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

Meghyn Bienvenu
CNRS & Université Paris Sud
Orsay, France

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

Carsten Lutz
University of Bremen
Bremen, Germany

Frank Wolter
University of Liverpool
Liverpool, UK

ABSTRACT

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

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages—*Query languages*; H.2.5 [**Database Management**]: Heterogeneous Databases

Keywords

Ontology-Based Data Access; Query Answering; Query Rewriting

1. INTRODUCTION

Ontologies are logical theories that formalize domain-specific knowledge, thereby making it available for machine processing. Recent years have seen an increasing interest in using ontologies in data-intensive applications, especially in the context of intelligent systems, the semantic web, and in data integration. A much studied scenario is that of answering queries over an incomplete database under the open world semantics, taking into account knowledge provided by an ontology [19, 18, 16]. We refer to this as *ontology-based data access (OBDA)*.

There are several important use cases for OBDA. A classical one is to enrich an incomplete data source with background knowledge, in order to obtain a more complete set of answers to a query. For example, if a medical patient database contains the facts that patient1 has finding Erythema Migrans and patient2 has finding Lyme disease, and the ontology provides the background knowledge that a finding of Erythema Migrans is sufficient for diagnosing Lyme disease, then both patient1 and patient2 can be returned when querying for patients that have the diagnosis Lyme disease. This use of ontologies is also central to query answering in the semantic web. OBDA can also be used to enrich the data schema (that is, the relation symbols used in the presentation of the data) with additional symbols to be used in a query. For example, a patient database may contain facts such as patient1 has diagnosis Lyme disease and patient2 has diagnosis Listeriosis, and an ontology could add the knowledge that Lyme disease and Listeriosis are both bacterial infections, thus enabling queries such as "return all patients with a bacterial infection" despite the fact that the data schema does not include a relation or attribute explicitly referring to bacterial infections. Especially in the bio-medical domain, applications of this kind are fueled by the availability of comprehensive professional ontologies such as SNOMED CT and FMA. A third prominent application of OBDA is in data integration, where an ontology can be used to provide a uniform view on multiple data sources [40]. This typically involves mappings from the source schemas to the schema of the ontology, which we will not explicitly consider here.

We may view the actual database query and the ontology as two components of one composite query, which we call an *ontology-mediated query*. OBDA can then be described as the problem of answering ontology-mediated queries. The database queries used in OBDA are typically unions of conjunctive queries, while the ontologies are typically specified in an ontology language that is either a description logic, or, more generally, a suitable fragment of first-order logic. For popular choices of ontology languages, the data complexity of ontology-mediated queries can be CONP-complete, which has resulted in extensive research on finding tractable classes of ontology-mediated queries, as well as on finding classes of ontology-mediated queries that are amenable to efficient query answering techniques [17, 29, 32]. In particular, relevant classes of ontology-mediated queries have been identified that admit an FO-rewriting (i.e., that are equivalent to a first-order query), or, alternatively, admit a datalog-rewriting. FO-rewritings make it possible to answer ontology-based queries using traditional database management systems. This is considered one of the most promising approaches for OBDA, and is currently the subject of significant research activity, see for example [18, 28, 30, 31, 42].

PODS'13, June 22–27, 2013, New York, New York, USA.

The main aims of this paper are (i) to characterize the expressive power of ontology-mediated queries, both in terms of more traditional database query languages and from a descriptive complexity perspective and (ii) to make progress towards complete and decidable classifications of ontology-mediated queries, with respect to their data complexity, as well as with respect to FO-rewritability and datalog-rewritability.

We take an ontology-mediated query to be a triple $(\mathbf{S}, \mathcal{O}, q)$ where $\mathbf{S}$ is a *data schema*, $\mathcal{O}$ an ontology, and q a query. Here, the data schema $\mathbf{S}$ fixes the set of relation symbols than can occur in the data and the ontology $\mathcal{O}$ is a logical theory that may use the relation symbols from $\mathbf{S}$ as well as additional symbols. The query q can use any relation symbol that occurs in $\mathbf{S}$ or $\mathcal{O}$. As ontology languages, we consider a range of standard description logics (DLs) and several fragments of first-order logic that embed ontology languages such as Datalog$^{\pm}$ [15], namely the guarded fragment (GF), the unary negation fragment (UNFO), and the guarded negation fragment (GNFO). As query languages for q, we focus on unions of conjunctive queries (UCQs) and unary atomic queries (AQs). The latter are of the form $A(x)$, with A a unary relation symbol, and correspond to what are traditionally called *instance queries* in the OBDA literature (note that A may be a relation symbol from $\mathcal{O}$ that is not part of the data schema). These two query languages are among the most used query languages in OBDA. In the following, we use $(\mathcal{L}, \mathcal{Q})$ to denote the query language that consists of all ontology-mediated queries $(\mathbf{S}, \mathcal{O}, q)$ with $\mathcal{O}$ specified in the ontology language $\mathcal{L}$ and q specified in the query language $\mathcal{Q}$. For example, (GF,UCQ) refers to ontology-mediated queries in which $\mathcal{O}$ is a GF-ontology and q is a UCQ. We refer to such query languages $(\mathcal{L}, \mathcal{Q})$ as *ontology-mediated query languages (or, OBDA languages)*.

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

In Sections 4 and 5, we study ontology-mediated queries from a *descriptive complexity* perspective. In particular, we establish an intimate connection between OBDA query languages, constraint satisfaction problems, and MMSNP. Recall that constraint satisfaction problems (CSPs) form a subclass of the complexity class NP that, although it contains NP-hard problems, is in certain ways more computationally well-behaved. The widely known Feder-Vardi conjecture [24] states that there is a dichotomy between PTIME and NP for the class of all CSPs, that is, each CSP is either in PTIME or NP-hard. In other words, the conjecture asserts that there are no CSPs which are NP-intermediate in the sense of Ladner's theorem. Monotone monadic strict NP without inequality (abbreviated MMSNP) was introduced by Feder and Vardi as a logical generalization of CSP that enjoys similar computational properties [24]. In particular, it was shown in [24, 33] that there is a dichotomy between PTIME and NP for MMSNP sentences if and only if the Feder-Vardi conjecture holds.

In Section 4, we observe that ($\mathcal{ALC}$,UCQ) and many other OBDA languages based on UCQs have the same expressive power as the query language coMMSNP, consisting of all queries whose complement is definable by an MMSNP formula with free variables. In the spirit of descriptive complexity theory, we say that ($\mathcal{ALC}$,UCQ) *captures* coMMSNP. In fact, this result is a consequence of the results in Section 3 and the observation that MDDlog has the same expressive power as coMMSNP. It has fundamental consequences regarding the data complexity of ontology-mediated queries and the containment problem for such queries, which we describe next.

First, we obtain that there is a dichotomy between PTIME and CONP for ontology-mediated queries from ($\mathcal{ALC}$,UCQ) if and only if the Feder-Vardi conjecture holds, and similarly for many other OBDA languages based on UCQs. To appreciate this result, recall that considerable effort has been directed towards identifying tractable classes of ontology-mediated queries. Ideally, one would like to classify the data complexity of every ontology-mediated query within a given OBDA language such as ($\mathcal{ALC}$,UCQ). Our aforementioned result ties this task to proving the Feder-Vardi conjecture. Significant progress has been made in understanding the complexity of CSPs and MMSNPs [14, 12, 34], and the connection established in this paper facilitates the transfer of techniques and results from CSP and MMSNP in order to analyze the data complexity of query evaluation in ($\mathcal{ALC}$,UCQ). We also consider the standard extension $\mathcal{ALCF}$ of $\mathcal{ALC}$ with functional roles and note that, for query evaluation in ($\mathcal{ALCF}$,AQ), there is no dichotomy between PTIME and CONP unless PTIME = NP.

To establish a counterpart of (GF,UCQ) and (GNFO,UCQ) in the MMSNP world, we introduce guarded monotone strict NP (abbreviated GMSNP) as a generalization of MMSNP; specifically, GMSNP is obtained from MMSNP by allowing guarded second-order quantification in the place of monadic second-order quantification, similarly as in the transition from MDDlog to frontier-guarded disjunctive datalog. The resulting query language coGMSNP has the same expressive power as frontier-guarded disjunctive datalog, and therefore, in particular, (GF,UCQ) and (GNFO,UCQ) capture coGMSNP. We observe that GMSNP has the same expressive power as the extension MMSNP_2 of MMSNP proposed in [37]. It follows from our results in Section 3 that GMSNP (and thus MMSNP_2) is strictly more expressive than MMSNP, closing an open problem from [37]. We leave it as an open problem whether GMSNP is computationally as well-behaved as MMSNP, that is, whether there is a dichotomy between PTIME and NP if the Feder-Vardi conjecture holds.

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

In Section 5, we consider OBDA languages based on atomic queries and establish a tight connection to (certain generalizations of) CSPs. This connection is most easily stated for *Boolean* atomic queries (BAQs): we prove that ($\mathcal{ALC}$,BAQ) captures the query language that consists of all Boolean queries definable as the complement of a CSP. Similarly ($\mathcal{ALC}$,AQ) extended with the uni-

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

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

Related Work A connection between query answering in DLs and the negation-free fragment of disjunctive datalog was first discovered and utilized in the influential [39, 29], see also [44]. This research is concerned with answer-preserving translations of ontology-mediated queries into disjunctive datalog. In contrast to the current paper, it does not consider the expressive power of ontology-mediated queries, nor their descriptive complexity. A connection between DL-based OBDA and CSPs was first found and exploited in [36], in a setup that is different from the one studied in this paper. In particular, instead of focusing on ontology-mediated queries that consist of a data schema, an ontology, and a database query, [36] concentrates on ontologies while quantifying universally over all database queries and without fixing a data schema. It establishes links to the Feder-Vardi conjecture that are incomparable to the ones found in this paper, and does not consider the expressive power and descriptive complexity of queries used in OBDA.

2. PRELIMINARIES

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

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

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

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

To simplify exposition, we assume that FO-queries do not contain constants. We use FOQ to denote the set of all first-order queries, as defined above. Similarly, we use CQ and UCQ to refer to the class of conjunctive queries and unions of conjunctive queries, defined as usual and allowing the use of equality. AQ denotes the set of *atomic queries*, which are of the form $A(x)$ with A a unary relation symbol. Each of these is called a *query language*, which is defined abstractly as a set of queries. Besides FOQ, CQ, UCQ, and AQ, we consider various other query languages introduced later, including ontology-mediated ones and variants of datalog.

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

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

An *ontology-mediated query* over a schema $\mathbf{S}$ is a triple $(\mathbf{S}, \mathcal{O}, q)$, where $\mathcal{O}$ is an ontology and q a query over $\mathbf{S} \cup \mathsf{sig}(\mathcal{O})$, with $\mathsf{sig}(\mathcal{O})$ the set of relation symbols used in $\mathcal{O}$. Here, we call $\mathbf{S}$ the *data schema*. Note that the ontology can introduce symbols that are not in the data schema. As explained in the introduction, this allows the ontology to enrich the schema of the query q. Of course, we do not require that every relation of the data schema needs to occur in the ontology. We have explicitly included $\mathbf{S}$ in the specification of the ontology-mediated query to emphasize that the ontology-mediated query is interpreted as a query over $\mathbf{S}$.

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

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

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

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

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

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

$\mathsf{finding}(\mathsf{pat1}, \mathsf{jan12find1})$	$\mathsf{ErythemaMigrans}(\mathsf{jan12find1})$
$\mathsf{diagnosis}(\mathsf{pat2}, \mathsf{may7diag2})$	$\mathsf{Listeriosis}(\mathsf{may7diag2})$

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

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

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

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

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

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

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

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

where A ranges over concept names and R over role names. An *$\mathcal{ALC}$-ontology* $\mathcal{O}$ is a finite set of *concept inclusions* $C \sqsubseteq D$, with C and D $\mathcal{ALC}$-concepts. We define the semantics of $\mathcal{ALC}$-concepts by translation to FO-formulas with one free variable, as shown in Table 2. An $\mathcal{ALC}$-ontology $\mathcal{O}$ then translates into the set of FO-sentences $\mathcal{O}^* = \{\forall x.(C^*(x) \rightarrow D^*(x)) \mid C \sqsubseteq D \in \mathcal{O}\}$. On the right-hand side of Table 1, we show the $\mathcal{ALC}$-version of the guarded fragment ontology displayed on the left-hand side. Note that, although the translation is equivalence-preserving in this case, in general, the guarded fragment is a more expressive ontology language than $\mathcal{ALC}$. Throughout the paper, we do not explicitly distinguish between a DL ontology and its translation into FO.

We remark that, from a DL perspective, the above definitions of instances and certain answers correspond to making the *standard name assumption (SNA)* in ABoxes, which in particular implies the *unique name assumption*. We make the SNA only to facilitate uniform presentation; the SNA is inessential for the results presented in this paper.

Example 2 *Let $\mathcal{O}$ and $\mathbf{S}$ be as in Example 1. For $q_1(x) = \mathsf{BacterialInfection}(x)$, the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q_1)$ is equivalent to the union of conjunctive queries $\mathsf{LymeDisease}(x) \vee \mathsf{Listeriosis}(x)$. For $q_2(x) = \mathsf{HereditaryDisposition}(x)$, the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q_2)$ is equivalent to the query defined by the datalog program*

$$P(x) \leftarrow \mathsf{HereditaryDisposition}(x) \qquad \mathsf{goal}(x) \leftarrow P(x)$$
$$P(x) \leftarrow \mathsf{parent}(y,x) \wedge P(y)$$

but not to any first-order query.

3. OBDA AND DISJUNCTIVE DATALOG

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

A *disjunctive datalog rule* ρ has the form

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

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

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

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

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

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

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

3.1 Ontologies Specified in Description Logics

We show that ($\mathcal{ALC}$,UCQ) has the same expressive power as MDDlog and identify a fragment of MDDlog that has the same expressive power as ($\mathcal{ALC}$,AQ). In addition, we consider the extensions of $\mathcal{ALC}$ with inverse roles, role hierarchies, transitive roles, and the universal role, which we also relate to MDDlog and its fragments. To match the syntax of $\mathcal{ALC}$ and its extensions, we generally assume schemas to be binary throughout this section.[1]

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

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

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

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

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

We introduce a fresh unary relation symbol P_τ for every type τ, and we denote by $\mathbf{S}'$ the schema that extends $\mathbf{S}$ with these additional symbols. In the MDDlog program that we aim to construct, the relation symbols P_τ will be used as IDB relations, and the symbols from $\mathbf{S}$ will be the EBD relations.

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

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

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

The desired MDDlog program Π consists of the following collections of rules:

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

Intuitively, these rules 'guess' a pointwise extension $\mathfrak{D}'$ of $\mathfrak{D}$. Specifically, the types P_τ guessed in the first line determine which subconcepts of $\mathcal{O}$ are made true at each element of $\mathfrak{D}'$. Since MDDlog does not support existential quantifiers, the $\mathfrak{D}'_a$ parts of $\mathfrak{D}'$ cannot be guessed explicitly. Instead, the CQs included in the guessed types determine those non-core component queries that matched in the $\mathfrak{D}'_a$ parts. The second line ensures coherence of the guesses and the last line guarantees that q has the required match in $\mathfrak{D}'$. It is proved in the full version of this paper that the MDDlog query q_Π is indeed equivalent to $(\mathbf{S}, \mathcal{O}, q)$.

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

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

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

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

by taking the conjunctive query

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

It can be shown that the ontology-mediated query $(\mathbf{S}, \mathcal{O}, q)$, where $\mathbf{S}$ is the schema that consists of the EDB relations of Π, is equivalent to the query defined by Π. ❑

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

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

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

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

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

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

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

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

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

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

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

Theorem 3

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

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

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

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

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

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

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

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

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

3.2 Ontologies Specified in First-Order Logic

Ontologies formulated in description logic are not able to speak about relation symbols of arity greater than two.[2] To overcome this restriction, we consider the guarded fragment of first-order logic and the unary-negation fragment of first-order logic [6, 46]. Both generalize the description logic $\mathcal{ALC}$ in different ways. We also consider their natural common generalization, the guarded negation fragment of first-order logic [7]. Our results from the previous subsection turn out to generalize to all these fragments. We start by considering the unary negation fragment.

