak{D}_a) \models \psi_i[\pi(u)]$, so $\psi_i \in \mu(\pi(u))$. Applying the induction hypothesis, we obtain $(\mathsf{dom}'', \mathfrak{D}'') \models \psi_i[\pi(u)]$. It follows that π is a satisfying assignment for φ in $(\mathsf{dom}'', \mathfrak{D}'')$, hence $(\mathsf{dom}'', \mathfrak{D}'') \models \varphi[d]$.

Conversely, suppose $(\mathsf{dom}'', \mathfrak{D}'') \models \varphi[d]$, that is, $(\mathsf{dom}'', \mathfrak{D}'')$ satisfies $\bigwedge_i \psi_i(x, \mathbf{y})$ for some assignment π of elements of dom'' to the variables $x, \mathbf{y}$ such that $\pi(x) = d$. First assume that the image of π is entirely contained in dom_a. Using the induction hypothesis to treat the non-atomic ψ_i as before, we then get that $(\mathsf{dom}_a, \mathfrak{D}_a) \models \varphi[d]$, hence $\varphi \in \mu(d)$ as required.

Next suppose that the image of π is not wholly contained in dom_a, and let I be the set consisting of the elements of $\mathsf{adom}(\mathfrak{D})$ that are in the range of π. By the connectedness assumption and the fact that $d \in \mathsf{dom}_a$, the set I contains a. In what follows, we will consider a number of formulas obtained from φ by identifying variables and removing atoms. It will follow from the definition of $\mathsf{cl}_k(\mathcal{O}, q)$ that each of these formulas again belongs to $\mathsf{cl}_k(\mathcal{O}, q)$, and hence, is subject to the induction hypothesis. Let φ' be obtained from φ by identifying all variables z, z' such that $\pi(z) = \pi(z') \in I$. We assume that the free variable x retains its name, and use ψ_i' to denote the conjunct of φ' which corresponds to ψ_i. For each $b \in I$, let $z_b \in \mathbf{y} \cup \{x\}$ be the unique variable in φ' with $\pi(z_b) = b$. Let φ_b' be the restriction of φ' to those ψ_i' which contain only variables z with $\pi(z) \in \mathsf{dom}_b$, with free variable z_b. We have $(\mathsf{dom}'', \mathfrak{D}'') \models \varphi_b'[b]$ via the restriction of π to the variables in φ_b', thus, by the earlier argument (since all witnessing elements are contained in dom_b), we have $\varphi_b' \in \mu(b)$. Let φ_0' be φ', but with free variable z_a instead of x. Note that $(\mathsf{dom}'', \mathfrak{D}'') \models \varphi_0'[a]$.

Consider the diagram δ obtained by taking the restriction of $\mathfrak{D}'$ to I, and then replacing each $b \in I$ with z_b. Since δ is made true by $\mathfrak{D}'$, and $\mathfrak{D}'$ is a model of Π, we have that δ is a realizable diagram. Moreover, using the fact that $P_{\mu(b)}(z_b) \in \delta$ and $\varphi_b' \in \mu(b)$ for every $b \in I$, one can show that the diagram δ implies the query φ_0'. This together with the realizability of δ yields $\varphi_0' \in \mu(a)$, hence $(\mathsf{dom}_a, \mathfrak{D}_a) \models \varphi_0'[a]$. Let π' be a satisfying assignment of φ_0' in $\mathfrak{D}_a$ such that $\pi'(z_a) = a$. We use π' to construct a satisfying assignment π'' of φ' mapping x to d, such that the range of π'' lies entirely inside dom_a. The assignment π'' is defined as follows: for all u with $\pi(u)$ in dom_a, set $\pi''(u) = \pi(u)$; for all other u, set $\pi''(u) = \pi'(u)$. To see that π'' is indeed a satisfying assignment of φ', note that each conjunct of φ' contains, besides z_a, either only variables u with $\pi(u) \in \mathsf{dom}_a$, or only variables u with $\pi(u) \notin \mathsf{dom}_a$. The former conjuncts are satisfied because π is a match, and the latter conjuncts are satisfied because π' is a match. Moreover, $\pi''(x) = d$. Therefore, $(\mathsf{dom}_a, \mathfrak{D}_a) \models \varphi[d]$ and hence $\varphi \in \mu(d)$ as required.

Finally, we can show (ii) in a similar way. We suppose, for the sake of contradiction, that $\mathbf{a} \in q(\mathfrak{D}'')$ under some assignment π to the existentially quantified variables in q. Let $\mathbf{b}$ be the elements of $\mathsf{adom}(\mathfrak{D})$ belonging to the range of π (here again we focus on the case in which q is connected and contains at least one free variable). Then, in the same way as above, we can decompose q into unary subqueries q_b that are satisfied in the different subinstances $\mathfrak{D}_b$ with $b \in \mathbf{b}$, and conclude that $q_b \in \mu(b)$ for each $b \in \mathbf{b}$. We can then show that the diagram obtained by taking all facts in $\mathfrak{D}'$ over elements in $\mathbf{b}$ and replacing each $b \in \mathbf{b}$ by z_b implies the query q. This yields the desired contradiction since $\mathfrak{D}'$ is a model of Π. $\square$

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

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

PROOF. (continued) It remains to translate (GNFO, UCQ) to frontier-guarded DDlog. Recall that we used a specific normal form for UNFO sentences in the proof of Theorem 3.14. For GNFO, we can use an analogous normal form. Specifically, we can assume that $\mathcal{O}$ is generated by the following grammar:

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

where each ψ_i is either a relational atom or a formula generated by the same grammar whose free variables are among $x, \mathbf{y}$. The "guard" α is an atomic formula, possibly an equality, containing *all variables* in $\mathbf{x}$.

Let $\mathsf{sub}(\mathcal{O})$ be the set of all subformulas of $\mathcal{O}$. Let k be the maximum of the number of variables in $\mathcal{O}$ and the number of variables in q. For $\ell \geq 0$, we denote by $\mathsf{cl}^\ell_k(\mathcal{O}, q)$ the set of all formulas $\chi(\mathbf{x})$ with $\mathbf{x} = (x_1, \dots, x_\ell)$ of the form

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

with $\mathbf{y} = (y_1, \dots, y_m)$, $m + \ell \leq k$, $n \leq |\mathcal{O}| + |q|$, and such that each ψ_i is either an atomic formula that uses a symbol from q or is of the form $\chi(\mathbf{z})$ for some $\chi(\mathbf{z}') \in \mathsf{sub}(\mathcal{O})$. Note that the length of each formula in $\mathsf{cl}^\ell_k(\mathcal{O}, q)$ is polynomial in $|\mathcal{O}| + |q|$, and consequently, the cardinality of $\mathsf{cl}^\ell_k(\mathcal{O}, q)$ is single exponential in $|\mathcal{O}| + |q|$.

A *guarded ℓ-type* τ is a subset of $\mathsf{cl}^\ell_k(\mathcal{O}, q)$ that contains at least one atomic relation (possibly equality) containing all variables $x_1, \dots, x_\ell$, and also contains the sentence $\mathcal{O}$ itself. We denote the set of all guarded ℓ-types by $\mathsf{type}_\ell(\mathcal{O})$. Note that, by definition, there are no guarded ℓ-types for ℓ greater than the maximal arity of a relation symbol from $\mathbf{S}$.

We now proceed the same way as we did in the case of UNFO (but using guarded ℓ-types instead of unary types). We introduce a fresh ℓ-ary relation symbol P_τ for each guarded ℓ-type τ, and we denote by $\mathbf{S}'$ the schema that extends $\mathbf{S}$ with these additional symbols. Diagrams, realizability, and implying a query are defined in the same way as before. The DDlog program is also constructed in essentially the same manner, except that the first rule of the program is replaced by the following:

$$\bigvee_{\substack{\tau \text{ a guarded } \ell\text{-type} \\ \text{with } R(\mathbf{x}) \in \tau}} P_\tau(\mathbf{x}) \leftarrow R(\mathbf{x}) \quad \text{for each relation } R \text{ of arity } \ell \geq 0.$$

We establish the correctness of the translation. That is, we show that, for every instance $\mathfrak{D}$ and elements $\mathbf{a} = (a_1, \ldots, a_n) \in \mathsf{adom}(\mathfrak{D})^n$, we have $\mathbf{a} \in \mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$ if and only if $\mathbf{a} \in q_\Pi(\mathfrak{D})$.

"if". Assume that $\mathbf{a} \notin \mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$. Then there is $(\mathsf{dom}, \mathfrak{D}') \in \mathsf{Mod}(\mathcal{O})$ with $\mathfrak{D} \subseteq \mathfrak{D}'$ such that $\mathbf{a} \notin q(\mathfrak{D}')$. For every fact $R(\mathbf{b})$ of $\mathfrak{D}$, let $\mu(\mathbf{b})$ be the unique guarded ℓ-type (with $\ell = |\mathbf{b}|$) realized at $\mathbf{b}$ in $\mathfrak{D}'$. Let $\mathfrak{D}''$ be the instance that consists of the atoms in $\mathfrak{D}$ and the atom $P_{\mu(\mathbf{b})}(\mathbf{b})$ for each fact $R(\mathbf{b})$ in $\mathfrak{D}$. It can be checked that $\mathfrak{D}''$ is a model of Π. Since $\mathsf{goal}(\mathbf{a}) \notin \mathfrak{D}''$, $\mathbf{a} \notin q_\Pi(\mathfrak{D})$.

