Rewritability in Monadic Disjunctive Datalog, MMSNP, and Expressive Description Logics[*]

Cristina Feier[†], Antti Kuusisto[†], and Carsten Lutz[†]

† **Fachbereich Informatik, University of Bremen, Germany**
lastname@uni-bremen.de

Abstract

We study rewritability of monadic disjunctive Datalog programs, (the complements of) MMSNP sentences, and ontology-mediated queries (OMQs) based on expressive description logics of the $\mathcal{ALC}$ family and on conjunctive queries. We show that rewritability into FO and into monadic Datalog (MDLog) are decidable, and that rewritability into Datalog is decidable when the original query satisfies a certain condition related to equality. We establish 2NExpTime-completeness for all studied problems except rewritability into MDLog for which there remains a gap between 2NExpTime and 3ExpTime. We also analyze the shape of rewritings, which in the MMSNP case correspond to obstructions, and give a new construction of canonical Datalog programs that is more elementary than existing ones and also applies to non-Boolean queries.

1998 ACM Subject Classification F.4.1, I.2.3, I.2.4, I.1.2, F.2.2, H.2.4, H.2.3

Keywords and phrases FO-Rewritability, MDDLog, MMSNP, DL, Ontology Mediated Queries

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

In data access with ontologies, the premier aim is to answer queries over incomplete and heterogeneous data while taking advantage of the domain knowledge provided by an ontology [10, 17]. Since traditional database systems are often unaware of ontologies, it is common to rewrite the emerging ontology-mediated queries (OMQs) into more standard database query languages. For example, the DL-Lite family of description logics (DLs) was designed specifically so that any OMQ $Q = (\mathcal{T}, \Sigma, q)$ where $\mathcal{T}$ is a DL-Lite ontology, Σ a data signature, and q a conjunctive query, can be rewritten into an equivalent first-order (FO) query that can then be executed using a standard SQL database system [2, 18]. In more expressive ontology languages, it is not guaranteed that for every OMQ there is an equivalent FO query. For example, this is the case for DLs of the $\mathcal{EL}$ and Horn-$\mathcal{ALC}$ families [23, 37], and for DLs of the expressive $\mathcal{ALC}$ family. In many members of the $\mathcal{EL}$ and Horn-$\mathcal{ALC}$ families, however, rewritability into monadic Datalog (MDLog) is guaranteed, thus enabling the use of Datalog engines for query answering. In $\mathcal{ALC}$ and above, not even Datalog-rewritability is generally ensured. Since ontologies emerging from practical applications tend to be structurally simple, though, there is reason to hope that (FO-, MDLog-, and Datalog-) rewritings do exist in many practically relevant cases even when the ontology is formulated in an expressive language. This has in fact been experimentally confirmed for FO-rewritability in the $\mathcal{EL}$ family of DLs [27], and it has led to the implementation of rewriting tools that, although incomplete, are able to compute rewritings in many practical cases [28, 36, 40].

[*] The authors were funded by ERC grant 647289.

Fundamental problems that emerge from this situation are to understand the exact limits of rewritability and to provide (complete) algorithms that decide the rewritability of a given OMQ and that compute a rewriting when it exists. These problems have been adressed in [8, 9, 27] for DLs from the $\mathcal{EL}$ and Horn-$\mathcal{ALC}$ families. For DLs from the $\mathcal{ALC}$ family, first results were obtained in [11] where a connection between OMQs and constraint satisfaction problems (CSPs) was established that was then used to transfer decidability results from CSPs to OMQs. In fact, rewritability is an important topic in CSP (where it would be called definability) as it constitutes a central tool for analyzing the complexity of CSPs [21, 22, 24, 29]. In particular, rewritability of (the complement of) CSPs into FO and into Datalog is NP-complete [4, 19, 29], and rewritability into MDLog is NP-hard and in ExpTime [19]. In [11], these results were used to show that FO- and Datalog-rewritability of OMQs $(\mathcal{T}, \Sigma, q)$ where $\mathcal{T}$ is formulated in $\mathcal{ALC}$ or a moderate extension thereof and q is an atomic query (AQ) of the form $A(x)$ is decidable and, in fact, NExpTime-complete. For MDLog-rewritability, one can show NExpTime-hardness and containment in 2ExpTime.