The *unary-negation fragment of first-order logic* (UNFO) [46] is the fragment of first-order logic that consists of those formulas that are generated from atomic formulas, including equality, using conjunction, disjunction, existential quantification, and *unary negation*, that is, negation applied to a formula with at most one free variable. Thus, for example, $\neg\exists xyR(x,y)$ belongs to UNFO, whereas $\exists xy\neg R(x,y)$ does not. It is easy to show that every $\mathcal{ALC}$-TBox is equivalent to a UNFO sentence.

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

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

Proof. (sketch) The translation from MDDlog to (UNFO,UCQ) is given by Theorem 1. Here, we provide the translation from (UNFO,UCQ) to MDDlog. Let $Q = (\mathbf{S}, \mathcal{O}, q) \in$ (UNFO,UCQ) be given. We assume that $\mathcal{O}$ is a single UNFO sentence that is in the normal form generated by the following grammar:

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

where each ψ_i is either a relational atom or a formula with at most one free variable generated by the same grammar, and the free variables in ψ_i are among $x, \mathbf{y}$. Note that no equality is used and that all generated formulas have at most one free variable. Easy syntactic manipulations show that every UNFO-formula with at most one free variable is equivalent to a disjunction of formulas generated by the above grammar. In the case of $\mathcal{O}$, we may furthermore assume that it is a *single* such sentence, rather than a disjunction, because $\mathsf{cert}_{q,\mathcal{O}_1 \vee \mathcal{O}_2}(\mathfrak{D})$ is the intersection of $\mathsf{cert}_{q,\mathcal{O}_1}(\mathfrak{D})$ and $\mathsf{cert}_{q,\mathcal{O}_2}(\mathfrak{D})$, and MDDlog is closed under taking intersections of queries.

Let $\mathsf{sub}(\mathcal{O})$ be the set of all subformulas of $\mathcal{O}$ with at most one free variable z (we apply a one-to-one renaming of variables as needed to ensure that each formula in $\mathsf{sub}(\mathcal{O})$ with a free variable has the same free variable z). Let k be the maximum of the number of variables in $\mathcal{O}$ and the number of variables in q. We denote by $\mathsf{cl}_k(\mathcal{O})$ the set of all formulas $\varphi(x)$ of the form

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

with $\mathbf{y} = (y_1, \ldots, y_m)$, $m \leq k$, 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, as in the proof of Theorem 1, $\mathsf{cl}_k(\mathcal{O})$ contains all CQs that use only symbols from q and whose size is bounded by the size of q. A *type* τ is a subset of $\mathsf{cl}_k(\mathcal{O})$; the set of all types is denoted $\mathsf{type}(\mathcal{O})$.

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

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

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

Proposition 1 *The Boolean query*

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

is definable in (GF,UCQ) and not in MDDlog.

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

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

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

As fragments of first-order logic, the unary-negation fragment and the guarded fragment are incomparable in expressive power. They have a common generalization, which is known as the guarded-negation fragment (GNFO) [8]. This fragment is defined in the same way as UNFO, except that, besides unary negation, we allow *guarded negation* of the form $\alpha \wedge \neg\varphi$, where the guard α is an atomic formula that contains all the variables of φ. Again, for simplicity, we consider here the equality-free version of the language, except that we allow the use of trivial equalities of the form $x = x$ as guards. As we will see, for the purpose of OBDA, GNFO is no more powerful than GF. Specifically, (GF, UCQ) and (GNFO, UCQ) are expressively equivalent to a natural generalization of MDDlog, namely *frontier-guarded DDlog*. Recall that a datalog rule is *guarded* if its body includes an atom that contains all variables which occur in the rule [27]. A weaker notion of guardedness, which we call here *frontier-guardedness*, inspired by [5, 7], requires that, for each atom α in the head of the rule, there is an atom β in the rule body such that all variables that occur in α occur also in β. We define a frontier-guarded DDlog query to be a query defined by a DDlog program in which every rule is frontier-guarded. Observe that frontier-guarded DDlog subsumes MDDlog.

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

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

4. OBDA AND MMSNP

We show that MDDlog captures coMMSNP and thus, by the results obtained in the previous section, the same is true for many OBDA languages based on UCQs. We then use this connection to transfer results from MMSNP to OBDA languages with UCQs, linking the data complexity of these languages to the Feder-Vardi conjecture and establishing decidability of query containment. We also propose GMSNP, an extension of MMSNP inspired by frontier guarded DDlog, and show that (GF,UCQ) and (GNFO,UCQ) capture coGMSNP, and that GMSNP has the same expressive power as a previously proposed extension of MMSNP called MMSNP_2.

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

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

where each α_i is of the form $X_i(\mathbf{x})$, $R(\mathbf{x})$ (with $R \in \mathbf{S}$), or $x = y$, and each β_i is of the form $X_i(\mathbf{x})$. In order to use MMSNP as a query language, and in contrast to the standard definition, we admit free FO-variables and speak of *sentences* to refer to MMSNP formulas without free variables. To connect with the query languages studied thus far, we are interested in queries obtained by the complements of MMSNP formulas: each MMSNP formula Φ over schema $\mathbf{S}$ with n free variables gives rise to a query

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

where we set $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi$ to true when $\mathfrak{D}$ is the empty instance (that is, $\mathsf{adom}(\mathfrak{D}) = \emptyset$) and Φ is a sentence. We observe that the resulting query language *coMMSNP* has the same expressive power as MDDlog.

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

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

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

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

Conversely, let Π be a k-ary MDDlog program and assume w.l.o.g. that each rule uses a disjoint set of variables. Reserve fresh variables $y_1, \ldots, y_k$ as free variables for the desired MMSNP formula, and let $X_1, \ldots, X_n$ be the IDB predicates in Π and $x_1, \ldots, x_m$ the FO-variables in Π that do not occur in the goal predicate. Set $\Phi_\Pi = \exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ where φ is the conjunction of all non-goal rules in Π plus the implication $\vartheta' \rightarrow \bot$ 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 i-th position with y_i, and then conjunctively adding $y_i = y_j$ whenever the i-th and j-th position in the rule head have the same variable. It can be proved that $q_{\Pi,\mathbf{S}} = q_{\Phi_\Pi,\mathbf{S}}$ for all schemas $\mathbf{S}$. ❑

Thus, the characterizations of OBDA languages in terms of MDDlog provided in Section 3 also establish the descriptive complexity of these languages by identifying them with (the complement of) MMSNP. Furthermore, Proposition 2 allow us to transfer results from MMSNP to OBDA. We start by considering the data complexity of the query evaluation problem: for a query q, the *evaluation problem* is to decide, given an instance $\mathfrak{D}$ and a tuple $\mathbf{a}$ of elements from $\mathfrak{D}$, whether $\mathbf{a} \in q(\mathfrak{D})$. Our first result is that the Feder-Vardi dichotomy conjecture for CSPs is true if and only if there is a dichotomy between PTIME and CONP for query evaluation in ($\mathcal{ALC}$,UCQ), and the same is true for several other OBDA languages. For brevity, we say that a query language *has a dichotomy between* PTIME *and* CONP, referring only implicitly to the evaluation problem.

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

Theorem 8 *($\mathcal{ALC}$,UCQ) has a dichotomy between* PTIME *and* CONP *iff the Feder-Vardi conjecture holds. The same is true for ($\mathcal{ALCHIU}$,UCQ) and (UNFO,UCQ).*

Recall that ($\mathcal{ALCF}$,UCQ) and ($\mathcal{S}$,UCQ) are two extensions of ($\mathcal{ALC}$,UCQ) that were identified in Section 3 to be more expressive than ($\mathcal{ALC}$,UCQ) itself. It was already proved in [36] (Theorem 27) that, compared to ontology-mediated queries based on $\mathcal{ALC}$, the functional roles of $\mathcal{ALCF}$ dramatically increase the computational power. This is true even for atomic queries.

Theorem 9 ([36]) *For every* NP*-Turing machine M, there is a query q in ($\mathcal{ALCF}$,AQ) such that the complement of the word problem of M has the same complexity as evaluating q, up to polynomial-time reductions. Consequently, ($\mathcal{ALCF}$,AQ) does not have a dichotomy between* PTIME *and* CONP *(unless* PTIME = NP*).*

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

There are other interesting results that can be transferred from MMSNP to OBDA. Here, we consider query containment. Specifically, the following general containment problem was proposed in [10] as a powerful tool for OBDA: given ontology-mediated queries $(\mathbf{S}, \mathcal{O}_i, q_i)$, $i \in \{1, 2\}$, decide whether for all $\mathbf{S}$-instances $\mathfrak{D}$, we have $\mathsf{cert}_{q_1,\mathcal{O}_1}(\mathfrak{D}) \subseteq \mathsf{cert}_{q_2,\mathcal{O}_2}(\mathfrak{D})$.[3] Applications include the optimization of ontology-mediated queries and managing the effects on query answering of replacing an ontology with a new, updated version. In terms of OBDA languages such as ($\mathcal{ALC}$,UCQ), the above problem corresponds to query containment in the standard sense: an $\mathbf{S}$-query q_1 is *contained in* an $\mathbf{S}$-query q_2, written $q_1 \subseteq q_2$, if for every $\mathbf{S}$-instance $\mathfrak{D}$, we have $q_1(\mathfrak{D}) \subseteq q_2(\mathfrak{D})$. Note that there are also less general (and computationally simpler) notions of query containment in OBDA that do not fix the data schema [19].

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

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

Note that this result is considerably stronger than those in [10], which considered only containment of ontology-mediated queries $(\mathbf{S}, \mathcal{O}, q)$ with q an atomic query since already this basic case turned out to be technically intricate. The treatment of CQs and UCQs was left open, including all cases stated in Theorem 10.

We now consider OBDA languages based on the guarded fragment and GNFO. By Proposition 1, (GF,UCQ) and (GNFO,UCQ) are strictly more expressive than MDDlog and we cannot use Proposition 2 to relate these query languages to the Feder-Vardi conjecture. Theorem 7 suggests that it would be useful to have

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

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 1 and Theorem 7, this yields the second statement of the following lemma; the first statement can be proved similarly to Proposition 2.

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

Although defined in a different way, GMSNP is essentially the same logic as MMSNP_2, which is studied in [37]. Specifically, MMSNP_2 is the extension of MMSNP in which monadic SO-variables range over sets of domain elements *and facts*, and where atoms of the form $X(R(\mathbf{x}))$ are allowed in place of atoms $X(x)$ with X an SO-variable and R from the data schema $\mathbf{S}$. Additionally, a guardedness condition is imposed, requiring that whenever an atom $X(R(\mathbf{x}))$ occurs in a rule head, then the atom $R(\mathbf{x})$ must occur in the rule body. Formally, the SO-variables X_i are interpreted in an instance $\mathfrak{D}$ as sets $\pi(X_i) \subseteq \mathsf{adom}(\mathfrak{D}) \cup \mathfrak{D}$ and $\mathfrak{D} \models_\pi X(R(x_1, \ldots, x_n))$ if $R(\pi(x_1), \ldots, \pi(x_n)) \in \pi(X)$. We observe the following.

Proposition 3 *GMSNP and MMSNP_2 have the same expressive power.*

Details for the proofs of both Theorem 11 and Lemma 3 are in the full version of this paper. In [37], it was left as an open question whether MMSNP_2 is more expressive than MMSNP, which is resolved by the results above.

We leave it as an interesting open question whether Theorem 8 can be extended to (GF,UCQ) and (GNFO,UCQ), that is, whether GMSNP (equivalently: MMSNP_2) has a dichotomy between PTIME and NP if the Feder-Vardi conjecture holds. While this question is implicit already in [37], the results established in this paper underline its significance from a different perspective.

5. OBDA AND CSP

We show that OBDA languages based on AQs capture CSPs (and generalizations thereof), and we transfer results from CSPs to OBDA languages. In comparison to the previous section, we obtain a richer set of results, and often even worst-case optimal decision procedures. Recall that each finite relational structure $\mathfrak{B}$ over a schema $\mathbf{S}$ gives rise to a *constraint satisfaction problem* which is to decide, given a finite relational structure $\mathfrak{A}$ over $\mathbf{S}$, whether there is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ (written $\mathfrak{A} \rightarrow \mathfrak{B}$). In this context, the relational structure $\mathfrak{B}$ is also called the *template* of the CSP.

CSPs give rise to a query language coCSP in the spirit of the query language coMMSNP introduced in the previous section. In its basic version, this language is Boolean and turns out to have exactly the same expressive power as ($\mathcal{ALC}$,BAQ), where BAQ is the class of *Boolean* atomic queries. To also cover non-Boolean AQs, we consider two natural generalizations of CSPs. First, a *generalized CSP* is defined by a finite *set $\mathcal{F}$ of templates*, rather than only a single one [25]. The problem then consists in deciding, given an input structure $\mathfrak{A}$, whether there is a template $\mathfrak{B} \in \mathcal{F}$ such that $\mathfrak{A} \rightarrow \mathfrak{B}$. Second, in a *(generalized) CSP with constant symbols*, both the template(s) and the input structure are endowed with constant symbols [23, 1]. To be more precise, let $\mathbf{S}$ be a schema and $\mathbf{c} = c_1, \ldots, c_m$ a finite sequence of distinct constant symbols. A *finite relational structure over* $\mathbf{S} \cup \mathbf{c}$ has the form $(\mathfrak{A}, d_1, \ldots, d_m)$ with $\mathfrak{A}$ a finite relational structure over $\mathfrak{A}$ that, in addition, interprets the constant symbols c_i by elements d_i of the domain dom of $\mathfrak{A}$, for $1 \leq i \leq m$. Let $(\mathfrak{A}, \mathbf{a})$ and $(\mathfrak{B}, \mathbf{b})$ be finite relational structures over $\mathbf{S} \cup \mathbf{c}$. A mapping h is a *homomorphism* from $(\mathfrak{A}, \mathbf{a})$ to $(\mathfrak{B}, \mathbf{b})$, written $(\mathfrak{A}, \mathbf{a}) \rightarrow (\mathfrak{B}, \mathbf{b})$, if it is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ and $h(a_i) = b_i$ for $1 \leq i \leq m$. A (generalized) CSP with constant symbols is then defined like a (generalized) CSP, based on this extended notion of homomorphism.

We now introduce the query languages obtained from the different versions of CSPs, where generalized CSPs with constant symbols constitute the most general case. Specifically, each finite set of templates $\mathcal{F}$ over $\mathbf{S} \cup \mathbf{c}$ with $\mathbf{c} = c_1, \ldots, c_m$ gives rise to an m-ary query coCSP($\mathcal{F}$) that maps every $\mathbf{S}$-instance $\mathfrak{D}$ to

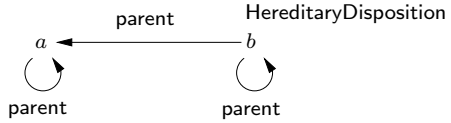
$$\{\mathbf{d} \in \mathsf{adom}(\mathfrak{D})^m \mid \forall (\mathfrak{B}, \mathbf{b}) \in \mathcal{F} : (\mathfrak{D}, \mathbf{d}) \not\rightarrow (\mathfrak{B}, \mathbf{b})\},$$

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

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

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

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



It can be shown that for all instances $\mathfrak{D}$ over $\mathbf{S}$ and for all $d \in \mathsf{adom}(\mathfrak{D})$, we have $d \in \mathsf{cert}_{q_2,\mathcal{O}}(\mathfrak{D})$ iff $(\mathfrak{D}, d) \not\rightarrow (\mathfrak{B}, a)$ and thus the query coCSP($\mathfrak{B}$) is as required.