"only if". Assume that $\mathbf{a} \notin q_\Pi(\mathfrak{D})$ and let $\mathfrak{D}'$ be a model of Π with $\mathfrak{D} \subseteq \mathfrak{D}'$ that does not contain $\mathsf{goal}(\mathbf{a})$. We say that a tuple $\mathbf{b}$ is "live" in $\mathfrak{D}$ if $\mathfrak{D}$ contains $R(\mathbf{b})$ for some relation symbol R. For each live tuple $\mathbf{b}$ of $\mathfrak{D}$, let $\mu(\mathbf{b})$ be the unique guarded ℓ-type (with $\ell = |\mathbf{b}|$) such that $P_{\mu(\mathbf{b})}(\mathbf{b}) \in \mathfrak{D}'$, and let $(\mathsf{dom}_\mathbf{b}, \mathfrak{D}_\mathbf{b})$ be a model of $\mathcal{O}$ in which $\mu(\mathbf{b})$ is realized at $\mathbf{b}$ (such a model must exist because otherwise the diagram $P_{\mu(\mathbf{b})}(\mathbf{x})$ would be non-realizable and Π would include a rule $\bot \leftarrow P_{\mu(\mathbf{b})}(\mathbf{x})$). We may assume that for distinct live tuples $\mathbf{b}$ and $\mathbf{c}$, $\mathsf{dom}_\mathbf{b}$ and $\mathsf{dom}_\mathbf{c}$ overlap only (possibly) on $\{\mathbf{b}\} \cap \{\mathbf{c}\}$. Let $(\mathsf{dom}'', \mathfrak{D}'')$ be obtained by first taking the union of $(\mathsf{dom}_\mathbf{b}, \mathfrak{D}_\mathbf{b})$ for all live tuples $\mathbf{b}$ of $\mathfrak{D}$, and then adding to it all facts of $\mathfrak{D}$. We show that

(i) $(\mathsf{dom}'', \mathfrak{D}'')$ is a model of $\mathcal{O}$ and
(ii) $\mathbf{a} \notin q(\mathfrak{D}'')$.

For all live tuples $\mathbf{d}$ of $\mathfrak{D}_\mathbf{b}$, let $\mu(\mathbf{d})$ be the unique guarded ℓ-type realized by $\mathbf{d}$ in $(\mathsf{dom}_\mathbf{b}, \mathfrak{D}_\mathbf{b})$, for all $d \in \mathsf{dom}_a$. Note that a tuple $\mathbf{d}$ may be live in $\mathfrak{D}_\mathbf{b}$ for several different choices of $\mathbf{b}$, but then the guarded ℓ-type realized by $\mathbf{d}$ in each such $(\mathsf{dom}_\mathbf{b}, \mathfrak{D}_\mathbf{b})$ is the same: otherwise, there must be some atom $R(\mathbf{y})$ that belongs to $\mu(\mathbf{b})$, but not to $\mu(\mathbf{b}')$, and then the diagram $P_{\mu(\mathbf{b}')}(\mathbf{x}) \wedge R(\mathbf{y})$ is non-realizable and thus ruled out by Π. Claim (i) is proved by establishing the following, by induction on the length of φ:

($*$) For all formulas $\varphi(\mathbf{x}) \in \mathsf{cl}^\ell_k(\mathcal{O}, q)$ and for each live ℓ-tuple $\mathbf{d}$ of $\mathfrak{D}''$, $(\mathsf{dom}'', \mathfrak{D}'') \models \varphi[\mathbf{d}]$ iff $\varphi \in \mu(\mathbf{d})$.

We omit the proofs of ($*$) and of (ii), as they proceed similarly to the proofs of the corresponding statements in the proofs of Theorems 3.3 and 3.14. □

B. PROOFS FOR SECTION 4

Theorem 4.3 *GMSNP and MMSNP$_2$ have the same expressive power.*

PROOF. For simplicity, we prove the result for sentences (no free variables) and without equality in the body of implications.

We start by proving that every MMSNP$_2$ sentence is equivalent to a GMSNP sentence. Assume $\Phi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ is a MMSNP$_2$ sentence. Introduce for each X_i a monadic SO variable X_i^1 and, for every $R \in \mathbf{S}$ of arity n, an n-ary SO variable X_i^R. Now replace in φ every $X_i(x)$ by $X_i^1(x)$ and every $X_i(R(\mathbf{x}))$ by $X_i^R(\mathbf{x})$. The resulting formula is a GMSNP sentence that is equivalent to Φ.

Conversely, assume we are given a GMSNP sentence $\Phi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$. It is straightforward to show that Φ is equivalent to a GMSNP sentence in which

— each $X_i(\mathbf{x})$ in the head of an implication is guarded by an input relation: for every $X_i(\mathbf{x})$ in the head of an implication ψ there exists an $R \in \mathbf{S}$ such that $R(\mathbf{y})$ is in the body of ψ and $\mathbf{x} \subseteq \mathbf{y}$. (If this is not the case, one can introduce additional conjuncts $R(\mathbf{y})$ in the body of implications).

— φ is closed under identifying FO variables: if ψ' is the result of identifying variables in an implication ψ of φ, then ψ is a conjunct of φ (modulo renaming of FO variables).
— the FO variables used in distinct implications of φ are disjoint.

It follows that we may also assume that distinct occurrences of SO variables X_i in φ determine distinct atoms $X_i(\mathbf{x}_i)$. From now we assume that Φ satisfies these conditions.

For the translation, we take for every atom $A = X_i(\mathbf{x})$ in the head of an implication ψ in φ, a fresh second-order domain and fact variable X_A. Moreover, we fix a guard $R_A(\mathbf{y}_A)$ with $R_A \in \mathbf{S}$ for A from the body of the (unique) implication in which A occurs. Consider now an implication ψ in φ of the form

$$R_1(\mathbf{x}_1) \wedge \cdots \wedge R_k(\mathbf{x}_k) \wedge X_{k+1}(\mathbf{x}_{k+1}) \wedge \cdots \wedge X_n(\mathbf{x}_n) \rightarrow X_{n+1}(\mathbf{x}_{n+1}) \vee \cdots \vee X_m(\mathbf{x}_m)$$

First replace all atoms $A_j = X_j(\mathbf{x}_j)$, $n+1 \leq j \leq m$, by $X_{A_j}(R_{A_j}(\mathbf{y}_{A_j}))$, where $R_{A_j}(\mathbf{y}_{A_j})$ is the guard for A_j selected above. Next consider every possible choice

$$A_{k+1} = X_{k+1}(\mathbf{z}_{k+1}), \ldots, A_n = X_n(\mathbf{z}_n)$$

of atoms in the heads of implications in φ such that the componentwise mappings $\rho_l : \mathbf{x}_l \rightarrow \mathbf{z}_l$, $k+1 \leq l \leq n$, are bijections between the sets of variables in $\mathbf{x}_l$ and $\mathbf{z}_l$ and replace every $X_l(\mathbf{x}_l)$, $k+1 \leq l \leq n$, by

$$X_{A_l}(R_{A_l}(\mathbf{y}'_l))$$

where $\mathbf{y}'_l$ is obtained from the guard $R_{A_l}(\mathbf{y}_{A_l})$ associated with A_l above by replacing each $\rho_l(x)$ by x and each FO variable that is not in the range of ρ_l by some fresh FO variable. Let ψ' be the conjunction over all implications derived from ψ in this manner, let φ' be the conjunction of all of the ψ', and let Φ' be the resulting MMSNP$_2$ sentence when existential quantification over non-monadic variables is replaced by existential quantification over all X_A such that A an atom in a head of an implication of φ. Note that Φ' contains all FO variables in Φ, but may also contain additional FO variables not in Φ.