The aim of this paper is to study the above questions for OMQs where the ontology is formulated in an expressive DL from the $\mathcal{ALC}$ family and where the actual query is a conjunctive query (CQ) or a union of conjunctive queries (UCQ). As observed in [11], transitioning in OMQs from AQs to UCQs corresponds to the transition from CSP to its logical generalization MMSNP introduced by Feder and Vardi [24] and studied, for example, in [12, 31–33]. More precisely, while the OMQ language $(\mathcal{ALC}, \mathrm{AQ})$ that consists of all OMQs $(\mathcal{T}, \Sigma, q)$ where $\mathcal{T}$ is formulated in $\mathcal{ALC}$ and q is an AQ has the same expressive power as the complement of CSP (with multiple templates and a single constant), the OMQ language $(\mathcal{ALC}, \mathrm{UCQ})$ has the same expressive power as the complement of MMSNP (with free variables)—which in turn is a notational variant of monadic disjunctive Datalog (MDDLog). It should be noted, however, that while all these formalisms are equivalent in expressive power, they differ significantly in succinctness [11]; in particular, the best known translation of OMQs into MMSNP/MDDLog involves a double exponential blowup. In contrast to the CSP case, FO-, MDLog-, and Datalog-rewritability of (the complement of) MMSNP sentences was not known to be decidable. In this paper, we establish decidability of FO- and MDLog-rewritability in $(\mathcal{ALC}, \mathrm{UCQ})$ and related OMQ languages, in MDDLog, and the complement of MMSNP. We show that FO-rewritability is 2NExpTime-complete in all three cases, and that MDLog-rewritability is in 3ExpTime; a 2NExpTime lower bound was established in [15]. Let us discuss our results on FO-rewritability from three different perspectives. From the OMQ perspective, the transition from AQs to UCQs results in an increase of complexity from NExpTime to 2NExpTime. From the monadic Datalog perspective, adding disjunction (transitioning from monadic Datalog to MDDLog) results in a moderate increase of complexity from 2ExpTime [6] to 2NExpTime. And from the CSP perspective, the transition from CSPs to MMSNP results in a rather dramatic complexity jump from NP to 2NExpTime.

For Datalog-rewritability, we obtain only partial results. In particular, we show that Datalog-rewritability is decidable and 2NExpTime-complete for MDDLog programs that, in a certain technical sense made precise in the paper, *have equality.* For the general case, we only obtain a potentially incomplete procedure. It is well possible that the procedure is in fact complete, but proving this remains an open issue for now. These results also apply to analogously defined classes of MMSNP sentences and OMQs that have equality.

While we mainly focus on deciding whether a rewriting exists rather than actually computing it, we also analyze the shape that rewritings can take. Since the shape turns out to be rather restricted, this is important information for algorithms (complete or incomplete) that seek to compute rewritings. In the CSP/MMSNP world, this corresponds to analyzing obstruction sets for MMSNP, in the style of CSP obstructions [3, 16, 34] and not to be confused with colored forbidden patterns sometimes used to characterize MMSNP [33]. More precisely, we show that an OMQ $(\mathcal{T}, \Sigma, q)$ from $(\mathcal{ALC}, \text{UCQ})$ is FO-rewritable if and only if it is rewritable into a UCQ in which each CQ has treewidth $(1, \max\{2, n_q\})$, n_q the size of q;[1] similarly, the complement of an MMSNP sentence φ is FO-definable if and only if it admits a finite set of finite obstructions of treewidth $(1, k)$ where k is the diameter of φ (the maximum size of a negated conjunction in its body, in Feder and Vardi's terminology). We also show that $(\mathcal{T}, \Sigma, q)$ is MDLog-rewritable if and only if it is rewritable into an MDLog program of diameter $(1, \max\{2, n_q\})$; similarly, the complement of an MMSNP sentence φ is MDLog-definable if and only if it admits a (potentially infinite) set of finite obstructions of treewidth $(1, k)$ where k is the diameter of φ. For the case of rewriting into unrestricted Datalog, we give a new and direct construction of canonical Datalog-rewritings. It has been observed in [24] that for every CSP and all ℓ, k, it is possible to construct a canonical Datalog program Π of width ℓ and diameter k in the sense that if any such program is a rewriting of the CSP, then so is Π; moreover, even when there is no (ℓ, k)-Datalog rewriting, then Π is the best possible approximation of such a rewriting. The existence of canonical Datalog-rewritings for (the complement of) MMSNP sentences was already known from [13]. However, the construction given there is quite complex, proceeding via an infinite template that is obtained by applying an intricate construction due to Cherlin, Shelah, and Shi [20], which makes them rather hard to analyze. In contrast, our construction is elementary and essentially parallels the CSP case; it also applies to MMSNP formulas with free variables, where the canonical program takes a rather special form that involves *parameters*, similar in spirit to the parameters to least fixed-point operators in FO(LFP) [5].