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

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

1. *($\mathcal{ALCU}$,AQ), ($\mathcal{SHIU}$,AQ), unary simple MDDlog, and generalized coCSP with one constant symbol;*

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

3. *($\mathcal{ALCU}$,BAQ), ($\mathcal{SHIU}$,BAQ), Boolean simple MDDlog, and generalized coCSP;*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Theorem 13 *($\mathcal{ALC}$,BAQ) has a dichotomy between* PTIME *and* CONP *iff the Feder-Vardi conjecture holds. The same is true for ($\mathcal{SHIU}$,AQ), and ($\mathcal{SHIU}$,BAQ).*

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

5.1 Query Containment

In Section 4, we have established decidability results for query containment in OBDA languages based on UCQs. For OBDA languages based on AQs and BAQs, we even obtain a tight complexity bound. It is easy to see that query containment in coCSP is characterized by homomorphisms between templates. Consequently, it is straightforward to show that query containment for generalized coCSP with constant symbols is NP-complete. Thus, Theorem 12 yields the following NEXPTIME upper bound for query containment in OBDA languages. The corresponding lower bound is proved in the full version of this paper by a non-trivial reduction of a NEXPTIME-complete tiling problem.

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

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

5.2 FO- and Datalog-Rewritability

One prominent approach to answering ontology-mediated queries is to make use of existing relational database systems or

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

Proposition 4 *If $Q = (\mathbf{S}, \mathcal{O}, q)$ is an ontology-mediated query with $\mathcal{O}$ formulated in equality-free FO and q a UCQ, then q_Q is preserved by homomorphisms. Consequently, it follows from [43] that if q_Q is FO-rewritable, then q_Q is rewritable into a UCQ (thus into datalog).*

Example 2 illustrates that ontology-mediated queries are not always rewritable into an FO-query, and the same holds for datalog-rewritability. It is a central problem to decide, given an ontology-mediated query, whether it is FO-rewritable and whether it is datalog-rewritable. By leveraging the CSP connection, we show that both problems are decidable and pinpoint their complexities.

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

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

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

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

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

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

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

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

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

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

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

From Theorems 12 and 16, we obtain a NEXPTIME upper bound for deciding FO-rewritability and datalog-rewritability of ontology-mediated queries based on DLs and (B)AQs. The corresponding lower bounds are proved in the full version of this paper using a reduction from a NEXPTIME-hard tiling problem (in fact, the same problem as in the lower bound for query containment).

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

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

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

6. CONCLUSION

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

We mention two natural lines of future research. First, it would be interesting to understand the data complexity and query containment problem for (GF,UCQ) and (GNFO,UCQ). In particular, we would like to know whether Theorems 8 and 10 extend to (GF,UCQ) and (GNFO,UCQ). As explained in Section 4, resolving this question for Theorem 8 is equivalent to clarifying the computational status of GMSNP and MMSNP_2.

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

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

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

7. REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[46] B. ten Cate and L. Segoufin. Unary negation. In *STACS*, 2011.
[47] W3C OWL Working Group. OWL 2 Web Ontology Language. http://www.w3.org/TR/owl2-overview/, 2012.

APPENDIX

A. PROOFS FOR SECTION 3

A.1 Proofs for Section 3.1

We remark that the direction "from ($\mathcal{ALC}$,AQ) to MDDlog" of Theorem 1 is actually a consequence of Theorem 6, which makes a strictly more general statement. We still provide it here (and in the main paper) as a warmup for the proof of Theorem 6. As an extra bit of notation, we say that an assignment π of elements of an instance $\mathfrak{D}$ to the variables of a CQ q is a *match of q in $\mathfrak{D}$* if $\mathfrak{D}$ satisfies q under π.

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

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

Claim. For all instances $\mathfrak{D}$ over $\mathbf{S}$ and all $\mathbf{a} \in \mathsf{adom}(\mathfrak{D})^m$, 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,

$$\mu(b) = \{q' \in \mathsf{cl}(\mathcal{O}, q) \mid q' \text{ is Boolean and } \mathfrak{D}' \models q'\} \cup \{C \in \mathsf{cl}(\mathcal{O}, q) \mid C \text{ is unary and } \mathfrak{D}' \models C[b]\}.$$

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 model $(\mathsf{dom}_a, \mathfrak{D}_a)$ of $\mathcal{O}$ in which $\mu(a)$ is realized at 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 $d \in \mathsf{dom}_a$. It is not difficult to show by induction on the structural complexity of C that for all concepts $C \in \mathsf{cl}(\mathcal{O}, q) \cap \mathsf{sub}(\mathcal{O})$ and all $d \in \mathsf{dom}''$, we have

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

(refer to the proof of Theorem 2 for details). Since $\mathsf{cl}(\mathcal{O}, q)$ by definition includes C and D whenever $C \sqsubseteq D$ is in $\mathcal{O}$, 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 $\delta(\mathbf{x})$ based on the restriction of the original model $\mathfrak{D}'$ of Π , as follows: $\delta(\mathbf{x})$ contains (a) all atoms $A(x)$ such that $\pi(x) \in \mathsf{adom}(\mathfrak{D}')$ and $A(\pi(x)) \in \mathfrak{D}'$ (where A can be either a concept name or of the form P_τ), (b) all atoms $R(x, y)$ such that $\pi(x), \pi(y) \in \mathsf{adom}(\mathfrak{D}')$ and $R(\pi(x), \pi(y)) \in \mathfrak{D}'$, and (c) all atoms $P_{\mu(d)}(z_d)$ (with z_d a fresh variable) such that $P_{\mu(d)}(d) \in \mathfrak{D}'$ and there is some $\pi(w) \in \mathsf{dom}_d$. Atoms of type (c) are used to handle the case in which a Boolean subquery q'' of q' is mapped inside $\mathcal{D}_d$, but the element d 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 $\pi(z_d) = d$. Since $\delta(\mathbf{x})$ is satisfied in $\mathfrak{D}'$ under π and $\pi(\mathbf{x}') = \mathbf{a}$, 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 $\mathfrak{B}$. Consider the following CQs:

- q_0 is the restriction of q' to those variables that π maps to elements of $\mathfrak{D}$;
- for each $a \in \mathsf{adom}(\mathfrak{D})$ such that some element of dom_a is in the range of π, the CQ q_a is obtained by first taking the restriction of q' to those variables that π maps to elements of dom_a and then identifying all variables that π maps to the same element (preserving the names of free variables).

Clearly, each q_a has at most one free variable, which, if it exists, is mapped to a by π.

We start by showing that q_0 is satisfied in $\mathfrak{B}$ under τ. For role atoms in q_0, this is immediate since all such atoms also belong to $\delta(\mathbf{x})$. Thus, consider some concept atom $A(x) \in q_0$. Since $A(x) \in q'$ and π is a match for q' in $\mathcal{D}''$, we have $A(\pi(x)) \in \mathcal{D}''$. Then using the fact that $A \in \mathsf{cl}(\mathcal{O}, q) \cap \mathsf{sub}(\mathcal{O})$ and Equation (1) above, we obtain $A \in \mu(\pi(x))$. We know that $P_\mu(\pi(x))(\pi(x)) \in \mathcal{D}'$, so by construction of $\delta(\mathbf{x})$, we must have $P_\mu(\pi(x))(x) \in \delta(\mathbf{x})$, hence $P_\mu(\pi(x))(\tau(x)) \in \mathfrak{B}$. Using the type-coherence of $\mathfrak{B}$ and the fact that $A \in \mu(\pi(x))$, we obtain $A(\tau(x)) \in \mathfrak{B}$, as desired.

Now consider a query q_a. By construction, the length of q_a cannot exceed the length of q, and so $q_a \in \mathsf{cl}(\mathcal{O}, q)$. Since q_a has a match in $\mathfrak{D}_a$ (such that, if q_a has a free variable, it is mapped to a) and $\mathfrak{D}_a$ realizes the type $\mu(a)$ at a, we must have $q_a \in \mu(a)$. By construction of $\delta(\mathbf{x})$, there is an atom $P_{\mu(a)}(x) \in \delta(\mathbf{x})$. 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_a of q_a in $\mathfrak{B}$ (such that, if q_a has a free variable, τ_a maps it to $\tau(x)$). It is not hard to see that the matches τ and τ_a can be assembled into a match τ' of q' in $\mathfrak{B}$ which coincides with τ on $\mathbf{x}'$. ❑

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

Proof. (continued) We establish here the correctness of the translation from ($\mathcal{ALC}$,AQ) to MDDlog. That is, we show that, for every instance $\mathfrak{D}$ and elements $a \in \mathsf{adom}(\mathfrak{D})$, we have $a \in \mathsf{cert}_{q,\mathcal{O}}(\mathfrak{D})$ if and only if $a \in q_\Pi(\mathfrak{D})$.

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

"only if". Assume that $a \notin q_\Pi(\mathfrak{D})$ and let $\mathfrak{D}'$ be a model of Π with $\mathfrak{D} \subseteq \mathfrak{D}'$ that does not contain $\mathsf{goal}(a)$. For each $b \in \mathsf{adom}(\mathfrak{D})$, let $\mu(b)$ be a type such that $P_{\mu(b)}(b) \in \mathfrak{D}'$ (in fact, the rules in Π enforce that there is exactly one such $\mu(b)$). Note that $A \notin \mu(a)$. Also note that each type $\mu(b)$ must be realizable in some model of $\mathcal{O}$ (else, there would be a rule forbidding $P_\mu(b)$ atoms). Thus, for each $b \in \mathsf{adom}(\mathfrak{D})$, we can find a model $(\mathsf{dom}_b, \mathfrak{D}_b)$ of $\mathcal{O}$ in which the type $\mu(b)$ is realized at b. We may assume that these models have disjoint domains. Let $(\mathsf{dom}'', \mathfrak{D}'')$ be obtained by first taking the union of $(\mathsf{dom}_b, \mathfrak{D}_b)_{b \in \mathsf{adom}(\mathfrak{D})}$, and then adding all facts in $\mathfrak{D}$. By construction, $\mathfrak{D} \subseteq \mathfrak{D}''$ and $a \notin q(\mathfrak{D}'')$. It remains to show that $(\mathsf{dom}'', \mathfrak{D}'')$ is a model of $\mathcal{O}$.

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 structural complexity of C:

$(*)$ For every concept $C \in \mathsf{sub}(\mathcal{O})$ and every $d \in \mathsf{dom}''$, we have $(\mathsf{dom}'', \mathfrak{D}'') \models C(d)$ iff $C \in \mu(d)$.

Note that it follows from $(*)$ that $(\mathsf{dom}'', \mathfrak{D}'')$ is a model of $\mathcal{O}$.

For the base case, first suppose that $A \in \mu(d)$, with A a concept name and $d \in \mathsf{dom}_a$. Then $A(d) \in \mathfrak{D}_a \subseteq \mathfrak{D}''$, so $(\mathsf{dom}'', \mathfrak{D}'') \models A(d)$. Next suppose that $(\mathsf{dom}'', \mathfrak{D}'') \models A(d)$. Then $A(d) \in \mathfrak{D}''$, so either $A(d) \in \mathfrak{D}_a$, or $d = a$ and $A(d) \in \mathfrak{D}$. In the former case, we immediately obtain $A \in \mu(d)$. In the latter case, note that if $A \notin \mu(d)$, then Π would contain the rule $\bot \leftarrow P_{\mu(d)}(x) \wedge A(x)$, and this would yield a contradiction since $\{A(d), P_{\mu(d)}(d)\} \subseteq \mathfrak{D}'$.

The inductive step for the Boolean operators is trivial, so we consider only the case of the $\exists R$ constructor (the argument for the $\forall R$ constructor is similar). Thus, let $C = \exists R.D$ and $d \in \mathsf{dom}_a$, and suppose that $C \in \mu(d)$. Then $(\mathsf{dom}_a, \mathfrak{D}_a) \models \exists R.D(d)$, so there exists $e \in \mathsf{dom}_a$ such that $R(d, e) \in \mathfrak{D}_a$ and $(\mathsf{dom}_a, \mathfrak{D}_a) \models D(e)$. It follows that $D \in \mu(e)$, and hence by the induction hypothesis, we must have $(\mathsf{dom}'', \mathfrak{D}'') \models D(e)$. Since $\mathfrak{D}_a \subseteq \mathfrak{D}''$, we have $R(d, e) \in \mathfrak{D}''$, which yields $(\mathsf{dom}'', \mathfrak{D}'') \models C(d)$.

Conversely, suppose $(\mathsf{dom}'', \mathfrak{D}'')$ satisfies $\exists R.D(d)$, that is, there is an element e such that $(\mathsf{dom}'', \mathfrak{D}'')$ satisfies $R(d, e)$ and $D(e)$. If $e \in \mathsf{dom}_a$, the claim $(*)$ follows immediately from the induction hypothesis. Otherwise, we must have that $e \in \mathsf{adom}(\mathfrak{D})$ and, by induction hypothesis, $D \in \mu(e)$. It follows that $\exists R.D \in \mu(d)$, because otherwise $P_{\mu(d)}(x) \wedge R(x, y) \wedge P_{\mu(e)}(y)$ would be a non-realizable diagram, and Π would derive an inconsistency. ❑

Theorem 3.

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

To complete the proof of Theorem 3, we need to show that the queries from ($\mathcal{S}$,UCQ) and ($\mathcal{ALCF}$,UCQ) indicated in the proof sketch cannot be expressed in ($\mathcal{ALC}$,UCQ), or equivalently, MDDlog. We start by providing a means of identifying queries which cannot be expressed in MDDlog, using the notion of colored instances, defined as follows:

Definition 1 *Let $\mathbf{S}$ be a schema and $\mathcal{C}$ be a set of unary predicates (colors) $\{C_1, \ldots, C_n\}$ disjoint from $\mathbf{S}$. A* $\mathcal{C}$-colored $\mathbf{S}$-structure *is an $\mathbf{S} \cup \mathcal{C}$-structure $(\mathsf{dom}, \mathfrak{D})$ such that*

- *For every $d \in \mathsf{dom}$, $C_i(d) \in \mathfrak{D}$ for some i;*
- *If $C_i(d) \in \mathfrak{D}$, then $C_j(d) \notin \mathfrak{D}$ for every $j \neq i$.*

$\mathfrak{D}$ is called a $\mathcal{C}$-coloring *of an $\mathbf{S}$-structure $\mathfrak{D}'$ if $\mathfrak{D}'$ is the $\mathbf{S}$-reduct of $\mathfrak{D}'$.*

*Now for each $k > 0$, fix $\mathcal{C}_k$ with $|\mathcal{C}_k| = k$ and $\mathcal{C}_k \cap \mathbf{S} = \emptyset$. Then a k-*coloring *of $\mathfrak{D}$ is simply a $\mathcal{C}_k$-coloring of $\mathfrak{D}$.*

We will also utilize the notion of forbidden pattern problems from [38, 34, 12], whose definition we recall here.

Definition 2 *Given a set $\mathcal{F}$ of $\mathcal{C}$-colored $\mathbf{S}$-structures (called* forbidden patterns*), we define $\mathsf{Forb}(\mathcal{F})$ as the set of all $\mathbf{S}$-structures $\mathfrak{D}$ such that there exists a $\mathcal{C}$-coloring $\mathfrak{D}'$ of $\mathfrak{D}$ for which $\mathfrak{F} \not\to \mathfrak{D}'$ for every $\mathfrak{F} \in \mathcal{F}$. The* forbidden patterns problem defined by $\mathcal{F}$ *is to decide whether a given $\mathbf{S}$-structure belongs to $\mathsf{Forb}(\mathcal{F})$.*

Analogously to coMMSNP, we can define a query language coFPP consisting of all those Boolean queries $q_{\mathcal{F},\mathbf{S}}$ defined by

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

with $\mathcal{F}$ a set of $\mathcal{C}$-colored $\mathbf{S}$-structures. It follows directly from results in [38] that coMMSNP and coFPP have the same expressive power. Combining this result with Proposition 2 (from Section 4), we obtain the following:

Proposition 6 *coFPP and Boolean MDDlog have the same expressive power.*

We use Proposition 6 in the proof of the following lemma, whose purpose is to establish a sufficient condition for non-expressibility in MDDlog.