We show that Φ and Φ' are equivalent. Assume first that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi'$. Take an assignment π for the second-order domain and fact variables of Φ' such that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models_\pi \forall x_1 \cdots \forall x_m \varphi'$. For every non-monadic second-order variable X of Φ, define $\pi(X)$ as the union of all

$$\{\rho(\mathbf{x}) \mid R_A(\rho(\mathbf{y}_A)) \in \pi(X_A),\ \rho \text{ injective variable assignment}\},$$

such that $A = X(\mathbf{x})$ appears in the head of some implication in φ and $R_A(\mathbf{y}_A)$ is the guard selected for A. We show that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models_\pi \Phi$. Assume for a contradiction that this is not the case. Take an implication ψ in φ of the form

$$R_1(\mathbf{x}_1) \wedge \cdots \wedge R_k(\mathbf{x}_k) \wedge X_{k+1}(\mathbf{x}_{k+1}) \wedge \cdots \wedge X_n(\mathbf{x}_n) \rightarrow X_{n+1}(\mathbf{x}_{n+1}) \vee \cdots \vee X_m(\mathbf{x}_m)$$

and let ρ be an assignment for the FO variables such that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \not\models_{\pi,\rho} \psi$. We may assume that ρ is injective. The following holds:

(1) for every $1 \leq i \leq k$, we have $R_i(\rho(\mathbf{x}_i)) \in \mathfrak{D}$.
(2) for every $k+1 \leq i \leq n$, there exists $A_i = X_i(\mathbf{z}_i)$ in the head of some implication of φ with $R_{A_i}(\mathbf{z}'_i)$ the guard selected for A_i, and an injective variable assignment ρ_i such that $R_{A_i}(\rho_i(\mathbf{z}'_i)) \in \pi(X_{A_i})$ and $\rho_i(\mathbf{z}_i) = \rho(\mathbf{x}_i) \in \pi(X_i)$.
(3) for no $n+1 \leq i \leq m$ does there exist $A_i = X_i(\mathbf{z}_i)$ in the head of some implication of φ with $R_{A_i}(\mathbf{z}'_i)$ the guard selected for A_i, and an injective variable assignment ρ' such that $R_{A_i}(\rho'(\mathbf{z}'_i)) \in \pi(X_{A_i})$ and $\rho'(\mathbf{z}_i) = \rho(\mathbf{x}_i) \in \pi(X_i)$.

Consider the following sequences of atoms

$$A_{k+1} = X_{k+1}(\mathbf{z}_{k+1}), \ldots, A_n = X_n(\mathbf{z}_n)$$
$$A_{n+1} = X_{n+1}(\mathbf{x}_{n+1}), \ldots, A_m = X_m(\mathbf{x}_m)$$

It follows from construction of Φ' that the formula φ' contains the implication

$$\begin{aligned}\zeta = \quad & R_1(\mathbf{x}_1) \wedge \cdots \wedge R_k(\mathbf{x}_k) \wedge \\ & X_{A_{k+1}}(R_{A_{k+1}}(\mathbf{y}'_{k+1})) \wedge \cdots \wedge X_{A_n}(R_{A_n}(\mathbf{y}'_n)) \\ \rightarrow\ & X_{A_{n+1}}(R_{A_n+1}(\mathbf{y}_{A_n+1})) \vee \cdots \vee X_{A_m}(R_{A_m}(\mathbf{y}_{A_m}))\end{aligned}$$

where the $\mathbf{y}'_i$ are defined in the same way as earlier. Let μ be a variable assignment for the FO variables in Φ' satisfying:

— $\mu(x) = \rho(x)$ if x is in the domain of ρ
— $\mu(u) = \rho_i(z)$ if u is the fresh variable introduced to replace $z \in \mathbf{z}'_i$

Note that such an assignment must exist since every variable in Φ' is in the domain of exactly one assignment among ρ and the ρ_i. It follows from the properties of μ and Points 1 and 2 above that the body of the implication ζ is satisfied under assignments π, μ. From Point 3, we can derive that none of the head atoms is satisfied under π, μ. It follows that the implication ζ is refuted, so $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \not\models \Phi'$, and we have the desired contradiction.

For the other direction, assume that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi$. Take an assignment π for the SO variables of Φ such that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models_\pi \forall x_1 \cdots \forall x_m\ \varphi$. Now define, for $A = X(\mathbf{x})$ in the head of an implication of φ with selected guard $R_A(\mathbf{y}_A)$:

$$\pi(X_A) = \{R_A(\rho(\mathbf{y}_A)) \in \mathfrak{D} \mid \rho(\mathbf{x}) \in \pi(X), \rho \text{ variable assignment}\}$$

It can be verified that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi'$. $\square$

C. PROOFS FOR SECTION 5

Our first aim is to prove Proposition 5.2. To this end we extend forbidden pattern problems to n-ary marked instances. Recall from Section 4.2 that an n-ary *marked instance* is of the form $(\mathfrak{D}, \mathbf{a})$, where $\mathfrak{D}$ is an instance and $\mathbf{a} = (a_1, \ldots, a_n) \in \mathsf{adom}(\mathfrak{D})^n$, and that we write $(\mathfrak{D}, \mathbf{a}) \rightarrow (\mathfrak{D}', \mathbf{a}')$ if there is a homomorphism from $\mathfrak{D}$ to $\mathfrak{D}'$ that maps each a_i to a'_i. By the *evaluation problem* of an MMSNP formula Φ with free variables $y_1, \ldots, y_m$ we mean the problem, given an m-ary marked instance $(\mathfrak{D}, \mathbf{d})$, to decide whether it satisfies the formula, i.e. whether $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi[\mathbf{d}]$. n-ary forbidden pattern problems are defined in the same way as ordinary forbidden pattern problems (refer to page 10 in Section 3), but replacing every occurrence of the word *instance* by *n-ary marked instance*. We denote by FPP$_n$ the class of n-ary forbidden patterns problems thus obtained (and we will continue to use FPP to refer to ordinary, zero-ary forbidden pattern problems).

It was shown in [Madelaine and Stewart 2007] that MMSNP sentences and FPP have the same expressive power on unmarked instances. This result can be straightforwardly extended to marked instances. For the sake of completeness, we provide a proof.

PROPOSITION C.1. *MMSNP formulas with free variables $x_1, \ldots, x_n$ and FPP$_n$ have the same expressive power (over n-ary marked instances).*

PROOF. Consider a forbidden patterns problem given by a collection $\mathcal{F}$ of $\mathcal{C}$-colored n-ary marked S-instances, where $\mathcal{C} = \{C_1, \ldots, C_k\}$. For every instance $(\mathfrak{F}, \mathbf{f}) \in \mathcal{F}$, we create a formula $\psi_{\mathfrak{F},\mathbf{f}}$ as follows:

— Take the conjunction of all facts in $\mathfrak{F}$.
— For every element $d \notin \mathbf{f}$, replace all occurrences of d by some fresh variable.
— For every element $f_i \in \mathbf{f}$ such that $f_j \neq f_i$ for every $j < i$, replace f_i by y_i.
— For every $1 \leq i \leq n$ such that $f_i = f_j$ for $j < i$, add the equality $y_i = y_j$.

The desired MMSNP formula takes the form

$$\begin{aligned}\Phi = \exists C_1, \ldots, C_k \forall x_1, \ldots, x_m \; & (\mathsf{adom}(x) \to \textstyle\bigvee_{1 \leq i \leq k} C_i(x)) \\ & \wedge \textstyle\bigwedge_{1 \leq i < j \leq k} (C_i(x) \wedge C_j(x) \to \bot) \\ & \wedge \textstyle\bigwedge_{(\mathfrak{F},\mathbf{f}) \in \mathcal{F}} (\psi_{\mathcal{F},\mathbf{f}} \to \bot)\end{aligned}$$

where $x_1, \ldots, x_m$ are all variables which appear in some conjunct, excepting the variables $y_1, \ldots, y_n$. It is immediate from the construction that for all n-ary marked $\mathbf{S}$-instances $(\mathfrak{D}, \mathbf{d})$, we have $(\mathfrak{D}, \mathbf{d}) \in \mathsf{Forb}(\mathcal{F})$ iff $(\mathsf{adom}(\mathfrak{D}, \mathbf{d}), \mathfrak{D}) \models \Phi[\mathbf{d}]$. We observe that the translation is polynomial-time computable.

For the converse direction, we give an exponential translation. Let Φ be any MMSNP formula with n free variables. By Proposition 4.1, there is an MDDlog program Π such that for every n-ary marked instance $(\mathfrak{D}, \mathbf{d})$, $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi[\mathbf{d}]$ iff $\mathbf{d} \notin q_\Pi(\mathfrak{D})$. Let $X_1, \ldots, X_k$ be the IDBs of the program Π, and let $\mathcal{C}$ consist of a distinct color $C_\mathcal{X}$ for every subset $\mathcal{X} \subseteq \{X_1, \ldots, X_k\}$. Observe that there is a natural one-to-one correspondence between $\mathcal{C}$-colored $\mathbf{S}$-instances $\mathfrak{D}$ and $\mathbf{S} \cup \{X_1, \ldots, X_k\}$-instances $\mathfrak{D}'$. More precisely, the correspondence is given by $C_\mathcal{X}(a) \in \mathfrak{D}$ iff $\mathcal{X} = \{X_i \mid 1 \leq i \leq k,\ X_i(a) \in \mathfrak{D}'\}$.

We wish to construct a set $\mathcal{F}$ of $\mathcal{C}$-colored n-ary marked $\mathbf{S}$-instances such that $(\mathfrak{D}, \mathbf{d}) \in \mathsf{Forb}(\mathcal{F})$ iff $\mathbf{d} \notin q_\Pi(\mathfrak{D})$. The latter holds just in the case that there is some $\mathfrak{D}' \in \mathsf{Mod}(\Pi)$ such that $\mathfrak{D} \subseteq \mathfrak{D}'$ and $\mathsf{goal}(\mathbf{d}) \notin \mathfrak{D}'$. This can be captured using two types of forbidden patterns, the first corresponding to satisfaction of a goal rule body, and the second corresponding to the violation of a non-goal rule.