Our main technical tool is the translation of an MMSNP sentence into a generalized CSP, i.e. a CSP in which there are multiple templates, exhibited by Feder and Vardi in [24]. The translation is not equivalence preserving and involves a double exponential blowup, but was designed so as to preserve complexity up to polynomial time reductions. Here, we are not so much interested in the complexity aspect, but rather in the semantic relationship between the original MMSNP sentence and the constructed CSP. It turns out that the translation does not quite preserve rewritability. In particular, when the original MMSNP sentence has a rewriting, then the natural way of constructing from it a rewriting for the CSP is sound only on instances of high girth. However, FO- and MDLog-rewritings that are sound on high girth (and unconditionally complete) can be converted into rewritings that are unconditionally sound (and complete). The same is true for Datalog-rewritings when the MMSNP sentence has equality, but it remains open whether it is true in the general case.

The structure of this paper is as follows. In Section 2, we give some preliminaries. In Section 3, we summarize the main properties of Feder and Vardi's translation of MMSNP into CSP. This is used in Section 4 to show that FO- and MDLog-rewritability of Boolean MDDLog programs and of the complement of MMSNP sentences is decidable, also establishing the announced complexity results. In Section 5, we analyze the shape of FO- and MDLog-rewritings and of obstructions for MMSNP sentences. In Section 6, we study Datalog-rewritability of MDDLog programs that have equality and construct canonical Datalog programs. Section 7 lifts the results from the Boolean case to the general case. Section 8 introduces OMQs and further lifts our results to this setting. We conclude in Section 9.

A long version of the paper is available at: `http://arxiv.org/abs/1701.02231`.

[1] What we mean here is that q has a tree decomposition in which every bag has at most $\max\{2, n_q\}$ elements and in which neighboring bag overlap in at most one element.

2 Preliminaries

A *schema* is a finite collection $\mathbf{S} = (S_1, \dots, S_k)$ of relation symbols with associated arity. An $\mathbf{S}$-*fact* is an expression of the form $S(a_1, \dots, a_n)$ where $S \in \mathbf{S}$ is an n-ary relation symbol, and $a_1, \dots, a_n$ are elements of some fixed, countably infinite set const of *constants*. For an n-ary relation symbol S, $\mathsf{pos}(S)$ is $\{1, \dots, n\}$. An $\mathbf{S}$-*instance* I is a finite set of $\mathbf{S}$-facts. The *active domain* $\mathsf{dom}(I)$ of I is the set of all constants that occur in a fact in I. For an instance I and a schema $\mathbf{S}$, we write $I|_\mathbf{S}$ to denote the restriction of I to the relations in $\mathbf{S}$.

A *tree decomposition* of an instance I is a pair $(T, (B_v)_{v \in V})$, where $T = (V, E)$ is an undirected tree and $(B_v)_{v \in V}$ is a family of subsets of $\mathsf{dom}(I)$ such that: for all $a \in \mathsf{dom}(I)$, $\{v \in V \mid a \in B_v\}$ is nonempty and connected in T; for every fact $R(a_1, \dots a_r)$ in I, there is a $v \in V$ such that $a_1, \dots, a_r \in B_v$. Unlike in the traditional setup [25], we are interested in two parameters of tree decompositions instead of only one. We call $(T, (B_v)_{v \in V})$ an (ℓ, k)-*tree decomposition* if for all $v, v' \in V$, $|B_v \cap B_{v'}| \leq \ell$ and $|B_v| \leq k$. An instance I *has treewidth* (ℓ, k) if it admits an (ℓ, k)-tree decomposition.