Lemma 1 *A Boolean query Q over schema $\mathbf{S}$ does not belong to MDDlog if for every $m, n > 0$, 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 m-coloring $\mathfrak{B}_0$ of $(\mathsf{adom}(\mathfrak{D}_0), \mathfrak{D}_0)$, there exists an m-coloring $\mathfrak{B}_1$ of $(\mathsf{adom}(\mathfrak{D}_1), \mathfrak{D}_1)$ such that from every substructure of $\mathfrak{B}_1$ having at most n elements there is a homomorphism to $\mathfrak{B}_0$.*

Proof. Assume for a contradiction that the conditions of the lemma hold for every $n, m > 0$ but that Q is equivalent to some query in MDDlog. Then, by Proposition 6, there is a set $\mathcal{F}$ of $\mathcal{C}$-colored $\mathbf{S}$-structures such that for all $\mathbf{S}$-instances $\mathfrak{D}$, we have $Q(\mathfrak{D}) = 1$ if and only if $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \notin \mathsf{Forb}(\mathcal{F})$. Let $m_0 = |\mathcal{C}|$, and let n_0 be the maximal number of elements in the domain of some $\mathfrak{F} \in \mathcal{F}$. We can assume w.l.o.g. that $\mathcal{C} = \mathcal{C}_{m_0}$.

Take $\mathbf{S}$-instances $\mathfrak{D}_0$ and $\mathfrak{D}_1$ satisfying the conditions of the lemma for m_0, n_0. As $Q(\mathfrak{D}_0) = 0$, there exists a $\mathcal{C}$-coloring $\mathfrak{B}_0$ of $(\mathsf{adom}(\mathfrak{D}_0), \mathfrak{D}_0)$ such that $\mathfrak{F} \not\to \mathfrak{B}_0$ for every $\mathfrak{F} \in \mathcal{F}$. It follows that there exists a $\mathcal{C}$-coloring $\mathfrak{B}_1$ of $(\mathsf{adom}(\mathfrak{D}_1), \mathfrak{D}_1)$ such that from every substructure of $\mathfrak{B}_1$with at most n_0 elements, there exists 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 $(\mathsf{adom}(\mathfrak{D}_0), \mathfrak{D}_0) \in \mathsf{Forb}(\mathcal{F})$. ❑

Using the preceding lemma, we can now prove that the queries mentioned in the proof sketch cannot be expressed in MDDlog.

Lemma 2 *There exist queries in ($\mathcal{S}$,UCQ) which do not belong to MDDlog.*

Proof. Consider $Q = (\mathbf{S}, \mathcal{O}, q)$ where $\mathbf{S} = \{R, S\}$, $\mathcal{O}$ asserts transitivity of R and S, and $q = \exists xy(R(x, y) \wedge S(x, y))$.

We apply Lemma 1. Assume that $m, n > 0$ are given. Let $k = n - 1$ and $k' = m^{k+2} + 1$. Define $\mathfrak{D}_1$ and $\mathfrak{D}_0$ as follows:

- $\mathfrak{D}_1$ has elements e, f and $a_1, \dots, a_k$ and $b_1, \dots, b_k$ and the atoms $R(e, a_1), R(a_k, f)$ and $R(a_i, a_{i+1})$ for $1 \leq i < k$, and $S(e, a_1), S(a_k, f)$ and $S(b_i, b_{i+1})$ for $1 \leq i < k$.
- $\mathfrak{D}_0$ has elements $e^1, \dots, e^{k'}$ and $f^1, \dots, f^{k'}$ as well as $a_1^j, \dots, a_k^j$ for $1 \leq j \leq k'$ and $b_1^{i,j}, \dots, b_k^{i,j}$ for $1 \leq j < i \leq k'$. The atoms of $\mathfrak{D}_0$ consist of:
  - $R(e^i, a_1^i), R(a_k^i, f^i)$, and $R(a_j^i, a_{j+1}^i)$ for $1 \leq i \leq k'$ and $1 \leq j < k$;
  - $S(e_i, b_1^{i,j})$ and $S(b_k^{i,j}, f_j)$ for $1 \leq j < i \leq k'$, and $S(b_l^{i,j}, b_{l+1}^{i,j})$ for $1 \leq l < k$ and $1 \leq j < i \leq k'$.

It is readily checked that $Q(\mathfrak{D}_0) = 0$ and $Q(\mathfrak{D}_1) = 1$, as required. Let $\mathfrak{B}_0$ be an m-coloring of $(\mathsf{adom}(\mathfrak{D}_0), \mathfrak{D}_0)$. Since $k' = m^{k+2} + 1$, we can find i, i' with $i > i'$ such that the colorings of $e^i, a_1^i, \dots, a_k^i, f^i$ and $e^{i'}, a_1^{i'}, \dots, a_k^{i'}, f^{i'}$ coincide. Define an m-coloring of $(\mathsf{adom}(\mathfrak{D}_1), \mathfrak{D}_1)$ by taking the coloring of $e^i, a_1^i, \dots, a_k^i, f^i$ for $e, a_1, \dots, a_k, f$ and the coloring of $b_1^{i,i'}, \dots, b_k^{i,i'}$ for $b_1, \dots, b_k$. Denote by $\mathfrak{B}_1$ the resulting colored structure.

Consider a subset C of $\mathsf{adom}(\mathfrak{B}_1)$ having at most n elements, and let $\mathfrak{B}_1'$ be the restriction of $\mathfrak{B}_1$ to the elements in C. We define a function h from C to $\mathsf{adom}(\mathfrak{B}_0)$ as follows:

- If $e \notin C$, then let h be the restriction of the following mapping to C: $h(a_l) = a_l^{i'}$, $h(b_l) = b_l^{i,i'}$ and $h(f) = f^{i'}$;
- If $f \notin C$, then let h be the restriction of the following mapping to C: $h(a_l) = a_l^i$, $h(b_l) = b_l^{i,i'}$ and $h(e) = e^i$;
- Otherwise there exists $a_{i_0} \notin C$. Then let h be the restriction of the following mapping to C: $h(e) = e^i$, $h(a_l) = a_l^i$ for all $l < i_0$, $h(a_l) = a_l^{i'}$ for all $l > i_0$, $h(b_l) = b_l^{i,i'}$ for all $1 \leq l \leq k$, and $h(f) = f^{i'}$.

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

Lemma 3 *There exist queries in (ALCF,UCQ) which do not belong to MDDlog.*

Proof. Consider $Q = (\mathbf{S}, \mathcal{O}, \exists x.A(x))$ where $\mathbf{S} = \{S, A\}$ and $\mathcal{O}$ states that S is functional. Set $\mathfrak{D}_1 = \{S(a, b), S(a, c)\}$ and $\mathfrak{D}_0 = \{S(a, b)\}$. Note that $q_Q(\mathfrak{D}_1) = 1$ (since no model of $\mathcal{O}$ contains $\mathfrak{D}_1$) and $q_Q(\mathfrak{D}_0) = 0$. Let $\mathfrak{B}_0$ be any m-coloring of $(\mathsf{adom}(\mathfrak{D}_0), \mathfrak{D}_0)$. We define an m-coloring $\mathcal{B}_1$ of $\mathfrak{D}_1$ by assigning a, b the same colors as in $\mathfrak{B}_0$ and giving c the same color as b. Then the mapping sending a to itself and b, c to b defines a homomorphism from $\mathfrak{B}_1$ to $\mathfrak{B}_0$ (and hence also defines a homomorphism from any substructure of $\mathfrak{B}_1$ to $\mathfrak{B}_0$). It follows by Lemma 1 that Q is not definable in MDDlog. ❑

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

Proof. We first show

- (ALCU,AQ) is at least as expressive as unary simple MDDlog;
- unary simple MDDlog is at least as expressive as (ALCIU,AQ).

For Point 1, let Π be a unary simple MDDlog program. The rewriting of each rule of Π into an equivalent $\mathcal{ALCU}$-concept inclusion is similar to the proof of Theorem 2 except that now one also has to concider non-connected bodies. They 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 $A \sqcap \exists U.(B \sqcap \neg P_2) \sqsubseteq P_1$.

Now consider Point 2. The translation from (ALCIU,AQ) to unary simple MDDlog queries is a modified version of the translation given in the proof of Theorem 2 for the translation from (ALC,AQ) to *connected* unary simple MDDlog queries.

Assume that $(\mathbf{S}, \mathcal{O}, q)$ with $q = A(x)$ is given. As in Theorem 2, we take types to be subsets of $\mathsf{sub}(\mathcal{O})$. The MDDlog program Π consists of the following rules:

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

Note that the only difference with the rules in the proof of Theorem 2 is the presence of rules of the form

$$\bot \leftarrow P_{\tau_1}(x_1) \wedge P_{\tau_2}(x_2)$$

which are not connected. Π is still unary and simple. Equivalence of $(\mathbf{S}, \mathcal{O}, q)$ and q_Π can now be proved similarly to Theorem 2.

It remains to be shown that (ALCIU,AQ) and (SHIU,AQ) are equally expressive. But this is again folkore [39, 45]: it is known that for every $\mathcal{SHIU}$-ontology $\mathcal{O}$, there exists an $\mathcal{ALCIU}$-ontology $\mathcal{O}'$ (possibly using additional concept names) such that (i) $\mathcal{O}' \models \mathcal{O}$ and (ii) for every $\mathfrak{A} \in \mathsf{Mod}(\mathcal{O})$, there exists a model $\mathfrak{A}' \in \mathsf{Mod}(\mathcal{O}')$ with the same domain and interpreting the concept names of $\mathcal{O}$ in the same way as $\mathfrak{A}$ and interpreting the role names as relations containing their interpretation in $\mathfrak{A}$. It follows that (ALCIU,AQ) and (SHIU,AQ) are equally expressive. ❑

We briefly discuss *Boolean atomic queries* (BAQs), i.e., queries of the form $\exists x.A(x)$, where A is a unary relation symbol. BAQs behave similarly to AQs and one can show modified versions of Theorems 2 to Theorem 5 above in which AQs are replaced by BAQs and unary goal predicates by 0-ary goal-predicate, respectively.

Theorem 19 *Theorems 2 to Theorem 5 hold if AQs are replaced by BAQs and unary goal predicates by* 0*-ary goal-predicate, respectively.*

Proof. We show the required modifications to the proof of Theorem 2. The remaining results are proved by similar modifications and left to the reader. For the translation from (ALC,BAQ) to Boolean connected simple MDDlog, the only difference to the program constructed in the proof of Theorem 2 is that rules of the form $\mathsf{goal}(x) \leftarrow P_\tau(x)$ are replaced by rules of the form $\mathsf{goal} \leftarrow P_\tau(x)$. Conversely, for the translation from Boolean connected simple MDDlog to (ALC,BAQ), we regard goal as a concept name and take the BAQ $\exists x.\mathsf{goal}(x)$. The rewriting of goal rules must also be accordingly modified. For example, $\mathsf{goal} \leftarrow R(x, y)$ is rewritten into $\exists R.\top \sqsubseteq \mathsf{goal}$. ❑

A.2 Proofs for Section 3.2

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

Proof. (continued) We establish here the correctness of the translation from (UNFO,UCQ) to MDDlog. That is, 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 1, 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 $\varphi \in \mathsf{cl}_k(\mathcal{O})$ 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 $\mathsf{cl}_k(\mathcal{O})$ 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 define a number of formulas by syntactic operations on φ. It will follow from the definition of $\mathsf{cl}_k(\mathcal{O})$ that each of these formulas again belongs to $\mathsf{cl}_k(\mathcal{O})$, 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.

❑

Proposition 1. *The Boolean query*

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

is definable in (GF,UCQ) and not in MDDlog.

Proof. Let $\mathbf{S}$ consist of unary predicates A, B and a ternary predicate P, and let Q be the $\mathbf{S}$-query defined by (†). A (GF,UCQ) query expressing Q was given in the body of the paper. It thus remains to show that Q cannot be expressed in MDDlog. We make use of the characterization of MDDlog queries in terms of k-colorings provided by Lemma 1.

Assume that m, n are given. Let $k = m^n + 2n$. Define $\mathbf{S}$-instances $\mathfrak{D}_1$ and $\mathfrak{D}_0$ as follows:

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

It is readily checked that $Q(\mathfrak{D}_1) = 1$ and $Q(\mathfrak{D}_0) = 0$, as required. Let $\mathfrak{B}_0$ be an m-coloring of $\mathfrak{D}_0$. Define an m-coloring $\mathfrak{B}_1$ of $\mathfrak{D}_1$ by giving all elements of $\{d_1, \ldots, d_k\}$ exactly the same color as in $\mathfrak{B}_0$. Choose i with $n < i < k - n$ in such a way that for every sequence $d_l, \ldots, d_{l+n}$ with $l > 1$ and $l + n < k$ there exists a sequence $d_{l'}, \ldots, d_{l'+n}$ with $l' > 1$ and $l' + n < k$ such that the coloring of $d_l, \ldots, d_{l+n}$ coincides with the coloring of $d_{l'}, \ldots, d_{l'+n}$

and $i \notin \{l', l'+n\}$. Such an i exists since $k \geq m^n + 2n$. Now give e the color of e_i. One can now easily construct, for every structure corresponding to an n-element subset of $\mathfrak{B}_1$, a homomorphism to $\mathfrak{B}_0$. ❑

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

Proof. We start by describing the translation from frontier-guarded DDlog to (GNFO,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.

Next, we explain how to translate (GNFO, UCQ) to frontier-guarded DDlog. Since every sentence of GF is equivalent to a sentence of GNFO [8], this also yields a translation of (GF,UCQ) to frontier-guarded DDlog. Recall that we used a specific normal form for UNFO sentences. 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 $\mathsf{cl}^\ell_k(\mathcal{O})$ the set of all formulas $\chi(\mathbf{x})$ with $\mathbf{x} = (x_1, \ldots, x_\ell)$ of the form

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

with $\mathbf{y} = (y_1, \ldots, y_m)$, $m + \ell \leq k$, 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 $\chi(\mathbf{z}') \in \mathsf{sub}(\mathcal{O})$.

A *guarded ℓ-type* τ is a subset of $\mathsf{cl}^\ell_k(\mathcal{O})$ that contains at least one atomic relation (possibly equality) containing all variables $x_1, \ldots, x_\ell$, 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 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 relations. 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 $\mathbf{a} = a_1, \ldots, a_n \in \mathsf{adom}(\mathfrak{D})$, 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 a in $\mathfrak{D}'$. Let $\mathfrak{D}''$ be the instance that consists of the atoms in $\mathfrak{D}$ and the atom $P_{\mu(\mathbf{a})}(\mathbf{b})$ 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 $\varphi(\mathbf{x}) \in \mathsf{cl}^\ell_k(\mathcal{O})$ and for each live ℓ-tuple $\mathbf{d}$ of $\mathfrak{D}''$, we have $(\mathsf{dom}'', \mathfrak{D}'') \models \varphi[\mathbf{d}]$ iff $\varphi \in \mu(\mathbf{d})$.

We omit the proofs of $(*)$ and of (ii), as they proceed similarly to the proofs of Theorem 1 and 6. ❑

B. PROOFS FOR SECTION 4

In Section B.1, we start by establishing a central technical result about MMSNP extended with constant symbols which allows us to lift key results from MMSNP sentences to coMMSNP queries (with free variables). Then in Section B.2, we provide the proofs for the results stated in Section 4 of the main paper.