To formally define the forbidden patterns in $\mathcal{F}$, we associate to every rule ρ of Π the $\mathbf{S}$-instance $\mathfrak{A}_\rho$ whose domain is the set of variables in ρ and whose facts are the $\mathbf{S}$-facts occurring in ρ. If $\rho = \mathbf{goal}(\mathbf{x}) \leftarrow \alpha$ is a goal rule in Π, then we let $\mathcal{F}_\rho$ consist of all $\mathcal{C}$-colored n-ary marked instances $(\mathfrak{F}, \mathbf{x})$ where $\mathfrak{F}$ is obtained by adding to $\mathfrak{A}_\rho$ a single fact $C_\mathcal{X}(x)$ for every variable x in ρ in such a way that if $X_i(x)$ belongs to the body of ρ, then $X_i \in \mathcal{X}$. If ρ is a non-goal rule, then we are interested in the $\mathcal{C}$-colorings of $\mathfrak{A}_\rho$ that correspond (via the above correspondence) to an $\mathbf{S} \cup \{X_1, \ldots, X_n\}$-instance violating ρ. Specifically, we consider the set $\mathcal{G}_\rho$ of all $\mathcal{C}$-colored $\mathbf{S}$-instances $\mathfrak{G}$ that have the same $\mathbf{S}$-facts as $\mathfrak{A}_\rho$, and such that (i) for each conjunct $X_i(x_j)$ in the body of ρ, we have that $C_\mathcal{X}(x_i) \in \mathfrak{G}$ for some $\mathcal{X}$ with $X_i \in \mathcal{X}$, and (ii) for each disjunct $X_i(x_j)$ in the head, we have $C_\mathcal{X}(x_i) \in \mathfrak{G}$ for some $\mathcal{X}$ with $X_i \notin \mathcal{X}$. The instances in $\mathcal{G}_\rho$ capture the violation of ρ, but we actually require n-ary marked instances. This mismatch is easily resolved by adding to the instances in $\mathcal{G}_\rho$ tuples of fresh marked elements as well as new facts containing these elements. In this way, we can obtain a set $\mathcal{F}_\rho$ of $\mathcal{C}$-colored n-ary marked instances with the property that for every n-ary marked $\mathbf{S}$-instance $(\mathfrak{D}, \mathbf{d})$, we have $\mathfrak{G} \to \mathfrak{D}$ for some $\mathfrak{G} \in \mathcal{G}_\rho$ iff $(\mathfrak{F}, \mathbf{f}) \to (\mathfrak{D}, \mathbf{d})$ for some $(\mathfrak{F}, \mathbf{f}) \in \mathcal{F}_\rho$. Finally, we define $\mathcal{F}$ as the union of the sets $\mathcal{F}_\rho$ for all rules ρ of Π. It is easily verified that $\mathsf{Forb}(\mathcal{F})$ has the desired properties. □

We define several operations that allow us to transform n-ary marked instances over $\mathbf{S}$ into (unmarked) instances over $\mathbf{S}_P$ and vice versa.

If an unmarked $\mathbf{S}_P$-instance $\mathfrak{D}$ is such that for every $1 \leq i \leq n$ there is an element a such that $P_i(a) \in \mathfrak{D}$, then we can associate to $\mathfrak{D}$ an n-ary marked $\mathbf{S}$-instance $(\mathfrak{D}^c, \mathbf{c})$, called the *collapse* of $\mathfrak{D}$, by factorizing through the $P_i^\mathfrak{D}$. Specifically, let $\sim$ be the smallest equivalence relation such that whenever $P_i(d), P_i(d') \in \mathfrak{D}$ for some $i \leq n$, then $d \sim d'$. Then the domain of $\mathfrak{D}^c$ is $\{[d] \mid d \in \mathsf{adom}(\mathfrak{D})\}$, where $[d]$ denotes the equivalence class of d w.r.t. $\sim$. For convenience, when $[d] = \{d\}$, we will use d in place of $[d]$. For $R \in \mathbf{S}$, we set $R([d_1], \ldots, [d_n]) \in \mathfrak{D}^c$ if and only if there exist $d'_1 \in [d_1], \ldots, d'_n \in [d_n]$ and such that $R(d'_1, \ldots, d'_n) \in \mathfrak{D}$. Note that the mapping $g : d \mapsto [d]$ defines an $\mathbf{S}$-homomorphism from $\mathfrak{D}$ to $\mathfrak{D}^c$, such that for each $i \leq n$, all elements d with $P_i(d) \in \mathfrak{D}$ have the same g-image, which we denote by c_i. We call g the *natural homomorphism*. Finally, collapse of $\mathfrak{D}$ is the n-ary marked instance $(\mathfrak{D}^c, \mathbf{c})$ where $\mathbf{c} = c_1, \ldots, c_n$.

For an n-ary marked $\mathbf{S}$-instance $(\mathfrak{A}, \mathbf{a})$ with $\mathbf{a} = a_1, \ldots, a_n$, we define $\widehat{(\mathfrak{A}, \mathbf{a})}$ to be the unmarked $\mathbf{S}_P$-instance $\mathfrak{A}'$ such that, for $R \in \mathbf{S}$ and for all elements $b_1, \ldots, b_m$, $R(b_1, \ldots, a_m) \in \mathfrak{A}'$ if and only if $R(b_1, \ldots, b_m) \in \mathfrak{A}$ and $\mathfrak{A}'$ contains the additional facts $P_i(a_i)$ for all $i \leq n$.

It is shown in [Alexe et al. 2011] that, with every n-ary marked $\mathbf{S}$-instance $(\mathfrak{B}, \mathbf{b})$, one can associate a finite set $(\mathfrak{B}, \mathbf{b})^{ac}$ of unmarked $\mathbf{S}_P$-instances, collectively called the *anti-collapse* of $(\mathfrak{B}, \mathbf{b})$ such that the following two properties hold:[10]

(1) for all unmarked $\mathbf{S}_P$-instances $\mathfrak{A}$:
$(\mathfrak{B}, \mathbf{b}) \to (\mathfrak{A}^c, \mathbf{c})$ (and the collapse $(\mathfrak{A}^c, \mathbf{c})$ is well-defined) if and only if there exists $\mathfrak{B}' \in (\mathfrak{B}, \mathbf{b})^{ac}$ such that $\mathfrak{B}' \to \mathfrak{A}$.

(2) for all n-ary marked $\mathbf{S}$-instances $(\mathfrak{A}, \mathbf{a})$:
$(\mathfrak{B}, \mathbf{b}) \to (\mathfrak{A}, \mathbf{a})$ iff there exists $\mathfrak{B}' \in (\mathfrak{B}, \mathbf{b})^{ac}$ such that $\mathfrak{B}' \to \widehat{(\mathfrak{A}, \mathbf{a})}$.

To employ the anti-collapse $(\mathfrak{B}, \mathbf{b})^{ac}$ for the reduction of FPP$_n$ to FPP, we require some properties from the construction of $(\mathfrak{B}, \mathbf{b})^{ac}$ (cf. pages 43-45 of [Alexe et al. 2011]). The active domain $\mathsf{adom}(\mathfrak{B}')$ of each $\mathfrak{B}' \in (\mathfrak{B}, \mathbf{b})^{ac}$ consists of $\mathsf{adom}(\mathfrak{B}) \setminus \{\mathbf{b}\}$ (the *unnamed* individuals in $\mathfrak{B}$) together with a union $\bigcup_{1 \leq i \leq n} D_i$ of fresh non-empty (but possibly not mutually disjoint) sets $D_1, \ldots, D_n$ with $D_i = \{a \mid P_i(a) \in \mathfrak{B}'\}$. Moreover, the above points (1) and (2) hold in the following more detailed way:

(1a). if $h : (\mathfrak{B}, \mathbf{b}) \to (\mathfrak{A}^c, \mathbf{c})$ (and the collapse $(\mathfrak{A}^c, \mathbf{c})$ is well-defined), and $g : \mathfrak{A} \to \mathfrak{A}^c$ is the natural homomorphism, then $h' : \mathfrak{B}' \to \mathfrak{A}$ can be chosen in such a way that $h'(d) \in g^{-1}(h(d))$ for all unnamed individuals d in $\mathfrak{B}$ and $h'(d) \in g^{-1}(c_i)$ for all $d \in D_i$.

(1b). if $h : \mathfrak{B}' \to \mathfrak{A}$, then $h' : (\mathfrak{B}, \mathbf{b}) \to (\mathfrak{A}^c, \mathbf{c})$ can be defined such that $h'(b_i) = c_i$ and $h'(d) = g(h(d))$ if d is unnamed.