An instance I has a *cycle* of length n if it contains distinct facts $R_0(\mathbf{a}_0), \dots, R_{n-1}(\mathbf{a}_{n-1})$, $\mathbf{a}_i = a_{i,1} \cdots a_{i,m_i}$, and there exist $p_i, p'_i \in \mathsf{pos}(R_i)$, $0 \leq i < n$ such that: $p_i \neq p'_i$ for $1 \leq i \leq n$, and $a_{i,p'_i} = a_{i \oplus 1, p_{i \oplus 1}}$ for $0 \leq i < n$, where $\oplus$ denotes addition modulo n. The *girth* of I is the length of its shortest cycle and ∞ if it has no cycle (in which case we say that I is a *tree*).

A *constraint satisfaction problem (CSP)* is defined by an instance T over a schema $\mathbf{S}_E$, called *template*. The problem associated with T, denoted $\text{CSP}(T)$, is to decide whether an input instance I over $\mathbf{S}_E$ admits a homomorphism to T, denoted $I \to T$. We use $\text{coCSP}(T)$ to denote the complement problem, that is, deciding whether $I \not\to T$. A *generalized CSP* is defined by a set of templates S over the same schema $\mathbf{S}_E$ and asks for a homomorphism from the input I to at least one templates $T \in S$, denoted $I \to S$.

An *MMSNP sentence* θ over schema $\mathbf{S}_E$ has the form $\exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_m \varphi$ with $X_1, \dots, X_n$ monadic second-order variables, $x_1, \dots, x_m$ first-order variables, and φ a conjunction of quantifier-free formulas of the form $\alpha_1 \wedge \cdots \wedge \alpha_n \to \beta_1 \vee \cdots \vee \beta_m$ with $n, m \geq 0$, where each α_i takes the form $X_i(x_j)$ or $R(\mathbf{x})$ with $R \in \mathbf{S}_E$, and each β_i takes the form $X_i(x_j)$. The *diameter* of θ is the maximum number of variables in some implication in φ.

A *conjunctive query (CQ)* takes the form $\exists \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$ where φ is a conjunction of relational atoms and $\mathbf{x}$, $\mathbf{y}$ denote tuples of variables; the equality relation may be used. Whenever convenient, we will confuse a CQ $\exists \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$ with the set of atoms in φ. A *union of conjunctive queries (UCQ)* is a disjunction of CQs with the same free variables.

A *disjunctive Datalog rule* ρ has the form $S_1(\mathbf{x}_1) \vee \cdots \vee S_m(\mathbf{x}_m) \leftarrow R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$ where $n > 0$ and $m \geq 0$. When $m \leq 1$, the rule is a *Datalog rule*. We refer to $S_1(\mathbf{x}_1) \vee \cdots \vee S_m(\mathbf{x}_m)$ as the *head* of ρ, and to $R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$ as the *body*. Every variable that occurs in the head of a rule ρ is required to also occur in the body of ρ. A *(disjunctive) Datalog ((D)DLog) program* Π is a finite set of (disjunctive) Datalog rules with a selected *goal relation* goal that does not occur in rule bodies and appears only in non-disjunctive *goal rules* $\mathsf{goal}(\mathbf{x}) \leftarrow R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n)$. The *arity of* Π is the arity of the goal relation; Π is *Boolean* if it has arity zero. Relation symbols that occur in the head of at least one rule of Π are *intensional (IDB) relations*, and all remaining relation symbols in Π are *extensional (EDB) relations*. relation. A (D)DLog program is called *monadic* or an *M(D)DLog program* if all its IDB relations with the possible exception of goal have arity at most one. The *size* of a DDLog program Π is the number of symbols needed to write it (where relation symbols and variables names count one), its *width* is the maximum arity of non-goal IDB relations used in it, and its *diameter* is the maximum number of variables that occur in a rule in Π.

An (ℓ, k)-*DLog program* is a DLog program of width ℓ and diameter k. Sometimes we omit k and speak of ℓ-*DLog programs*.