B.1 MMSNP with Constant Symbols

For readability, throughout this subsection, we will adopt a more convenient notation for schemas and structures involving constant symbols. If $\mathbf{S}$ is a schema and $\mathbf{c}$ a (possibly empty) set of constant symbols, then we will use $\mathbf{S}_\mathbf{c}$ as a shorthand for $\mathbf{S} \cup \mathbf{c}$. A $\mathbf{S}_\mathbf{c}$-structure $\mathfrak{B}$ will be given by a pair $(\mathsf{dom}(\mathfrak{B}), \cdot^\mathfrak{B})$, where $\mathsf{dom}(\mathfrak{B})$ is a finite, non-empty set and $\cdot^\mathfrak{B}$ is a function assigning to each n-ary predicate in $\mathbf{S}$ an n-ary relation $P^\mathfrak{B}$ over $\mathsf{dom}(\mathfrak{B})$ and to each constant symbol $c \in \mathbf{c}$ an element $c^\mathfrak{B} \in \mathsf{dom}(\mathfrak{B})$. We use $\mathsf{adom}(\mathfrak{B})$ to denote the active domain of $\mathfrak{B}$, and we call $\mathfrak{B}$ an *active domain structure* if $\mathsf{dom}(\mathfrak{B}) = \mathsf{adom}(\mathfrak{B})$.

Our objective is to establish the following theorem, which lifts the containment and dichotomy results for MMSNP sentences [24] to coMMSNP queries:

Theorem 20 *coMMSNP has a dichotomy between* PTIME *and* CONP *iff the Feder-Vardi conjecture holds. Containment of coMMSNP queries is decidable.*

We prove Theorem 20 in several steps. We consider the language *MMSNP with constant symbols* (abbreviated MMSNP$_c$), consisting of all sentences which can be obtained from MMSNP formulas by replacing each free variable by a constant symbol. The evaluation problem for MMSNP$_c$ consists in deciding whether an MMSNP$_c$ sentence with schema $\mathbf{S}$ and constant symbols $\mathbf{c}$ holds in a given $\mathbf{S_c}$-structure $\mathfrak{B}$. The containment problem for MMSNP$_c$ is to decide for two MMSNP$_c$ sentences Ψ_1, Ψ_2 with relations $\mathbf{S}$ and constants symbols $\mathbf{c}$, whether $\mathfrak{B} \models \Psi_1$ implies $\mathfrak{B} \models \Psi_2$ for all $\mathbf{S_c}$-structures $\mathfrak{B}$. We use $\Psi_1 \subseteq \Psi_2$ to denote containment.

MMSNP$_c$ will serve as a bridge between coMMSNP queries (with free variables) and MMSNP sentences. More precisely, we will first show that evaluation of coMMSNP queries is polynomially equivalent to evaluation of MMSNP$_c$ sentences, and show a polynomial reduction from coMMSNP query containment to containment of MMSNP$_c$ sentences. Afterwards, we will move from MMSNP$_c$ sentences to MMSNP sentences, again showing polynomial equivalence of the evaluation problems and a polynomial reduction for containment.

To link coMMSNP queries and MMSNP$_c$, it will actually prove more convenient to suppose that MMSNP$_c$ sentences are interpreted over active domain structures, whereas to relate MMSNP$_c$ with plain MMSNP, we will wish to work over arbitrary structures. Thus, as a preliminary step, we relate the two variants of the MMSNP$_c$ evaluation and containment problems.

Lemma 4 *The evaluation problem for MMSNP$_c$ restricted to active domain structures is polynomially equivalent to the evaluation problem for MMSNP$_c$ (over general structures).*

Proof. Let $\Phi = \exists X_1 \cdots \exists X_\ell \forall x_1 \cdots \forall x_m \varphi$ be an MMSNP$_c$ sentence over schema $\mathbf{S}$ and constants $\mathbf{c}$, which is interpreted over active domain structures. Pick a fresh second-order variable Y and a fresh constant c not appearing in $\mathbf{c}$. Let φ' be the formula obtained from φ by replacing every conjunct $\psi_1 \rightarrow \psi_2$ of φ by $\psi_1 \rightarrow (\psi_2 \vee Y(c))$. Let χ be the conjunction of all formulas of the form $R(x_1, \ldots, x_k) \rightarrow \neg Y(x_i)$, where R is a k-ary relation in $\mathbf{S}$, and x_i is one of the variables among $x_1, \ldots, x_k$. Define a new MMSNP$_c$ sentence

$$\Phi' = \exists X_1 \cdots \exists X_\ell \exists Y \forall x_1 \cdots \forall x_m (\varphi' \wedge \chi)$$

We claim that the evaluation problem for Φ over active domain structures is polynomially equivalent to the evaluation problem for Φ' over general structures. The first reduction is trivial since for every $\mathbf{S_c}$-structure $\mathfrak{A}$ such that $\mathsf{dom}(\mathfrak{A}) = \mathsf{adom}(\mathfrak{A})$, we have $\mathfrak{A} \models \Phi$ if and only if $\mathfrak{A} \models \Phi'$. To see why, notice that χ ensures that Y is false everywhere on the active domain, so the additional disjuncts have no effect. For the second reduction, we remark that $\mathfrak{B} \models \Phi'$ for a general $\mathbf{S_c}$-structure $\mathfrak{B}$ if and only if $\mathsf{dom}(\mathfrak{B}) \neq \mathsf{adom}(\mathfrak{B})$ (since we can trivially satisfy Φ' by sending c to an element outside the active domain and including that element in Y) or $\mathsf{dom}(\mathfrak{B}) = \mathsf{adom}(\mathfrak{B})$ and $\mathfrak{B} \models \Phi$.

It remains to be shown that every evaluation problem for MMSNP$_c$ over general structures is polynomially equivalent to an evaluation problem for MMSNP$_c$ over active domain structures. Let Φ be an MMSNP$_c$ sentence with schema $\mathbf{S}$ and constant symbols $\mathbf{c}$, and select a fresh monadic second order variable Y, a fresh input relation Elem, and and a fresh constant symbol c. We define Φ' as the sentence over $\mathbf{S} \cup \{\mathsf{Elem}\} \cup \mathbf{c} \cup \{c\}$ obtained from Φ by:

- replacing every conjunct $\psi_1 \rightarrow \psi_2$ by $\psi_1 \wedge \bigwedge_{t \in T} \mathsf{Elem}(t) \rightarrow \psi_2 \vee Y(c)$, where T is the set of terms appearing in $\psi_1 \rightarrow \psi_2$,
- adding a new conjunct $\mathsf{Elem}(x) \rightarrow \neg Y(x)$, and
- adding Y to the initial sequence of existentially quantified monadic second-order variables.

We claim that the evaluation problem for Φ over general structures is polynomially equivalent to the evaluation problem for Φ' over active domain structures. For the first reduction, we have that for every $\mathbf{S_c}$-structure $\mathfrak{B}$, $\mathfrak{B} \models \Phi$ if and only if $\mathfrak{B}' \models \Phi'$, where $\mathfrak{B}'$ extends $\mathfrak{B}$ by setting $\mathsf{Elem}^{\mathfrak{B}'} = \mathsf{dom}(\mathfrak{B})$ and letting $c^{\mathfrak{B}'}$ be any element in $\mathsf{dom}(\mathfrak{B})$. For the other reduction, we have that for every $\mathbf{S} \cup \{\mathsf{Elem}\} \cup \mathbf{c} \cup \{c\}$-structure $\mathfrak{B}$ with $\mathsf{dom}(\mathfrak{B}) = \mathsf{adom}(\mathfrak{B})$, $\mathfrak{B} \models \Phi'$ if and only if either $\mathsf{Elem}^{\mathfrak{B}} \neq \mathsf{dom}(\mathfrak{B})$ or $\mathfrak{B}' \models \Phi$, where $\mathfrak{B}'$ is obtained by taking the $\mathbf{S} \cup \mathbf{c}$-reduct of $\mathfrak{B}$. ❑

Lemma 5 *Containment of MMSNP$_c$ over active domain structures is polynomially reducible to containment of MMSNP$_c$ (over arbitrary structures).*

Proof. Consider MMSNP$_c$ sentences Φ_1, Φ_2 with schema $\mathbf{S}$ and constants $\mathbf{c}$. We apply the construction from the first part of the proof of Lemma 4 to obtain MMNSP$_c$ sentences Φ_1' and Φ_2' with the property that $\mathfrak{B} \models \Phi_i'$ for a general $\mathbf{S_c}$-structure $\mathfrak{B}$ if and only if $\mathsf{dom}(\mathfrak{B}) \neq \mathsf{adom}(\mathfrak{B})$ or $\mathsf{dom}(\mathfrak{B}) = \mathsf{adom}(\mathfrak{B})$ and $\mathfrak{B} \models \Phi_i$ (for $i \in \{1, 2\}$). It is readily verified that $\Phi_1 \subseteq \Phi_2$ for the class of active domain structures if and only if $\Phi_1' \subseteq \Phi_2'$. ❑

By the preceding lemmas, we can choose to work with active domain structures. It is then straightforward to relate the evaluation and containment problems for coMMSNP queries with the corresponding problems for MMSNP$_c$ sentences.

Lemma 6 *The evaluation problem for coMMSNP is polynomially equivalent to the evaluation problem for MMSNP$_c$. Containment of coMMSNP queries is polynomially reducible to containment of MMSNP$_c$ sentences.*

The next step, and the core technical contribution of this subsection, is to relate the evaluation and containment of MMSNP$_c$ sentences to the analogous problems for MMSNP sentences. To simplify the technical constructions, it will prove convenient to work with forbidden pattern problems [38, 34, 12].

We extend forbidden patterns problems to handle constant symbols, by simply substituting $\mathbf{S} \cup \mathbf{c}$-structures for $\mathbf{S}$-structures in Definitions 1 and 2. We denote by FPP$_c$ the class of forbidden patterns problems thus defined, and use FPP to refer to the restriction to structures without constant symbols. Note that both FPP$_c$ and FPP define problems over structures, not instances (although this distinction is irrelevant in the absence of constant symbols).

It was shown in [38] that MMSNP sentences and FPP have the same expressive power. This result can be straightforwardly extended to handle constant symbols:

Lemma 7 *MMSNP$_c$ and FPP$_c$ have the same expressive power (over structures with constant symbols).*

By the previous lemma and the fact that FPP is a subset of FPP$_c$, to show polynomial equivalence of MMSNP$_c$ and MMSNP it suffices to show that every problem in FPP$_c$ is polynomially equivalent to some problem in FPP. To formulate the reductions, we will require some additional notation and terminology, which we introduce next.

Let $\mathbf{S}$ be a schema, $\mathbf{c} = \{c_1, \ldots, c_n\}$ be a set of constant symbols, and $P = \{P_1, \ldots, P_n\}$ be a set of unary predicates which do not appear in $\mathbf{S}$. We will abbreviate $\mathbf{S} \cup P$ to $\mathbf{S}_P$.

We define operations which allow us to transform $\mathbf{S}_P$-structures into $\mathbf{S_c}$-structures, and vice-versa. With every $\mathbf{S}_P$-structure $\mathfrak{B}$ with $P_i^{\mathfrak{B}} \neq \emptyset$ for all $1 \leq i \leq n$, we associate the $\mathbf{S_c}$-structure $\mathfrak{B}^c$, called the *collapse* of $\mathfrak{B}$, by factorizing through the $P_i^{\mathfrak{B}}$. Specifically, let $\sim$ be the smallest equivalence relation such that whenever $d, d' \in P_i^{\mathfrak{B}}$ for some i, then $d \sim d'$. Then $\mathsf{dom}(\mathfrak{B}^c)$ is $\{[d] \mid d \in \Delta^{\mathfrak{B}}\}$, where $[d]$ denotes the equivalence class of d w.r.t. $\sim$. For convenience, when $[d] = \{d\}$, we will use d in place of $[d]$. Set $c_i^{\mathfrak{B}^c} = [d]$, for some $d \in P_i^{\mathfrak{B}}$, and define $R^{\mathfrak{B}^c}$ as follows: $([d],[e]) \in R^{\mathfrak{B}^c}$ if and only if there exist $d' \in [d]$ and $e' \in [e]$ such that $(d', e') \in R^{\mathfrak{B}}$. Note that the mapping $g : d \mapsto [d]$ defines an $\mathbf{S}$-homomorphism from $\mathfrak{B}$ to $\mathfrak{B}^c$, which we call the *canonical homomorphism*.

For a $\mathbf{S_c}$-structure $\mathfrak{A}$, we define the $\mathbf{S}_P$-structure $\hat{\mathfrak{A}}$ which interprets the predicates in $\mathbf{S}$ in the same way as $\mathfrak{A}$ and interprets the predicates in P as follows: $P_i^{\hat{\mathfrak{A}}} = \{c_i^{\mathfrak{A}}\}$. With every $\mathbf{S_c}$-structure $\mathfrak{B}$, one can associate a finite set of finite $\mathbf{S}_P$-structures, $\mathfrak{B}^{ac}$, called its *anti-collapse*, such that the following two properties hold:

1. for all $\mathbf{S}_P$-structures $\mathfrak{A}$:
   $\mathfrak{B} \to \mathfrak{A}^c$ (and $\mathfrak{A}^c$ is defined) if and only if there exists $\mathfrak{B}' \in \mathfrak{B}^{ac}$ such that $\mathfrak{B}' \to \mathfrak{A}$.
2. for all $\mathbf{S_c}$-structures $\mathfrak{A}$:
   $\mathfrak{B} \to \mathfrak{A}$ iff there exists $\mathfrak{B}' \in \mathfrak{B}^{ac}$ such that $\mathfrak{B}' \to \hat{\mathfrak{A}}$.

To employ the anti-collapse $\mathfrak{B}^{ac}$ for the reduction of FPP$_c$ to FPP, we require some properties from the construction of $\mathfrak{B}^{ac}$ (cf. pages 43-45 of [1]). The domain $\Delta^{\mathfrak{B}'}$ of each $\mathfrak{B}' \in \mathfrak{B}^{ac}$ consists of $\Delta^{\mathfrak{B}} \setminus \{c_1^{\mathfrak{B}}, \ldots, c_n^{\mathfrak{B}}\}$ (the *unnamed* individuals in $\mathfrak{B}$) together with the 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 $P_i^{\mathfrak{B}'} = D_i$. Moreover, in Point 1 and Point 2 we have the following more detailed statement:

(1a) if $h : \mathfrak{B} \to \mathfrak{A}^c$ (and $\mathfrak{A}^c$ is defined), and $g : \mathfrak{A} \to \mathfrak{A}^c$ is the canonical 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 $h'(d) \in g^{-1}(c_i^{\mathfrak{A}^c})$ for all $d \in D_i$.

(1b) if $h : \mathfrak{B}' \to \mathfrak{A}$, then $h' : \mathfrak{B} \to \mathfrak{A}^c$ can be defined such that $h'(c_i^{\mathfrak{B}}) = c_i^{\mathfrak{A}^c}$ and $h'(d) = g(h(d))$ if d is not named.