(2b). if $h : \mathfrak{B}' \to \widehat{(\mathfrak{A}, \mathbf{a})}$, then $h' : (\mathfrak{B}, \mathbf{b}) \to (\mathfrak{A}, \mathbf{a})$ can be constructed in such a way that $h'(d) = h(d)$ for all unnamed d.

In what follows, we will be interested in colorings of $\mathbf{S}_P$-instances which respect the intuitive meaning of the relations P_i. An $\mathcal{C}$-coloring $\mathfrak{B}[\mathcal{C}]$ of a $\mathbf{S}_P$-instance $\mathfrak{B}$ is said to be a *uniform $\mathcal{C}$-coloring* of $\mathfrak{B}$ if for every $1 \leq i \leq n$, $P_i(d), P_i(d') \in \mathfrak{B}$ implies that d and d' have the same color in $\mathfrak{B}[\mathcal{C}]$. Given a set $\mathcal{G}$ of $\mathcal{C}$-colored $\mathbf{S}_P$-instances, we define $\mathsf{Forb}^{un}(\mathcal{G})$ as the set of $\mathbf{S}_P$-instances $\mathfrak{A}$ such that there exists a uniform $\mathcal{C}$-coloring $\mathfrak{A}[\mathcal{C}]$ of $\mathfrak{A}$ such that there exists no $\mathfrak{G} \in \mathcal{G}$ with $\mathfrak{G} \to \mathfrak{A}[\mathcal{C}]$.

We are now ready to present the reduction from FPP$_n$ to FPP. Suppose that we are given a FPP$_n$ problem defined by the set $\mathcal{F}$ of $\mathcal{C}$-colored n-ary marked instances (where $\mathcal{C} = \{T_1, \ldots, T_k\}$). We construct a set $\mathcal{G}$ which contains all uniform $\mathcal{C}$-colored (unmarked) $\mathbf{S}_P$-instances $\mathfrak{G}$ for which the following condition holds:

[10]Strictly speaking, the results in [Alexe et al. 2011] pertain to finite relational structures with constant symbols, instead of marked instances. Note that, in finite relational structures with constant symbols, the constant symbols may be interpreted as elements outside the active domain. The difference, however, is not essential.

— There exists $(\mathfrak{F}, \mathbf{f}) \in \mathcal{F}$ and a member $\mathfrak{F}'$ of the anti-collapse of the S-reduct of $(\mathfrak{F}, \mathbf{f})$ such that $\mathfrak{G}$ is the $\mathcal{C}$-coloring of $\mathfrak{F}'$ defined as follows:

(†) $T_j(d) \in \mathfrak{G}$ iff d is unnamed in $\mathfrak{F}$ and $T_j(d) \in \mathfrak{F}$ or there exists $1 \leq i \leq n$ such that $d \in D_i$ and $T_j(f_i) \in \mathfrak{F}$.

(Note that if (†) yields $T_j(a), T_{j'}(a) \in \mathfrak{G}$ for some element a and for some $j \neq j'$, then the result is discarded as it is not a valid $\mathcal{C}$-coloring).

It is easy to see that this construction guarantees that for each $\mathfrak{G} \in \mathcal{G}$ and for each $1 \leq i \leq n$, there is an element a such that $P_i(a) \in \mathfrak{G}$.

We let $\mathcal{G}_u = \mathcal{G} \cup \mathcal{U}$, where $\mathcal{U}$ is the set of all $\mathbf{S}_P \cup \mathcal{C}$-instances of the form $\{P_i(d), P_i(e), T_j(d), T_\ell(e)\}$ with $1 \leq i \leq n$ and $1 \leq j < \ell \leq k$. Intuitively, the instances in $\mathcal{U}$ represent additional patterns that should be forbidden in order for an $\mathbf{S}_P \cup \mathcal{C}$-instance to be a valid uniform coloring. Notice that $\mathsf{Forb}^{un}(\mathcal{G}) = \mathsf{Forb}(\mathcal{G}_u)$.

LEMMA C.2. *The FPP$_n$ problem* $\mathsf{Forb}(\mathcal{F})$ *is polynomial-time equivalent to the FPP problem* $\mathsf{Forb}(\mathcal{G}_u)$. *Specifically:*

— *For all $\mathbf{S}_P$-instances $\mathfrak{A}$, $\mathfrak{A} \in \mathsf{Forb}(\mathcal{G}_u)$ iff either the collapse $(\mathfrak{A}^c, \mathbf{c})$ is undefined or $(\mathfrak{A}^c, \mathbf{c}) \in \mathsf{Forb}(\mathcal{F})$;*

— *For all n-ary marked* **S**-*instances $(\mathfrak{A}, \mathbf{a})$, $(\mathfrak{A}, \mathbf{a}) \in \mathsf{Forb}(\mathcal{F})$ iff $\widehat{(\mathfrak{A}, \mathbf{a})} \in \mathsf{Forb}(\mathcal{G}_u)$.*

PROOF. First let $\mathfrak{A}$ be an $\mathbf{S}_P$-instance such that $\mathfrak{A} \in \mathsf{Forb}(\mathcal{G}_u)$. Since $\mathsf{Forb}(\mathcal{G}_u) = \mathsf{Forb}^{un}(\mathcal{G})$, we have $\mathfrak{A} \in \mathsf{Forb}^{un}(\mathcal{G})$, and so there exists a uniform $\mathcal{C}$-colored expansion $\mathfrak{A}[\mathcal{C}]$ of $\mathfrak{A}$ such that there exists no $\mathfrak{G} \in \mathcal{G}$ with $\mathfrak{G} \to \mathfrak{A}[\mathcal{C}]$. Assume the collapse $(\mathfrak{A}^c, \mathbf{c})$ is defined (i.e., for each $1 \leq i \leq n$ there is an element a such that $P_i(a) \in \mathfrak{A}$). We want to show $(\mathfrak{A}^c, \mathbf{c}) \in \mathsf{Forb}(\mathcal{F})$. By uniformity of $\mathfrak{A}[\mathcal{C}]$, we obtain a $\mathcal{C}$-colored marked S-instance $(\mathfrak{A}^c[\mathcal{C}], \mathbf{c})$ extending $(\mathfrak{A}^c, \mathbf{c})$ by setting $T_j(d) \in \mathfrak{A}^c[\mathcal{C}]$ iff d is unnamed and $T_j(d) \in \mathfrak{A}[\mathcal{C}]$ or $d = c_i$ for some $i \leq n$ and $\{a \mid P_i(a) \in \mathfrak{A}[\mathcal{C}]\} \subseteq \{a \mid T_j(a) \in \mathfrak{A}[\mathcal{C}]\}$. Assume for a contradiction that $h : (\mathfrak{F}, \mathbf{f}) \to (\mathfrak{A}^c[\mathcal{C}], \mathbf{c})$ for $(\mathfrak{F}, \mathbf{f}) \in \mathcal{F}$. Then h is a homomorphism from the S-reduct $\mathfrak{F}^r$ of $\mathfrak{F}$ to the S-reduct $\mathfrak{A}^c$ of $\mathfrak{A}^c[\mathcal{C}]$. By (1a), we find $\mathfrak{F}' \in (\mathfrak{F}^r, \mathbf{f})^{ac}$ and $h' : \mathfrak{F}' \to \mathfrak{A}$ such that $h'(d) \in g^{-1}(h(d))$ for all unnamed individuals d in $\mathfrak{F}^r$ and $h'(d) \in g^{-1}(c_i)$ for all $d \in D_i$. Let $\mathfrak{F}'[\mathcal{C}]$ be the $\mathcal{C}$-coloring of $\mathfrak{F}'$ defined with (†). To see that $\mathfrak{F}'[\mathcal{C}]$ is well-defined, note that $d \in D_i \cap D_j$ implies that $P_i(d), P_j(d) \in \mathfrak{F}'$, which yields $P_i(h'(d)), P_j(h'(d)) \in \mathfrak{A}$, hence $c_i = c_j$. It follows that c_i and c_j have the same color in $\mathfrak{A}^c[\mathcal{C}]$, and thus also in $\mathfrak{F}$, which ensures that each element in $\mathfrak{F}'$ is assigned a unique color by (†). Now to obtain the desired contradiction, we show that h' is a $\mathbf{S}_P \cup \mathcal{C}$-homomorphism from $\mathfrak{F}'[\mathcal{C}]$ to $\mathfrak{A}[\mathcal{C}]$. Let $d \in \mathsf{adom}(\mathfrak{F}')$ and $T_j(d) \in \mathfrak{F}'[\mathcal{C}]$. If d is unnamed in $\mathfrak{F}$, then $T_j(d) \in \mathfrak{F}'[\mathcal{C}]$ implies that $T_j(d) \in \mathfrak{F}$. Hence $T_j(h(d)) \in \mathfrak{A}^c[\mathcal{C}]$ and $h'(d) \in g^{-1}(h(d)) \subseteq \{a \mid T_j(a) \in \mathfrak{A}[\mathcal{C}]\}$. If $d \in D_i$, then $T_j(d) \in \mathfrak{F}'[\mathcal{C}]$ implies $T_j(c_i) \in \mathfrak{F}$, hence $T_j(c_i) \in \mathfrak{A}^c[\mathcal{C}]$ and $\{a \mid P_i(a) \in \mathfrak{A}[\mathcal{C}]\} \subseteq \{a \mid T_j(a) \in \mathfrak{A}[\mathcal{C}]\}$. From $h'(d) \in g^{-1}(c_i)$, we know that there exists a sequence $A_{\ell_1}, \ldots, A_{\ell_p}$ of relation symbols from $\{P_1, \ldots, P_n\}$ such that $A_{\ell_1}(h'(d)) \in \mathfrak{A}[\mathcal{C}]$, $A_{\ell_p} = P_i$, and for each $1 \leq k \leq \ell_p$, there is an element a such that $A_{\ell_k}(a), A_{\ell_{k+1}}(a) \in \mathfrak{A}[\mathcal{C}]$. By uniformity of $\mathfrak{A}[\mathcal{C}]$ and $\{a \mid P_i(a) \in \mathfrak{A}[\mathcal{C}]\} \subseteq \{a \mid T_j(a) \in \mathfrak{A}[\mathcal{C}]\}$, we obtain $\{a \mid A_{\ell_1}(a) \in \mathfrak{A}[\mathcal{C}]\} \subseteq \{a \mid T_j(a) \in \mathfrak{A}[\mathcal{C}]\}$, hence $T_j(h'(d)) \in \mathfrak{A}[\mathcal{C}]$.

Conversely, if the collapse $(\mathfrak{A}^c, \mathbf{c})$ is undefined, then $\mathfrak{A} \in \mathsf{Forb}(\mathcal{G}_u)$ because, for all $\mathfrak{G} \in \mathcal{G}$ and $1 \leq i \leq n$, we have that $P_i(a) \in \mathfrak{G}$ for some element a, and hence any uniform $\mathcal{C}$-coloring of $\mathfrak{A}$ will avoid $\mathfrak{G}_u$. Assume now that $(\mathfrak{A}^c, \mathbf{c}) \in \mathsf{Forb}(\mathcal{F})$. There exists a $\mathcal{C}$-colored expansion $\mathfrak{A}^c[\mathcal{C}]$ of $\mathfrak{A}^c$ such that there exists no $(\mathfrak{F}, \mathbf{f}) \in \mathcal{F}$ with $(\mathfrak{F}, \mathbf{f}) \to (\mathfrak{A}^c[\mathcal{C}], \mathbf{c})$. We define a (uniform) $\mathcal{C}$-colored expansion $\mathfrak{A}[\mathcal{C}]$ of $\mathfrak{A}$ in the obvious way: let $g : \mathfrak{A} \to \mathfrak{A}^c$ be the natural homomorphism and set $T_j(a) \in \mathfrak{A}[\mathcal{C}]$ iff $T_j(g(a)) \in \mathfrak{A}^c[\mathcal{C}]$, for $1 \leq j \leq k$. Assume for a contradiction that $\mathfrak{G} \to \mathfrak{A}[\mathcal{C}]$ for $\mathfrak{G} \in \mathcal{G}$. Then $\mathfrak{G}$ is obtained from some $(\mathfrak{F}, \mathbf{f}) \in \mathcal{F}$ and some member $\mathfrak{F}'$ of the anti-collapse of the S-reduct $(\mathfrak{F}^r, \mathbf{f})$

of $(\mathfrak{F}, \mathbf{f})$ as described in (†). Assume $h : \mathfrak{G} \to \mathfrak{A}[\mathcal{C}]$. Then $h : \mathfrak{F}' \to \mathfrak{A}$ and so, by (1b) there exists $h' : (\mathfrak{F}^r, \mathbf{f}) \to (\mathfrak{A}^c, \mathbf{c})$ that can be defined such that $h'(f_i) = c_i$ and such that $h'(d) = g(h(d))$ if d is not named. We derive a contradiction by showing that h' a homomorphism from $(\mathfrak{F}, \mathbf{f})$ to $\mathfrak{A}^c[\mathcal{C}]$. First suppose that $T_j(d) \in \mathfrak{F}$, and d is unnamed in $(\mathfrak{F}, \mathbf{f})$. Then $T_j(d) \in \mathfrak{G}$, hence $h(d) \in T_j^{\mathfrak{A}[\mathcal{C}]}$. It follows from the definition of $T_j^{\mathfrak{A}[\mathcal{C}]}$ and $h'(d) = g(h(d))$ that $T_j(h'(d)) \in \mathfrak{A}^c[\mathcal{C}]$. Next consider the case where $T_j(f_i) \in \mathfrak{F}$. Then there must exist e such that $T_j(e) \in \mathfrak{G}$ and $P_i(e) \in \mathfrak{G}$. It follows that $T_j(h(e)) \in \mathfrak{A}[\mathcal{C}]$ and $P_i(h(e)) \in \mathfrak{A}[\mathcal{C}]$. The definition of the relation $T_j^{\mathfrak{A}[\mathcal{C}]}$ together with $g(h(e)) = c_i$ yields $h'(f_i) = c_i$ and $T_j(c_i) \in \mathfrak{A}^c[\mathcal{C}]$.

The second statement follows easily from the first, since every n-ary marked $\mathbf{S}$-instance $(\mathfrak{A}, \mathbf{a})$ is (up to isomorphism) equal to the collapse of $\widehat{(\mathfrak{A}, \mathbf{a})}$. □

By combining Proposition C.1 and Lemma C.2 (whose proofs all involve effective translations), we obtain Proposition 5.2.

To show Proposition 5.5 it is sufficient to prove the following reduction.

LEMMA C.3. *The containment problem for FPP$_n$ is polynomially reducible to the containment problem for FPP.*

PROOF. Consider FFP$_n$ problems $\mathsf{Forb}(\mathcal{F}_1)$ and $\mathsf{Forb}(\mathcal{F}_2)$, both over schema S. Let $\mathcal{G}_{u,1}$ and $\mathcal{G}_{u,2}$ be the corresponding FPPs over schema $\mathbf{S}_P$, which satisfy statements in Lemma C.2. We claim that $\mathsf{Forb}(\mathcal{F}_1) \subseteq \mathsf{Forb}(\mathcal{F}_2)$ iff $\mathsf{Forb}(\mathcal{G}_{u,1}) \subseteq \mathsf{Forb}(\mathcal{G}_{u,2})$.

For the first direction, suppose that $\mathsf{Forb}(\mathcal{F}_1) \subseteq \mathsf{Forb}(\mathcal{F}_2)$. Let $\mathfrak{A}$ be an $\mathbf{S}_P$-instance such that $\mathfrak{A} \in \mathsf{Forb}(\mathcal{G}_{u,1})$. If $\mathfrak{A}^c$ is undefined, then we immediately obtain $\mathfrak{A} \in \mathsf{Forb}(\mathcal{G}_{u,2})$. Otherwise, we have $\mathfrak{A}^c \in \mathsf{Forb}(\mathcal{F}_1)$, and hence $\mathfrak{A}^c \in \mathsf{Forb}(\mathcal{F}_2)$ and $\mathfrak{A} \in \mathsf{Forb}(\mathcal{G}_{u,2})$.

For the second direction, suppose that $\mathsf{Forb}(\mathcal{G}_{u,1}) \subseteq \mathsf{Forb}(\mathcal{G}_{u,2})$, and let $(\mathfrak{B}, \mathbf{b})$ be an n-ary marked S-instance such that $(\mathfrak{B}, \mathbf{b}) \in \mathsf{Forb}(\mathcal{F}_1)$. Then applying the previous lemma, we have $\widehat{(\mathfrak{B}, \mathbf{b})} \in \mathsf{Forb}(\mathcal{G}_{u,1})$, hence $\widehat{(\mathfrak{B}, \mathbf{b})} \in \mathsf{Forb}(\mathcal{G}_{u,2})$. Again applying the lemma, we obtain $(\mathfrak{B}, \mathbf{b}) \in \mathsf{Forb}(\mathcal{F}_2)$. □

We prove Theorems 5.8 and 5.17 that claim the undecidability of query containment, FO-rewritability and datalog-rewritability of queries in $(\mathcal{ALCF}, \text{AQ})$ and queries in $(\mathcal{ALCF}, \text{BAQ})$. In [Bienvenu et al. 2012; Lutz and Wolter 2012], alternative definitions of query containment and FO-rewritability are employed which consider only instances that are consistent with the ontologies involved. We say that $(\mathbf{S}, \mathcal{O}_1, q_1)$ *is contained in* $(\mathbf{S}, \mathcal{O}_2, q_2)$ *w.r.t. consistent instances* if $q_{(\mathbf{S},\mathcal{O}_1,q_1)}(\mathfrak{D}) \subseteq q_{(\mathbf{S},\mathcal{O}_2,q_2)}(\mathfrak{D})$ for all S-instance $\mathfrak{D}$ such that $\mathfrak{D}$ is consistent with $\mathcal{O}_1$. Similarly, a query $(\mathbf{S}, \mathcal{O}, q)$ is ***FO-rewritable*** w.r.t *consistent instances* if there exists an FO-query q' such that $q'(\mathfrak{D}) = q_{(\mathbf{S},\mathcal{O},q)}(\mathfrak{D})$ for all S-instance $\mathfrak{D}$ that are consistent with $\mathcal{O}$. Undecidability of query containment w.r.t. consistent instances and of FO-rewritability w.r.t. consistent instances were proven respectively in [Bienvenu et al. 2012] and [Lutz and Wolter 2012]. Here we show how the proofs can be modified to work for query containment, FO-rewritability, and datalog rewritability as defined in this paper.