(2b) if $h : \mathfrak{B}' \to \hat{\mathfrak{A}}$, then $h' : \mathfrak{B} \to \mathfrak{A}$ 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 $\mathbf{S}_P$-structures which respects the intuitive meaning of the predicates P_i. A $\mathcal{C}$-coloring $\mathfrak{B}[\mathcal{C}]$ of a $\mathbf{S}_P$-structure $\mathfrak{B}$ is said to be a *uniform $\mathcal{C}$-coloring* of $\mathfrak{B}$ if for every $1 \leq i \leq n$, $d, d' \in P_i^{\mathfrak{B}}$ implies that d and d' have the same color in $\mathfrak{B}[\mathcal{C}]$. Given a set $\mathcal{G}$ of $\mathcal{C}$-colored $\mathbf{S}_P$-structures, we define $\mathsf{Forb}^{un}(\mathcal{G})$ as the set of $\mathbf{S}_P$-structures $\mathfrak{A}$ 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$_c$ to FPP. Suppose that we are given a FPP$_c$ problem defined by the set $\mathcal{F}$ of $\mathcal{C}$-colored $\mathbf{S_c}$-structures (where $\mathcal{C} = \{T_1, \ldots, T_k\}$). We construct a set $\mathcal{G}$ which contains all uniform $\mathcal{C}$-colored $\mathbf{S}_P$-structures $\mathfrak{G}$ such that

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

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

  (Note that we require that in the resulting structure $T_j^{\mathfrak{G}} \cap T_{j'}^{\mathfrak{G}} = \emptyset$ for $j \neq j'$, otherwise $\mathfrak{G}$ is not in $\mathcal{G}$).

It is easy to see that this construction guarantees that every $\mathfrak{G} \in \mathcal{G}$ is such that $P_i^{\mathfrak{G}} \neq \emptyset$ for every $1 \leq i \leq n$.

We let $\mathcal{G}_u = \mathcal{G} \cup \mathcal{U}$, where $\mathcal{U}$ is the set of all $\mathbf{S}_P \cup \mathcal{C}$-structures of the form $\{P_i(d), P_i(e), T_j(d), T_\ell(e)\}$ with $1 \leq i \leq n$ and $1 \leq j < \ell \leq k$.

Notice that $\mathsf{Forb}^{un}(\mathcal{G}) = \mathsf{Forb}(\mathcal{G}_u)$.

Lemma 8 *FPP$_\mathbf{c}$ is polynomially equivalent to FPP. Specifically:*

- *For all $\mathbf{S}_P$-structures $\mathfrak{A}$, $\mathfrak{A} \in \mathsf{Forb}(\mathcal{G}_u)$ iff $\mathfrak{A}^c$ is undefined or $\mathfrak{A}^c \in \mathsf{Forb}(\mathcal{F})$;*
- *For all $\mathbf{S_c}$-structures $\mathfrak{A}$, $\mathfrak{A} \in \mathsf{Forb}(\mathcal{F})$ iff $\hat{\mathfrak{A}} \in \mathsf{Forb}(\mathcal{G}_u)$.*

Proof. First let $\mathfrak{A}$ be a $\mathbf{S}_P$-structure 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 $\mathfrak{A}^c$ is defined (i.e., $P_i^{\mathfrak{A}} \neq \emptyset$ for $1 \leq i \leq n$). We want to show $\mathfrak{A}^c \in \mathsf{Forb}(\mathcal{F})$. By uniformity of $\mathfrak{A}[\mathcal{C}]$, we obtain a $\mathcal{C}$-colored $\mathbf{S_c}$-structure $\mathfrak{A}^c[\mathcal{C}]$ extending $\mathfrak{A}^c$ by setting $d \in T_j^{\mathfrak{A}^c[\mathcal{C}]}$ iff d is unnamed and $d \in T_j^{\mathfrak{A}[\mathcal{C}]}$ or $d = c_i^{\mathfrak{A}^c}$ and $P_i^{\mathfrak{A}[\mathcal{C}]} \subseteq T_j^{\mathfrak{A}[\mathcal{C}]}$. Assume for a contradiction that $h : \mathfrak{F} \to \mathfrak{A}^c[\mathcal{C}]$ for $\mathfrak{F} \in \mathcal{F}$. Then h is a homomorphism from the $\mathbf{S_c}$-reduct $\mathfrak{F}^r$ of $\mathfrak{F}$ to the $\mathbf{S_c}$-reduct $\mathfrak{A}^c$ of $\mathfrak{A}^c[\mathcal{C}]$. By (1a), we find $\mathfrak{F}' \in (\mathfrak{F}^r)^{ac}$ 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 $h'(d) \in g^{-1}(c_i^{\mathfrak{A}^c})$ 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 $P_i^{\mathfrak{F}'} \cap P_j^{\mathfrak{F}'} \neq \emptyset$, which yields $P_i^{\mathfrak{A}} \cap P_j^{\mathfrak{A}} \neq \emptyset$, hence $c_i^{\mathfrak{A}^c} = c_j^{\mathfrak{A}^c}$. It follows that $c_i^{\mathfrak{A}^c}$ and $c_j^{\mathfrak{A}^c}$ have the same colour in $\mathcal{A}^c[\mathcal{C}]$, and thus also in $\mathfrak{F}$, which ensures that each element in $\mathfrak{F}'$ is assigned a unique colour 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 $d \in \mathsf{dom}(\mathfrak{F}')$ and $d \in T_j^{\mathfrak{F}'[\mathcal{C}]}$. If d is unnamed in $\mathfrak{F}$, then $d \in T_j^{\mathfrak{F}'[\mathcal{C}]}$ implies that $d \in T_j^{\mathfrak{F}}$. Hence $h(d) \in T_j^{\mathfrak{A}^c[\mathcal{C}]}$ and $h'(d) \in g^{-1}(h(d)) \subseteq T_j^{\mathfrak{A}[\mathcal{C}]}$. If $d \in D_i$, then $d \in T_j^{\mathfrak{F}'[\mathcal{C}]}$ implies $c_i^{\mathfrak{F}} \in T_j^{\mathfrak{F}}$, hence $c_i^{\mathfrak{A}^c} \in T_j^{\mathfrak{A}^c[\mathcal{C}]}$ and $P_i^{\mathfrak{A}[\mathcal{C}]} \subseteq T_j^{\mathfrak{A}[\mathcal{C}]}$. From $h'(d) \in g^{-1}(c_i^{\mathfrak{A}^c})$, we know that there exists a sequence $A_{\ell_1}, \ldots, A_{\ell_p}$ of predicates from $\{P_1, \ldots, P_n\}$ such that $h'(d) \in A_{\ell_1}^{\mathfrak{A}[\mathcal{C}]}$, $A_{\ell_p} = P_i$, and $A_{\ell_k}^{\mathfrak{A}[\mathcal{C}]} \cap A_{\ell_{k+1}}^{\mathfrak{A}[\mathcal{C}]} \neq \emptyset$ for every $1 \leq k \leq \ell_p$. By uniformity of $\mathfrak{A}[\mathcal{C}]$ and $P_i^{\mathfrak{A}[\mathcal{C}]} \subseteq T_j^{\mathfrak{A}[\mathcal{C}]}$, we obtain $A_{\ell_1}^{\mathfrak{A}[\mathcal{C}]} \subseteq T_j^{\mathfrak{A}[\mathcal{C}]}$, hence $h'(d) \in T_j^{\mathfrak{A}[\mathcal{C}]}$.

Conversely, if $\mathfrak{A}^c$ is undefined, then $\mathfrak{A} \in \mathsf{Forb}(\mathcal{G}_u)$ since $P_i^{\mathfrak{G}} \neq \emptyset$ for all $\mathfrak{G} \in \mathcal{G}$ and $1 \leq i \leq n$, and so any uniform $\mathcal{C}$-coloring of $\mathfrak{A}$ will avoid $\mathfrak{G}_u$. Assume now that $\mathfrak{A}^c \in \mathsf{Forb}(\mathcal{F})$. There exists a $\mathcal{C}$-colored expansion $\mathfrak{A}^c[\mathcal{C}]$ of $\mathfrak{A}^c$ such that there exists no $\mathfrak{F} \in \mathcal{F}$ with $\mathfrak{F} \to \mathfrak{A}^c[\mathcal{C}]$. 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 canonical mapping and set $T_j^{\mathfrak{A}[\mathcal{C}]} = g^{-1}(T_j^{\mathfrak{A}^c[\mathcal{C}]})$, 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 $\mathfrak{F} \in \mathcal{F}$ and some member $\mathfrak{F}'$ of the anti-collapse of the $\mathbf{S_c}$-reduct of $\mathfrak{F}$ 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 $h' : \mathfrak{F}^r \to \mathfrak{A}^c$ that can be defined such that $h'(c_i^{\mathfrak{F}^r}) = c_i^{\mathfrak{A}^c}$ and $h'(d) = g(h(d))$ if d is not named, where $\mathfrak{F}^r$ is the $\mathbf{S_c}$-reduct of $\mathfrak{F}$. We derive a contradiction by showing that h' a homomorphism from $\mathfrak{F}$ to $\mathfrak{A}^c[\mathcal{C}]$. First suppose that $d \in T_j^{\mathfrak{F}}$, and d is unnamed in $\mathfrak{F}$. Then $d \in T_j^{\mathfrak{G}}$, hence $h(d) \in T_j^{\mathfrak{A}[\mathcal{C}]}$. It follows from the definition of $T_j^{\mathfrak{A}[\mathcal{C}]}$ that $h'(d) = g(h(d)) \in T_j^{\mathfrak{A}^c[\mathcal{C}]}$. Next consider the case where $c_i^{\mathfrak{F}} \in T_j^{\mathfrak{F}}$. Then there must exist e such that $e \in T_j^{\mathfrak{G}}$ and $e \in P_i^{\mathfrak{G}}$. It

follows that $h(e) \in T_j^{\mathfrak{A}[\mathcal{C}]}$ and $h(e) \in P_i^{\mathfrak{A}[\mathcal{C}]}$. The definition of $T_j^{\mathfrak{A}[\mathcal{C}]}$ together with $g(h(e)) = c_i^{\mathfrak{A}^c[\mathcal{C}]}$ yields $h'(c_i^{\mathfrak{F}}) = c_i^{\mathfrak{A}^c[\mathcal{C}]} \in T_j^{\mathfrak{A}^c[\mathcal{C}]}$.

The second statement follows easily from the first, since for every $\mathbf{S_c}$-structure $\mathfrak{A}$, we have $\mathfrak{A} = (\hat{\mathfrak{A}})^c$. ❑

Lemma 9 *Containment of FPP$_c$ is polynomially reducible to containment of FPP.*

Proof. Consider $\mathsf{Forb}(\mathcal{F}_1)$ and $\mathsf{Forb}(\mathcal{F}_2)$, both over $\mathbf{S}_c$. Let $\mathcal{G}_{u,1}$ and $\mathcal{G}_{u,2}$ be the corresponding FPPs over schema $\mathbf{S}_P$, which satisfy statements in Lemma 8. 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 a Σ_P-structure 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 $\mathfrak{B}$ be a $\mathbf{S}_c$-structure such that $\mathfrak{B} \in \mathsf{Forb}(\mathcal{F}_1)$. Then applying the previous lemma, we have $\hat{\mathfrak{B}} \in \mathsf{Forb}(\mathcal{G}_{u,1})$, hence $\hat{\mathfrak{B}} \in \mathsf{Forb}(\mathcal{G}_{u,2})$. Again applying the lemma, we obtain $\mathfrak{B} \in \mathsf{Forb}(\mathcal{F}_2)$. ❑

By combining in a straightforward manner Lemmas 4 to 9, we obtain Theorem 20.

B.2 Proofs for Section 4

Theorem 8. *($\mathcal{ALC}$,UCQ) has a dichotomy between* PTIME *and* CONP *iff the Feder-Vardi conjecture holds. The same is true for ($\mathcal{ALCHIU}$,UCQ) and (UNFO,UCQ).*

Proof. Easily obtained by combining Proposition 2 and Theorems 1, 3, 6, and 20. ❑

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

Proof. Here again we straightforwardly combine Proposition 2 and Theorems 1, 3, 6, and 20 ❑

Theorem 11. *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 2 and is omitted. 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 2, 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 1, there is a (GF,UCQ) query q that is not expressible in MDDlog. By Proposition 2, q is not expressible in coMMSNP; by Theorem 7 and the first part of Theorem 11, q is expressible in coGMSNP. ❑

Proposition 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 individual variables: if ψ' is the result of identifying variables in an implication ψ of φ, then ψ is a conjunct of φ (module renaming of individual variables).
- the individual 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

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

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 $\rho_l : \mathbf{x}_l \to \mathbf{z}_l$, $k+1 \leq l \leq n$, 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 individual variable that is not in the range of ρ_l by some fresh individual 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 individual variables in Φ, but may also contain additional individual 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

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

and let ρ be an individual variable assignment 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 $R_i(\pi(\mathbf{x}_i)) \in \mathfrak{D}$.
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

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

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

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

where the $\mathbf{y}'_i$ are defined in the same way as earlier. Let μ be an individual variable assignment satisfying:

- $\mu(x) = \rho(x)$ for x in the image 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 image 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 $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \Phi'$. ❑

C. PROOFS FOR SECTION 5

Theorem 12 *In each case, the following query languages are equally expressive:*

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

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

Proof. Recall that the equivalences between the OBDA languages and fragments of monadic disjunctive datalog have been proved already. Moreover, Point 1 has been proved in the paper. It thus remains to be proved that the following query languages are equally expressive:

(a) ($\mathcal{ALC}$,AQ) and generalized coCSPs with one constant symbol such that all templates are identical except for the interpretation of the constant symbol;

(b) ($\mathcal{ALC}$,BAQ) and coCSP;

(c) ($\mathcal{ALCU}$,BAQ) and generalized coCSP.

We use the notation from the proof of Point 1. In particular, $\mathfrak{B}_T$ denotes the canonical $\mathbf{S}$-structure with domain T. For (a), assume $\mathbf{S}$, $\mathcal{O}$, and $A(x)$ are given, where $\mathcal{O}$ is an $\mathcal{ALC}$-ontology. Let T be the set of all types τ that are realizable for $\mathcal{O}$ and define

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

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

Conversely, assume that $\mathcal{F}$ is a finite set of $\mathbf{S} \cup \{c\}$-structures which coincide except for the interpretation of the constant symbol c, and let $\mathfrak{B}$ be the $\mathbf{S}$-reduct of these structures. Take for every d in the domain $\mathsf{dom}(\mathfrak{B})$ of $\mathfrak{B}$ a fresh concept name A_d, let A be another fresh concept name, and set

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

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

For (b) assume that a query $(\mathbf{S}, \mathcal{O}, \exists x.A(x)) \in (\mathcal{ALC}$,BAQ) is given. We assume w.l.o.g. that $\mathcal{O} \not\models \top \sqsubseteq \exists U.A$ because otherwise we have $q_Q(\mathfrak{D}) = 1$ for all $\mathbf{S}$-instances $\mathfrak{D}$, and so q_Q is trivial. Let T be the set of all types $\tau \subseteq \mathsf{sub}(\mathcal{O})$ that are realized in a model $\mathfrak{A}$ of $\mathcal{O}$ with $\mathfrak{A} \not\models \exists x.A(x)$. Since $\mathcal{O} \not\models \top \sqsubseteq \exists U.A$, the set T is non-empty. One can show that for every $\mathbf{S}$-instance $\mathfrak{D}$: $\mathfrak{D} \to \mathfrak{B}_T$ iff $Q_{\mathbf{S},\mathcal{O},\exists x.A(x)}(\mathfrak{D}) = 0$. Thus, the query defined by $(\mathbf{S}, \mathcal{O}, \exists x.A(x))$ is equivalent to the query defined by $\mathfrak{B}_T$.

Conversely, for a CSP template $\mathcal{B}$ over schema $\mathbf{S}$, we construct an ontology-mediated query $(\mathbf{S}, \mathcal{O}, q)$ as follows. Take for every d in the domain $\mathsf{dom}(\mathcal{B})$ of $\mathcal{B}$ a fresh concept name A_d, let A be

another fresh concept name, and set $q = \exists x.A(x)$ and

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

The query $(\mathbf{S}, \mathcal{O}, \exists x.A(x))$ is equivalent to the query defined by the template $\mathfrak{B}$.

The proof of Point (c) is similar and left to the reader.