THEOREM C.4. *Query containment, FO-rewritability, and datalog-rewritability are all undecidable for queries in $(\mathcal{ALCF}, \textbf{AQ})$ and queries in $(\mathcal{ALCF}, \textbf{BAQ})$.*

PROOF. The proof is by reduction of the following finite rectangle tiling problem. An instance of the *finite rectangle tiling problem* is given by a triple $\mathfrak{P} = (\mathbb{T}, \mathbb{H}, \mathbb{V})$ with

— $\mathbb{T} = \{T_1, \ldots, T_p\}$ a non-empty, finite set of *tile types* including an *initial tile* T_{init} to be placed on the lower left corner, a *final tile* T_{final} to be placed on the upper right

corner, and sets $\mathfrak{U} \subseteq \mathbb{T}$ and $\mathfrak{R} \subseteq \mathbb{T}$ of tile types to be placed on the upper and right borders respectively, satisfying $\mathfrak{U} \cap \mathfrak{R} = \{T_{\mathsf{final}}\}$;
— $\mathbb{H} \subseteq \mathbb{T} \times \mathbb{T}$ a *horizontal matching relation*; and
— $\mathbb{V} \subseteq \mathbb{T} \times \mathbb{T}$ a *vertical matching relation*.

A *tiling* for $(\mathbb{T}, \mathbb{H}, \mathbb{V})$ is a map $f : \{0, \dots, n\} \times \{0, \dots, m\} \to \mathbb{T}$ such that $n, m \geq 0$,

— $f(0,0) = T_{\mathsf{init}}$,
— $f(n,m) = T_{\mathsf{final}}$,
— $f(n,j) \in \mathfrak{R}$ for all $0 \leq j \leq m$;
— $f(j,i) \notin \mathfrak{R}$ for all $j < n$ and $0 \leq i \leq m$;
— $f(i,m) \in \mathfrak{U}$ for all $0 \leq i \leq n$;
— $f(i,j) \notin \mathfrak{U}$ for all $0 \leq i \leq n$ and $1 \leq j < m$.
— $(f(i,j), f(i+1,j)) \in \mathbb{H}$ for all $0 \leq i < n$, and
— $(f(i,j), f(i,j+1)) \in \mathbb{V}$ for all $0 \leq i < m$.

Thus, we can assume that $\mathbb{H}$, $\mathbb{V}$, $\mathfrak{U}$, and $\mathfrak{R}$ are such that:

— if $(T_i, T_j) \in \mathbb{H}$, then $T_i \in \mathfrak{U}$ if and only if $T_j \in \mathfrak{U}$;
— if $T_i \in \mathfrak{U}$, then there exists no T_j with $(T_i, T_j) \in \mathbb{V}$ or $(T_j, T_i) \in \mathbb{V}$;
— if $(T_i, T_j) \in \mathbb{V}$, then $T_i \in \mathfrak{R}$ if and only if $T_j \in \mathfrak{R}$;
— if $T_i \in \mathfrak{R}$, then there exists no T_j with $(T_i, T_j) \in \mathbb{H}$ or $(T_j, T_i) \in \mathbb{H}$.

It is undecidable whether an instance $\mathfrak{P}$ of the finite rectangle tiling problem has a tiling.

Fix a particular problem instance $\mathfrak{P} = (\mathbb{T}, \mathbb{H}, \mathbb{V})$. For the data schema, we use $\mathbf{S} = \{T_1, \dots, T_p, H, V, H^-, V^-\}$, where $T_1, \dots, T_p$ are treated as concept names, and H, V, H^-, and V^- are role names. We use H and V to specify horizontal and vertical adjacency of points in the rectangle, and the role names H^- and V^- to *simulate* the inverses of H and V (note that since H^- and V^- are regular role names, they need not be interpreted as the inverses of H and V). We construct an $\mathcal{ALCF}$-ontology $\mathcal{O}_{\mathfrak{P}}$ which asserts functionality of H, V, H^-, V^- and contains inclusions using additional concept names $U, R, Y, I_h, I_v, C, Z_{c,1}, Z_{c,2}, Z_{h,1}, Z_{h,2}, Z_{v,1}, Z_{v,2}$. The concept names U and R are used to mark the upper and right border of the rectangle, Y is used to mark points in the rectangle, and the remaining concept names are used for technical purposes explained below. In the following, for $e \in \{c, h, v\}$, we let $\mathcal{B}_e$ range over all Boolean combinations of the concept names $Z_{e,1}$ and $Z_{e,2}$, i.e., over all concepts $L_1 \sqcap L_2$ where L_i is a literal over $Z_{e,i}$, for $i \in \{1,2\}$. The ontology $\mathcal{O}_{\mathfrak{P}}$ contains the following concept inclusions, where $(T_i, T_j) \in \mathbb{H}$ and $(T_i, T_\ell) \in \mathbb{V}$:

$$\begin{aligned}
T_{\mathsf{final}} &\sqsubseteq Y \sqcap U \sqcap R \\
\exists H.(U \sqcap Y \sqcap T_j) \sqcap I_h \sqcap T_i &\sqsubseteq U \sqcap Y \\
\exists V.(R \sqcap Y \sqcap T_\ell) \sqcap I_v \sqcap T_i &\sqsubseteq R \sqcap Y \\
\exists H.(T_j \sqcap Y \sqcap \exists V.Y) & \\
\sqcap \exists V.(T_\ell \sqcap Y \sqcap \exists H.Y) & \\
\sqcap I_h \sqcap I_v \sqcap C \sqcap T_i &\sqsubseteq Y \\
\exists H.\exists V.\mathcal{B}_c \sqcap \exists V.\exists H.\mathcal{B}_c &\sqsubseteq C \\
\mathcal{B}_h \sqcap \exists H.\exists H^-.\mathcal{B}_h &\sqsubseteq I_h \\
\mathcal{B}_v \sqcap \exists V.\exists V^-.\mathcal{B}_v &\sqsubseteq I_v
\end{aligned}$$

$$\begin{array}{rcl} T_i & \sqsubseteq & \forall V.\bot \\ T_j & \sqsubseteq & \forall H.\bot \\ U & \sqsubseteq & \forall H.U \\ R & \sqsubseteq & \forall V.R \\ \bigsqcup_{1 \leq s < t \leq p} T_s \sqcap T_t & \sqsubseteq & \bot \end{array}$$

where $T_i \in \mathfrak{U}$ and $T_j \in \mathfrak{R}$.

The first four inclusions propagate the concept Y downwards and leftwards starting from a point marked with the final tile T_{final}. Note that these inclusions enforce the horizontal and vertical matching conditions. The concept inclusion with right-hand side C serves to enforce confluence, i.e., C is entailed at a constant a if there is a constant b that is both an H-V-successor and a V-H-successor of a. This is so because, intuitively, $\mathcal{B}_c$ is universally quantified: if confluence fails, then we can interpret $Z_{c,1}$ and $Z_{c,2}$ so that neither of the two conjuncts on the left-hand side of the inclusion for C is satisfied. In a similar manner, the inclusion for I_h (resp. I_v) is used to ensure that H^- (resp. V^-) act as the inverse of H (resp. V) at all points in the rectangle.

The following property can be obtained by a minor modification of Lemma 30 in [Baader et al. 2010]:

LEMMA C.5. *$\mathfrak{P}$ admits a tiling if and only if there is a* $\mathbf{S}$*-instance $\mathfrak{D}$ which is consistent with $\mathcal{O}_{\mathfrak{P}}$ and such that $q_{\mathbf{S},\mathcal{O}_{\mathfrak{P}},T_{\mathsf{init}}(x)\wedge Y(x)}(\mathfrak{D}) \neq \emptyset$.*

Let $\varphi_{\mathfrak{P}}$ be the first-order translation of the conjunction of all $T_i \sqsubseteq \forall V.\bot$, $T_i \in \mathfrak{U}$, $T_j \sqsubseteq \forall H.\bot$, $T_j \in \mathfrak{R}$, and of $\bigsqcup_{1 \leq s < t \leq p} T_s \sqcap T_t \sqsubseteq \bot$. The following is readily checked:

Claim. For all S-instances $\mathfrak{D}$, $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \varphi_{\mathfrak{P}}$ iff $\mathfrak{D}$ is consistent with $\mathcal{O}_{\mathfrak{P}}$.

We now prove undecidability of query containment. Let $E \notin \mathbf{S}$ be a fresh concept name and let

$$\mathcal{O}_2 = \mathcal{O}_{\mathfrak{P}}, \quad \mathcal{O}_1 = \mathcal{O}_{\mathfrak{P}} \cup \{Y \sqcap T_{\mathsf{init}} \sqsubseteq E\}$$

Now one can prove that the following conditions are equivalent:

— $\mathfrak{P}$ admits a tiling;
— $(\mathbf{S}, \mathcal{O}_1, E(x))$ is not contained in $(\mathbf{S}, \mathcal{O}_2, E(x))$;
— $(\mathbf{S}, \mathcal{O}_1, \exists x.E(x))$ is not contained in $(\mathbf{S}, \mathcal{O}_2, \exists x.E(x))$

Assume first that $\mathfrak{P}$ admits a tiling. Then, by Lemma C.5, there is an S-instance $\mathfrak{D}$ which is consistent with $\mathcal{O}_{\mathfrak{P}}$ and such that $q_{\mathbf{S},\mathcal{O}_{\mathfrak{P}},T_{\mathsf{init}}(x)\wedge Y(x)}(\mathfrak{D}) \neq \emptyset$. It follows immediately that $q_{\mathbf{S},\mathcal{O}_1,E(x)}(\mathfrak{D}) \neq \emptyset$ and $q_{\mathbf{S},\mathcal{O}_1,\exists x.E(x)}(\mathfrak{D}) = 1$. On the other hand, since $\mathfrak{D}$ is consistent with $\mathcal{O}_2$ and E does not appear in $\mathcal{O}_2$, we have $q_{\mathbf{S},\mathcal{O}_2,E(x)}(\mathfrak{D}) = \emptyset$ and $q_{\mathbf{S},\mathcal{O}_2,\exists x.E(x)}(\mathfrak{D}) = 0$.

Next suppose that $\mathfrak{P}$ does not admit a tiling, and let $\mathfrak{D}$ be an S-instance that is consistent with $\mathcal{O}_1$. By Lemma C.5, $q_{\mathbf{S},\mathcal{O}_{\mathfrak{P}},T_{\mathsf{init}}(x)\wedge Y(x)}(\mathfrak{D}) = \emptyset$, and hence $q_{\mathbf{S},\mathcal{O}_1,\exists x.E(x)}(\mathfrak{D}) = 0$. The desired containments trivially follow.

To prove undecidability of FO-rewritability, we expand $\mathcal{O}_1$ to a new ontology $\mathcal{O}_3$. To define $\mathcal{O}_3$ we take a fresh role name $S \notin \mathbf{S}$ and two concept names A and F and set

$$\mathcal{O}_3 = \mathcal{O}_1 \cup \{\exists S.E \sqsubseteq E, E \sqcap F \sqsubseteq A\}$$

and $\mathbf{S}_3 = \mathbf{S} \cup \{S, F\}$.

Claim. The following conditions are equivalent:

— $\mathfrak{P}$ admits a tiling;

— $q_{\mathbf{S}_3,\mathcal{O}_3,A(x)}$ is not FO-rewritable;
— $q_{\mathbf{S}_3,\mathcal{O}_3,\exists x.A(x)}$ is not FO-rewritable.

Assume first that $\mathfrak{P}$ admits a tiling. By Lemma C.5, we can find an S-instance $\mathfrak{D}_\mathfrak{P}$ that is consistent with $\mathcal{O}_\mathfrak{P}$ and $b \in \mathsf{adom}(\mathfrak{D}_\mathfrak{P})$ such that $b \in q_{\mathbf{S},\mathcal{O}_\mathfrak{P},T_{\mathsf{init}}(x)\wedge Y(x)}(\mathfrak{D}_\mathfrak{P})$, and hence $b \in q_{\mathbf{S},\mathcal{O}_1,E(x)}(\mathfrak{D}_\mathfrak{P})$. We can use essentially the same argument as in Theorem 5.16 (proving non-existence of a finite obstruction set) to show that $q_{\mathbf{S},\mathcal{O}_1,A(x)}$ and $q_{\mathbf{S},\mathcal{O}_1,\exists x.A(x)}$ are not FO-rewritable. Specifically, we construct S-instances $\mathfrak{D}_m$ by taking the union of $\mathfrak{D}_\mathfrak{P}$ and the facts

$$F(a_0), S(a_0,a_1), \ldots, S(a_m,b).$$

It can be checked that

— $a_0 \in q_{\mathbf{S}_3,\mathcal{O}_3,A(x)}(\mathfrak{D}_m)$ for all $m > 0$;
— $a_0 \notin q_{\mathbf{S}_3,\mathcal{O}_3,A(x)}(\mathfrak{D}'_m)$, where $\mathfrak{D}'_m$ results from $\mathfrak{D}_m$ by removing some fact (a_k,a_{k+1}) from $\mathfrak{D}_m$.

It follows that no finite obstruction set exists, and hence that $q_{\mathbf{S},\mathcal{O}_1,A(x)}$ is not FO-rewritable. We can proceed similarly for $q_{\mathbf{S},\mathcal{O}_1,\exists x.A(x)}$.

Assume now that $\mathfrak{P}$ does not admit a tiling. Then for every S-instance $\mathfrak{D}$, $\mathfrak{D}$ is consistent with $\mathcal{O}_\mathfrak{P}$ if and only if $q_{\mathbf{S},\mathcal{O}_3,\exists x.A(x)}(\mathfrak{D}) = 0$. Thus, the query defined by $\neg\varphi_\mathfrak{P}$ is equivalent to $q_{\mathbf{S},\mathcal{O}_3,\exists x.A(x)}$, and the query defined by $(x = x) \wedge \neg\varphi_\mathfrak{P}$ is equivalent to $q_{\mathbf{S},\mathcal{O}_3,A(x)}$.

To prove undecidability of datalog-rewritability, we expand $\mathcal{O}_1$ to a new ontology $\mathcal{O}_4$. To define $\mathcal{O}_4$, we take fresh role names S and S' and fresh concept names P_1, P_2, P_3 and encode the 3-colorability problem as follows:

$$\begin{aligned}\mathcal{O}_4 \;=\; & \mathcal{O}_1 \cup \{\exists S.E \sqsubseteq E, \exists S'.A \sqsubseteq A\} \cup \{E \sqsubseteq P_1 \sqcup P_2 \sqcup P_3\} \cup \\ & \{P_i \sqcap P_j \sqsubseteq A \mid 1 \leq i < j \leq 3\} \cup \{P_i \sqcap \exists S'.P_i \sqsubseteq A \mid 1 \leq i \leq 3\}\end{aligned}$$

We use the schema $\mathbf{S}_4 = \mathbf{S} \cup \{S, S'\}$.

Claim. The following conditions are equivalent:

— $\mathfrak{P}$ admits a tiling;
— $q_{\mathbf{S}_4,\mathcal{O}_4,A(x)}$ is not datalog-rewritable;
— $q_{\mathbf{S}_4,\mathcal{O}_4,\exists x.A(x)}$ is not datalog-rewritable.

First suppose that $\mathfrak{P}$ admits a tiling. We have seen previously that this implies the existence of an S-instance $\mathfrak{D}_\mathfrak{P}$ which is consistent with $\mathcal{O}_\mathfrak{P}$ and contains $b \in \mathsf{adom}(\mathfrak{D}_\mathfrak{P})$ such that $b \in q_{\mathbf{S},\mathcal{O}_1,E(x)}(\mathfrak{D}_\mathfrak{P})$. We proceed similarly to Theorem 5.16. Given a connected undirected graph G, we define an S-instance $\mathfrak{D}$ as the union of $\mathfrak{D}_\mathfrak{P}$ and the facts $S(d,d')$ for all d, d' in G and $S'(d,d')$ for every edge $\{d,d'\}$ in G. It can be checked that

— $b \in q_{\mathbf{S}_4,\mathcal{O}_4,A(x)}$ iff G is not 3-colorable;
— $q_{\mathbf{S}_4,\mathcal{O}_4,\exists x.A(x)}(\mathfrak{D}) = 1$ iff G is not 3-colorable.

It follows directly that neither $q_{\mathbf{S},\mathcal{O}',A(x)}$ nor $q_{\mathbf{S},\mathcal{O}',\exists x.A(x)}$ are datalog-rewritable.

Next suppose that $\mathfrak{P}$ does not admit a tiling. Then for every S-instance $\mathfrak{D}$, we have that $\mathfrak{D}$ is consistent with $\mathcal{O}_\mathfrak{P}$ if and only if $q_{\mathbf{S},\mathcal{O}_4,\exists x.A(x)}(\mathfrak{D}) = 0$. We can then simply reuse the FO-rewritings $\neg\varphi_\mathfrak{P}$ and $(x = x) \wedge \neg\varphi_\mathfrak{P}$ from above, since these can be equivalently expressed as datalog queries. □