❑

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

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

For the reduction, let $n > 0$ and $(\mathfrak{T}, H, V)$ be an instance of the $2^n \times 2^n$-tiling problem with $\mathfrak{T} = \{T_1, \dots, T_p\}$. We construct a schema $\mathbf{S}$, two $\mathcal{ALC}$-ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, and a query $E(x)$ with E a unary relation symbol such that $(\mathfrak{T}, H, V)$ has a solution if and only if $q_{\mathbf{S},\mathcal{O}_1,E(x)} \subseteq q_{\mathbf{S},\mathcal{O}_2,E(x)}$ if and only if $q_{\mathbf{S},\mathcal{O}_1,\exists x.E(x)} \subseteq q_{\mathbf{S},\mathcal{O}_2,\exists x.E(x)}$.

We first define an ontology $\mathcal{G}$ (for grid) which encodes the $2^n \times 2^n$-grid. To define $\mathcal{G}$, we use role names x and y to represent the $2^n \times 2^n$-grid and two binary counters X and Y for counting from 0 to $2^n - 1$. The counters use concept names $X_0, \dots, X_{n-1}, \overline{X}_0, \dots, \overline{X}_{n-1}$ and $Y_0, \dots, Y_{n-1}, \overline{Y}_0, \dots, \overline{Y}_{n-1}$ as their bits, respectively.

$\mathcal{G}$ contains the inclusions

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

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

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

is true. $\mathcal{G}$ contains the following well-known inclusions stating that the value of the counter X is incremented when going to x-successors (and Def is true) and the value of the counter Y is incremented when going to y-successors (and Def is true): for $k = 0, \dots, n-1$,

$$\mathsf{Def} \sqcap \bigsqcap_{j=0..k-1} X_j \sqsubseteq P_k$$

where

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

and

$$\mathsf{Def} \sqcap \bigsqcup_{j=0..k-1} \overline{X}_j \sqsubseteq Q_k$$

where

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

and similarly for Y and y. $\mathcal{G}$ also states that the value of the counter X does not change when going to y-successors and the value of the counter Y does not change when going to x-successors: for $i = 0, \dots, n-1$,

$$\mathsf{Def} \sqcap X_i \sqsubseteq \forall y.(\mathsf{Def} \to X_i), \quad \mathsf{Def} \sqcap \overline{X}_i \sqsubseteq \forall y.(\mathsf{Def} \to \overline{X}_i)$$

and similarly for Y and x. In addition, $\mathcal{G}$ states that when the counter X is $2^n - 1$, there is no x-successor (with Def) and if the counter Y is $2^n - 1$, there is no y-successor (with Def):

$$\mathsf{Def} \sqcap X_0 \sqcap \dots \sqcap X_{n-1} \sqsubseteq \forall x.(\mathsf{Def} \to \bot)$$

and

$$\mathsf{Def} \sqcap Y_0 \sqcap \dots \sqcap Y_{n-1} \sqsubseteq \forall y.(\mathsf{Def} \to \bot)$$

This finishes the definition of $\mathcal{G}$. Define the schema

$$\begin{aligned}\mathbf{S}_{\mathcal{G}} \;=\; & \{x, y, X_0, \dots, X_{n-1}, \overline{X}_0, \dots, \overline{X}_{n-1}\} \cup \\ & \{Y_0, \dots, Y_{n-1}, \overline{Y}_0, \dots, \overline{Y}_{n-1}\}.\end{aligned}$$

We set $\mathcal{O}_2 = \mathcal{G} \cup \{E \sqsubseteq E\}$ (the latter inclusion merely serves to ensure E is part of the schema of $\mathcal{O}_2$).

We now extend $\mathcal{G}$ to another ontology $\mathcal{G}^t$. In addition to the inclusions in $\mathcal{G}$, $\mathcal{G}^t$ states that T_{init} holds at $(0,0)$:

$$\neg X_0 \sqcap \dots \sqcap \neg X_{n-1} \sqcap \neg Y_0 \sqcap \dots \sqcap \neg Y_{n-1} \sqsubseteq T_{\mathsf{init}}$$

and that the tiling is complete on Def:

$$\mathsf{Def} \sqsubseteq \bigsqcup_{i=1..p} T_i,$$

Next, $\mathcal{G}^t$ states that if a tiling condition is violated, then a concept name E is true. For all $i \neq j$:

$$T_i \sqcap T_j \sqsubseteq E,$$

for all $(i,j) \notin H$:

$$T_i \sqcap \exists x.T_j \sqsubseteq E,$$

and for all $(i,j) \notin V$:

$$T_i \sqcap \exists y.T_j \sqsubseteq E.$$

Finally, E is propagated along x and y:

$$\exists x.E \sqsubseteq E, \quad \exists y.E \sqsubseteq E$$

We set $\mathcal{O}_1 = \mathcal{G}^t$ and show:

Claim. The following conditions are equivalent:

1. the $2^n \times 2^n$-tiling problem for $(\mathfrak{T}, H, V)$ has no solution;
2. $q_{\mathbf{S}_{\mathcal{G}},\mathcal{O}_1,E(x)}$ is not contained in $q_{\mathbf{S}_{\mathcal{G}},\mathcal{O}_2,E(x)}$;
3. $q_{\mathbf{S}_{\mathcal{G}},\mathcal{O}_1,\exists x.E(x)}$ is not contained in $q_{\mathbf{S}_{\mathcal{G}},\mathcal{O}_2,\exists x.E(x)}$.

Assume first that $(\mathfrak{T}, H, V)$ admits no $2^n \times 2^n$-tiling. Define a $\mathbf{S}_{\mathcal{G}}$-instance $\mathfrak{D}_{\mathcal{G}}$ as follows. We regard the pairs (i,j) with $i \leq 2^n - 1$ and $j \leq 2^n - 1$ as constants and let

- $x((i,j),(i+1,j)) \in \mathfrak{D}_{\mathcal{G}}$ for $i < 2^n - 1$ and
- $y((i,j),(i,j+1)) \in \mathfrak{D}_{\mathcal{G}}$ for $j < 2^n - 1$.

We also set

- $X_k(i,j) \in \mathfrak{D}_{\mathcal{G}}$ if the kth bit of i is 1,
- $\overline{X}_k(i,j) \in \mathfrak{D}_{\mathcal{G}}$ if the kth bit of i is 0,

- $Y_k(i,j) \in \mathfrak{D}_\mathcal{G}$ if the kth bit of j is 1, and
- $\overline{Y}_k(i,j) \in \mathfrak{D}_\mathcal{G}$ if the kth bit of j is 0.

Then

- $q_{\mathbf{S}_\mathcal{G},\mathcal{O}_2,E(x)}(\mathfrak{D}_\mathcal{G}) = \emptyset$ and
- $q_{\mathbf{S}_\mathcal{G},\mathcal{O}_2,\exists x.E(x)}(\mathfrak{D}_\mathcal{G}) = 0$

since $\mathfrak{D}_\mathcal{G}$ counts correctly, and hence is satisfiable w.r.t. $\mathcal{O}_2$. However, since $(\mathfrak{T}, H, V)$ admits no $2^n \times 2^n$-tiling, it follows that

- $(0,0) \in q_{\mathbf{S}_\mathcal{G},\mathcal{O}_1,E(x)}(\mathfrak{D}_\mathcal{G})$;
- $q_{\mathbf{S}_\mathcal{G},\mathcal{O}_1,\exists x.E(x)}(\mathfrak{D}_\mathcal{G}) = 1$.

We have proved Points 2 and 3.

Conversely, assume that $(\mathfrak{T}, H, V)$ admits a $2^n \times 2^n$-tiling given by $f : \{0,\dots,2^n-1\} \times \{0,\dots,2^n-1\} \to \mathfrak{T}$. We show that $q_{\mathbf{S}_\mathcal{G},\mathcal{O}_1,\exists x.E(x)}(\mathfrak{D}) = 0$ for all $\mathbf{S}_\mathcal{G}$-instances $\mathfrak{D}$ which are satisfiable w.r.t. $\mathcal{O}_2$. Then Points 2 and 3 are refuted, as required.

Assume $\mathfrak{D}$ is satisfiable w.r.t. $\mathcal{O}_2$. We define a model $(\mathsf{dom}, \mathfrak{D}')$ of $\mathcal{O}_1$ with $\mathfrak{D}' \supseteq \mathfrak{D}$ as follows: the domain of $\mathfrak{D}'$ coincides with $\mathsf{adom}(\mathfrak{D})$. Symbols from $\mathbf{S}_\mathcal{G}$ are defined in $\mathfrak{D}'$ in exactly the same way as in $\mathfrak{D}$. To define the facts involving tile types T_k associate with every $d \in \mathsf{adom}(\mathfrak{D})$ such that Def applies to d, the uniquely determined pair $v(d) = (i,j)$ given to the values of the counters X and Y by Def. Then set $T_k(d) \in \mathfrak{D}'$ iff $f(v(d)) = T_k$. Note that $\mathfrak{D}'$ contains no facts involving E. It is readily checked that the resulting structure is a model of $\mathcal{O}_1$. ❑

Proposition 4. *If $Q = (\mathbf{S}, \mathcal{O}, q)$ is an ontology-mediated query with $\mathcal{O}$ formulated in equality-free FO and q a UCQ, then q_Q is preserved by homomorphisms. Consequently, it follows from [43] that if q_Q is FO-rewritable, then q_Q is rewritable into a UCQ (thus into datalog).*

Proof. Let $h : \mathfrak{D}_1 \to \mathfrak{D}_2$ be a homomorphism, and $\mathbf{a}$ a tuple from $\mathsf{adom}(\mathfrak{D}_1)$ such that $\mathbf{a} \in q_Q(\mathfrak{D}_1)$. Furthermore, suppose for the sake of contradiction that $h(\mathbf{a}) \notin q_Q(\mathfrak{D}_2)$. 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 $(\mathsf{dom}_2, \mathfrak{D}_2')$. ❑

The following lemma reduces the problem of deciding FO-rewritability from generalized CSP with constants to generalized CSP without constants.

Lemma 10 *Let $\mathcal{F}$ be a finite set of $\mathbf{S} \cup \mathbf{c}$-structures. The following conditions are equivalent:*

1. *coCSP($\mathcal{F}$) is FO-definable;*
2. *coCSP($\mathcal{F}^c$) is FO-definable;*

Proof. If coCSP($\mathcal{F}^c$) is defined by a first-order sentence φ, then replacing every subformula of the form $P_i(x)$ in φ by $x = c_i$ yields a first-order sentence defining coCSP($\mathcal{F}$).

For the converse, we make use a characterization of FO-definability of generalized coCSPs with constants using finite obstruction sets. Let $\mathcal{F}$ be a finite set of $\mathbf{S} \cup \mathbf{c}$-structures. A set $\mathcal{D}$ of $\mathbf{S} \cup \mathbf{c}$-structures is an *obstruction set for CSP($\mathcal{F}$)* if for all $\mathbf{S} \cup \mathbf{c}$-structures $\mathfrak{D}$ the following conditions are equivalent:

- there exists $\mathfrak{B} \in \mathcal{F}$ such that $\mathfrak{D} \to \mathfrak{B}$;
- there does not exist $\mathfrak{A} \in \mathcal{D}$ such that $\mathfrak{A} \to \mathfrak{D}$.

It is known that, for any finite set of structures $\mathcal{F}$, coCSP($\mathcal{F}$) is FO-definable if and only if $\mathcal{F}$ has a finite obstruction set. This was shown in [2] for structures without constant symbols, and follows easily from results in [43] even for the case of structures with constants. Finally, it was shown in Proposition A.2 (1) in [1] that if coCSP($\mathcal{F}$) has a finite obstruction set, then so does coCSP($\mathcal{F}^c$). ❑

The following lemma reduces the problem of deciding FO-definability from generalized CSP without constants to CSP without constants.

Lemma 11 *Let $\mathcal{F}$ be a finite set of $\mathbf{S} \cup \mathbf{c}$-structures.*

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

Proof. For Point 1 choose for every $\mathfrak{B} \in \mathcal{F}$ a FO-sentence $\varphi_\mathfrak{B}$ such that $(\mathsf{dom}, \mathfrak{D}) \models \varphi_\mathfrak{B}$ iff $\mathfrak{D} \not\to \mathfrak{B}$ for all $\mathbf{S}$-instances $\mathfrak{D}$. Let φ be the conjunction over all $\varphi_\mathfrak{B}$ with $\mathfrak{B} \in \mathcal{F}$. Then $(\mathsf{dom}, \mathfrak{D}) \models \varphi$ iff $\mathfrak{D} \not\to \mathfrak{B}$ for any $\mathfrak{B} \in \mathcal{F}$ holds for all $\mathbf{S}$-instances $\mathfrak{D}$, as required.

To prove the other direction we require the notion of a *critical obstruction*: a $\mathbf{S}$-structure $\mathfrak{A}$ is called a critical obstruction for CSP($\mathcal{G}$) iff $\mathfrak{A} \not\to \mathfrak{B}$ for any $\mathfrak{B} \in \mathcal{G}$ but for any proper substructure $\mathfrak{A}'$ of $\mathfrak{A}$ there exists a $\mathfrak{B} \in \mathcal{F}$ such that $\mathfrak{A}' \to \mathfrak{B}$. It is readily checked that coCSP($\mathcal{G}$) has a finite obstruction set iff there only exist finitely many critical obstructions for CSP($\mathcal{G}$).

For Point 2 assume that all $\mathfrak{B} \in \mathcal{F}$ are mutually homomorphically incomparable and that coCSP($\mathcal{F}$) is FO-definable. Assume for a proof by contradiction that coCSP($\mathfrak{B}_0$) is not FO-definable for some $\mathfrak{B}_0 \in \mathcal{F}$. Then the set $\mathcal{C}$ of critical obstructions for CSP($\mathfrak{B}_0$) is infinite. Let $\mathfrak{B}_0'$ be a substructure of $\mathfrak{B}_0'$ such that no proper substructure of $\mathfrak{B}_0$ can be homomorphically mapped to any $\mathfrak{B} \in \mathcal{F} \setminus \{\mathfrak{B}_0\}$. It is readily checked that the set $\mathcal{C}'$ of disjoint unions $\mathfrak{A} \cup \mathfrak{B}_0'$, $\mathfrak{A} \in \mathcal{C}$, are critical obstructions for CSP($\mathcal{F}$). Thus coCSP($\mathcal{F}$) is not FO-definable and we have derived a contradiction. ❑

Next, we move on the datalog-definability.

Lemma 12 *Let $\mathcal{F}$ be a finite set of $\mathbf{S} \cup \mathbf{c}$-structures.*

1. *If coCSP($\mathfrak{B}^c$) is datalog-definable for all $\mathfrak{B} \in \mathcal{F}$, then coCSP($\mathcal{F}$) is datalog-definable.*
2. *Conversely, if all $\mathfrak{B} \in \mathcal{F}$ are mutually homomorphically incomparable, and coCSP($\mathcal{F}$) is datalog-definable, then each coCSP($\mathfrak{B}^c$), $\mathfrak{B} \in \mathcal{F}$, is datalog-definable.*

Proof. (1) If each coCSP($\mathfrak{B}^c$) is datalog-definable, then, since datalog is closed under conjunction, we also have that coCSP($\mathcal{F}^c$) is datalog-definable. Let Π be a datalog program that defines

coCSP($\mathcal{F}^c$). A datalog program Π' defining coCSP($\mathcal{F}$) may be obtained from Π by replacing every $P_i(x)$ with $x = c_i$.

For (2), we make use of a characterization of datalog-definability in terms of *obstruction sets of bounded treewidth*. Recall from the proof of Lemma 10 the notion of an obstruction set for a set of structures. Suppose that coCSP($\mathcal{F}$) is definable by a datalog program whose rules contain at most k variables. Then $\mathcal{F}$ has an obstruction set of treewidth k, namely, the set of all canonical structures of non-recursive datalog programs obtained by unfolding the given datalog program finitely many times (a standard argument).

We claim that, in fact, each $\mathfrak{B} \in \mathcal{F}$ has an obstruction set of treewidth k. We prove this claim by contraposition: if some $\mathfrak{B} \in \mathcal{F}$ does not have an obstruction set of treewidth at most k, there is a structure $\mathfrak{A}$ such that $\mathfrak{A} \not\to \mathfrak{B}$, while, at the same time, $\mathfrak{B}' \to \mathfrak{A}$ implies $\mathfrak{B}' \to \mathfrak{B}$ for all structures $\mathfrak{B}'$ of treewidth at most k. Now, take $\mathfrak{A}'$ to be the disjoint union of $\mathfrak{A}$ and $\mathfrak{B}$. Then we have that $\mathfrak{A} \not\to \mathcal{F}$ (here, we are using also the fact that $\mathcal{F}$ consists of homomorphically incomparable structures). At the same time, $\mathfrak{B}' \to \mathfrak{A}$ implies $\mathfrak{B}' \to \mathfrak{B}$ for all structures $\mathfrak{B}'$ of treewidth at most k. Therefore, coCSP($\mathcal{F}$) has no obstruction set of bounded treewidth, a contradiction.

So far, we have shown that, for each $\mathfrak{B} \in \mathcal{F}$, coCSP($\mathfrak{B}$) has an obstruction set of bounded tree width. By Proposition A.2 (1) in [1], we have that, for all structures $\mathfrak{A}$ with constant symbols, if coCSP($\mathfrak{A}$) has an obstruction set of bounded treewidth, then coCSP($\mathfrak{A}^c$) has an obstruction set of bounded treewidth too (although it is not explicitly stated, it can easily be verified that the relevant construction used there preserves bounded treewidth). Thus, we obtain that, for each $\mathfrak{B} \in \mathcal{F}$, coCSP($\mathfrak{B}^c$) has an obstruction set of bounded width. It was shown in [24] that, for any structure $\mathfrak{A}$ without constant symbols, coCSP($\mathfrak{A}$) is datalog-definable if and only if $\mathfrak{A}$ has an obstruction set of bounded tree-width. Therefore we have that, for each $\mathfrak{B} \in \mathcal{F}$, coCSP($\mathfrak{B}^c$) is datalog-definable. ❑

The above lemmas, together, establish Proposition 5.

We now proceed with the proof of Theorem 16.

We now give the lower bound proofs for Theorem 16.

Lemma 13 *It is* NEXPTIME*-hard to decide FO-rewritability of queries in (ALC,AQ) and of queries in (ALC,BAQ).*

Proof. We prove the lower bound and employ for the reduction the same tiling problem as in the lower bound proof of Theorem 14. We also employ the ontologies constructed in the proof of Theorem 14.

For the reduction, let $n > 0$ and $(\mathfrak{T}, H, V)$ be an instance of the $2^n \times 2^n$-tiling problem with $\mathfrak{T} = \{T_1, \dots, T_p\}$. We construct a schema $\mathbf{S}$, an $\mathcal{ALC}$-ontology $\mathcal{O}$ and a query $A(x)$ such that $(\mathfrak{T}, H, V)$ has a solution 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 $\mathcal{G}$, its extension $\mathcal{G}^t$, and the schema $\mathbf{S}_\mathcal{G}$ from the proof of Theorem 14. To define $\mathcal{O}$, we take a fresh role name S and two concept names A and F and set

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

and $\mathbf{S} = \mathbf{S}_\mathcal{G} \cup \{S, F\}$.

Claim. The following conditions are equivalent:

- $(\mathfrak{T}, H, V)$ 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.

Assume that $(\mathfrak{T}, H, V)$ admits no $2^n \times 2^n$-tiling. $q_{\mathbf{S},\mathcal{O},A(x)}$ is not FO-rewritable iff there does not exist a finite set $\mathcal{D}$ of $\mathbf{S} \cup \{c\}$-structures (an obstruction set) such that the following conditions are equivalent for every $\mathbf{S}$-instance $\mathfrak{D}$ and $d \in \mathsf{adom}(\mathfrak{D})$:

1. $d \in q_{\mathbf{S},\mathcal{O},A(x)}(\mathfrak{D})$.
2. there exists $\mathfrak{A} \in \mathcal{D}$ such that $(\mathfrak{A}, a) \to (\mathfrak{D}, d)$.

We show that no finite obstruction set exists. To this end, we define $\mathbf{S}$-instances $\mathfrak{D}_m$ as the union of $\mathfrak{D}_\mathcal{G}$ and the facts

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

It is readily 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 $(\mathfrak{T}, H, V)$ has a $2^n \times 2^n$-tiling given by $f : \{0, \dots, 2^n - 1\} \times \{0, \dots, 2^n - 1\} \to \mathfrak{T}$. 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 $d \in \mathsf{adom}(\mathfrak{D})$, $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \models \varphi[d]$ iff $d \in q_{\mathbf{S},\mathcal{O},A(x)}(\mathfrak{D})$.

Note that one can easily construct a first-order sentence $\varphi_\mathcal{G}$ over $\mathbf{S}_\mathcal{G}$ such that, for all $\mathbf{S}_\mathcal{G}$-instances $\mathfrak{D}$, the following are equivalent:

- $\mathfrak{D}$ is not satisfiable w.r.t. $\mathcal{G}$;
- $(\mathsf{adom}, \mathfrak{D}) \models \varphi_\mathcal{G}$.

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

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

The direction from Point 1 to Point 2 is trivial. Conversely, assume that $(\mathsf{adom}(\mathfrak{D}), \mathfrak{D}) \not\models \varphi_\mathcal{G}$. Then $\mathfrak{D}$ is satisfiable w.r.t. $\mathcal{G}$. 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 $\mathfrak{D}$. To define the facts involving tile types T_k, associate with every $d \in \mathsf{adom}(\mathfrak{D})$ such that Def applies to d, the uniquely determined pair $v(d) = (i, j)$ given to the values of the counters X and Y by Def. Then set $T_k(d) \in \mathfrak{D}'$ iff $f(v(d)) = T_k$. Note that $\mathfrak{D}'$ contains no facts involving E or A. It is readily checked that the resulting structure is a model of $\mathcal{O}$, as required. ❑

Lemma 14 *It is* NEXPTIME*-hard to decide datalog-rewritability of queries in (ALC,AQ) and of queries in (ALC,BAQ).*

Proof. The proof is based on a modification of the proof of Lemma 13. For the reduction, let $n > 0$ and $(\mathfrak{T}, H, V)$ be an instance of the $2^n \times 2^n$-tiling problem with $\mathfrak{T} = \{T_1, \dots, T_p\}$. We construct a schema $\mathbf{S}$, an $\mathcal{ALC}$-ontology $\mathcal{O}'$ and a query $A(x)$ such that $(\mathfrak{T}, H, V)$ 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 the ontology $\mathcal{G}$, its extension $\mathcal{G}^t$, and the schema $\mathbf{S}_\mathcal{G}$ from the proof of Theorem 14. To define $\mathcal{O}'$ we take fresh role names S and H and fresh concept names P_1, P_2, P_3 and encode the 3-colorability problem as follows:

$$\begin{aligned}\mathcal{O}' \;=\; & \mathcal{G}^t \cup \{\exists S.E \sqsubseteq E, \exists H.A \sqsubseteq A\} \cup \\ & \{E \sqsubseteq P_1 \sqcup P_2 \sqcup P_3\} \cup \\ & \{P_i \sqcap P_j \sqsubseteq A \mid 1 \le i < j \le 3\} \cup \\ & \{P_i \sqcap \exists H.P_i \sqsubseteq A \mid 1 \le i \le 3\}\end{aligned}$$

and $\mathbf{S} = \mathbf{S}_{\mathcal{G}} \cup \{S, H\}$.

Claim. The following conditions are equivalent:

- $(\mathfrak{T}, H, V)$ 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.

Assume that $(\mathfrak{T}, H, V)$ 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 $\mathfrak{D}_{\mathcal{G}}$ and the facts $S(d,d')$ for all d, d' in G and $H(d,d')$ for every edge $\{d,d'\}$ in G. It is readily checked that

- $(0,0) \in q_{\mathbf{S},\mathcal{O}',A(x)}(\mathfrak{D})$ iff G is not 3-colorable;
- $q_{\mathbf{S},\mathcal{O}',\exists x.A(x)}(\mathfrak{D}) = 1$ iff 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 $(\mathfrak{T}, H, V)$ admits a $2^n \times 2^n$-tiling then one can show datalog-rewritability using exactly the same argument as in the proof of Lemma 13. ❑

We now prove the undecidability results for $\mathcal{ALCF}$. In [10, 36], alternative definitions of query containment and FO-rewritability are employed which consider only instances that are satisfiable w.r.t. 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 $\mathbf{S}$-instance $\mathfrak{D}$ such that $\mathfrak{D}$ is satisfiable w.r.t. $\mathcal{O}_1$. 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 $\mathbf{S}$-instance $\mathfrak{D}$ that are satisfiable w.r.t. $\mathcal{O}$. Undecidability of query containment w.r.t. consistent instances and of FO-rewritability w.r.t. consistent instances were proven respectively in [10] and [36]. 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 21 *Query containment, FO-rewritability, and datalog-rewritability are all undecidable for queries in (ALCF,AQ) and queries in (ALCF,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 $\mathfrak{P} = (\mathfrak{T}, H, V)$ with

- $\mathfrak{T} = \{T_1, \dots, T_p\}$ 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 $\mathfrak{U} \subseteq \mathfrak{T}$ and $\mathfrak{R} \subseteq \mathfrak{T}$ of tile types to be placed on the upper and right borders respectively, satisfying $\mathfrak{U} \cap \mathfrak{R} = \{T_{\mathsf{final}}\}$;
- $H \subseteq \mathfrak{T} \times \mathfrak{T}$ a *horizontal matching relation*; and
- $V \subseteq \mathfrak{T} \times \mathfrak{T}$ a *vertical matching relation*.

A *tiling* for $(\mathfrak{T}, H, V)$ is a map $f : \{0, \dots, n\} \times \{0, \dots, m\} \to \mathfrak{T}$ 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 $1 \leq j < m$.
- $(f(i,j), f(i+1,j)) \in H$ for all $0 \leq i < n$, and
- $(f(i,j), f(i,j+1)) \in v$ for all $0 \leq i < m$.

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

- if $(T_i, T_j) \in H$, 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 $(T_i, T_j) \in V$ or $(T_j, T_i) \in V$;
- if $(T_i, T_j) \in V$, 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 $(T_i, T_j) \in H$ or $(T_j, T_i) \in H$.

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

Fix a particular $\mathfrak{P} = (\mathfrak{T}, H, V)$. For the data schema, we use $\mathbf{S} = \{T_1, \dots, T_p, x, y, x^-, y^-\}$, where $T_1, \dots, T_p$ are treated as concept names, and x, y, x^-, and y^- are role names. We use x and y to specify horizontal and vertical adjacency of points in the rectangle, and the role names x^- and y^- to simulate the inverses of x and y (note that since x^- and y^- are regular role names, they need not be interpreted as the inverses of x and y). We construct an $\mathcal{ALCF}$-ontology $\mathcal{O}_{\mathfrak{P}}$ which asserts functionality of x, y, x^-, y^- and contains inclusions using additional concept names $U, R, Y, I_x, I_y, C, Z_{c,1}, Z_{c,2}, Z_{x,1}, Z_{x,2}, Z_{y,1}$. 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 $e \in \{c, x, y\}$, 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 $(T_i, T_j) \in H$ and $(T_i, T_\ell) \in V$:

$$
\begin{aligned}
T_{\mathsf{final}} &\sqsubseteq Y \sqcap U \sqcap R \\
\exists x.(U \sqcap Y \sqcap T_j) \sqcap I_x \sqcap T_i &\sqsubseteq U \sqcap Y \\
\exists y.(R \sqcap Y \sqcap T_\ell) \sqcap I_y \sqcap T_i &\sqsubseteq R \sqcap Y \\
\exists x.(T_j \sqcap Y \sqcap \exists y.Y) & \\
\quad \sqcap \exists y.(T_\ell \sqcap Y \sqcap \exists x.Y) & \\
\sqcap I_x \sqcap I_y \sqcap C \sqcap T_i &\sqsubseteq Y \\
\exists x.\exists y.\mathcal{B}_c \sqcap \exists y.\exists x.\mathcal{B}_c &\sqsubseteq C \\
\mathcal{B}_x \sqcap \exists x.\exists x^-.\mathcal{B}_x &\sqsubseteq I_x \\
\mathcal{B}_y \sqcap \exists y.\exists y^-.\mathcal{B}_y &\sqsubseteq I_y
\end{aligned}
$$

$$
\begin{aligned}
T_i &\sqsubseteq \forall y.\bot \\
T_j &\sqsubseteq \forall x.\bot \\
U &\sqsubseteq \forall x.U \\
R &\sqsubseteq \forall y.R \\
\bigsqcup_{1 \leq s < t \leq p} T_s \sqcap T_t &\sqsubseteq \bot
\end{aligned}
$$

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 x-y-successor and a y-x-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_x (resp. I_y) is used to ensure that x^- (resp. y^-) act as the inverse of x (resp. y) at all points in the rectangle.

The following property can be obtained by a minor modification of Lemma 30 in [3]:

Lemma 15 *$\mathfrak{P}$ 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 $T_i \sqsubseteq \forall y.\bot$, $T_i \in \mathfrak{U}$, $T_j \sqsubseteq \forall x.\bot$, $T_j \in \mathfrak{R}$, and of $\bigsqcup_{1\leq s<t\leq p} T_s \sqcap T_t \sqsubseteq \bot$. The following is readily checked:

Claim. For all $\mathbf{S}$-instances $\mathfrak{D}$, $(\mathsf{adom}(\mathfrak{D}),\mathfrak{D}) \models \varphi_\mathfrak{P}$ iff $\mathfrak{D}$ is satisfiable w.r.t. $\mathcal{O}_\mathfrak{P}$.

We now prove undecidability of query containment. Let E be a fresh concept name and let

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

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 15, there is a $\mathbf{S}$-instance $\mathfrak{D}$ which is consistent with $\mathcal{O}_\mathfrak{P}$ and such that $q_{\mathbf{S},\mathcal{O}_\mathfrak{P},T_{\mathsf{init}}(x)\wedge Y(x)}(\mathcal{D}) \neq \emptyset$. 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 appears only trivially 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 $\mathbf{S}$-instance which is consistent with $\mathcal{O}_1$. By Lemma 15, $q_{\mathbf{S},\mathcal{O}_\mathfrak{P},T_{\mathsf{init}}(x)\wedge Y(x)}(\mathcal{D}) = \emptyset$, 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 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 15, we can find an $\mathbf{S}$-instance $\mathfrak{D}_\mathfrak{P}$ which 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 Lemma 13 to show that $q_{\mathbf{S},\mathcal{O}_1,E(x)}$ and $q_{\mathbf{S},\mathcal{O}_1,E(x)}$ are not FO-rewritable. Specifically, we construct $\mathbf{S}$-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 is readily 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 $\mathbf{S}$-instance $\mathfrak{D}$, $\mathfrak{D}$ is satisfiable w.r.t. $\mathcal{O}_\mathfrak{P}$ 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 H and fresh concept names P_1,P_2,P_3 and encode the 3-colorability problem as follows:

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

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

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 $\mathbf{S}$-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 Lemma 14. Given a connected undirected graph G, we define an $\mathbf{S}$-instance $\mathfrak{D}$ as the union of $\mathfrak{D}_\mathfrak{P}$ and the facts $S(d,d')$ for all d,d' in G and $H(d,d')$ for every edge $\{d,d'\}$ in G. It is readily 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 $\mathbf{S}$-instance $\mathfrak{D}$, we have that $\mathfrak{D}$ is satisfiable w.r.t. $\mathcal{O}_\mathfrak{P}$ 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. ❑