For Π an n-ary MDDLog program over schema $\mathbf{S}_E$, an $\mathbf{S}_E$-instance I, and $a_1, \dots, a_n \in \mathsf{dom}(I)$, we write $I \models \Pi(a_1, \dots, a_n)$ if $\Pi \cup I \models \mathsf{goal}(a_1, \dots a_n)$ where variables in all rules of Π are universally quantified and thus Π is a set of first-order (FO) sentences. A query q over $\mathbf{S}_E$ of arity n is:

- *sound for* Π if for all $\mathbf{S}_E$-instances I and $\mathbf{a} \in \mathsf{dom}(I)$, $I \models q(\mathbf{a})$ implies $I \models \Pi(\mathbf{a})$;
- *complete for* Π if for all $\mathbf{S}_E$-instances I and $\mathbf{a} \in \mathsf{dom}(I)$, $I \models \Pi(\mathbf{a})$ implies $I \models q(\mathbf{a})$;
- a *rewriting of* Π if it is sound for Π and complete for Π.

To additionally specify the syntactic shape of q, we speak of a UCQ-rewriting, an MDLog-rewriting, and so on. An *FO-rewriting* takes the form of an FO-query that uses only relations from the EDB schema and possibly equality, but neither constants nor function symbols. We say that Π is Q-*rewritable* if there is a Q-rewriting of Π, for $Q \in \{\text{FO}, \text{UCQ}, \text{MDLog}\}$.

It was shown in [11] that the complement of an MMSNP sentence can be translated into an equivalent Boolean MDDLog program in polynomial time and vice versa; moreover, the transformations preserve diameter and all other parameters relevant for this paper. Thus, we will not explicitly distinguish between Boolean MDDLog and (the complement of) MMSNP.

3 From MDDLog via Simple MDDLog to CSPs

Feder and Vardi show how to translate an MMSNP sentence into a generalized CSP that has the same complexity up to polynomial time reductions [24]. The generalized CSP has a different schema to the original MMSNP sentence and is thus not equivalent to it. We use this translation to reduce rewritability problems for MDDLog to corresponding problems for CSPs. In this section, we sum up the results obtained in [24] that are relevant for our reduction and refer to the long version for more details.

To capture the relation between the schema of an MDDLog program and the generalized CSP constructed from it, we introduce the notion of an aggregation schema. Let $\mathbf{S}_E$ be a schema. A schema $\mathbf{S}'_E$ is a k-*aggregation schema* for $\mathbf{S}_E$ if its relations have the form $R_{q(\mathbf{x})}$ where $q(\mathbf{x})$ is a CQ over $\mathbf{S}_E$ without quantified variables and the arity of $R_{q(\mathbf{x})}$ is identical to the number of variables in $\mathbf{x}$, which is at most k. For I an $\mathbf{S}_E$-instance, its *corresponding* $\mathbf{S}'_E$-*instance* I' consists of all facts $R_{q(\mathbf{x})}(\mathbf{a})$ such that $I \models q(\mathbf{a})$. Conversely, for I' an $\mathbf{S}'_E$-instance, the *corresponding* $\mathbf{S}_E$-*instance* I consists of all facts $S(\mathbf{b})$ such that $R_{q(\mathbf{x})}(\mathbf{a}) \in I'$ and $S(\mathbf{b})$ is a conjunct of $q(\mathbf{a})$.

▶ **Example 1.** Let $\mathbf{S}_E = \{r\}$, r a binary relation, $q(\mathbf{x}) = r(x_1, x_2) \wedge r(x_2, x_3) \wedge r(x_3, x_1)$ where $\mathbf{x} = (x_1, x_2, x_3)$, and $\mathbf{S}'_E = \{R_{q(\mathbf{x})}\}$. If I' is the $\mathbf{S}'_E$-instance $\{R_q(a, a', c'), R_q(b, b', a'), R_q(c, c', b')\}$, then its corresponding $\mathbf{S}_E$-instance I is $\{r(a, a'), r(a', c'), r(c', a), r(b, b'), r(b', a'), r(a', b), r(c, c'), r(c', b), r(b', c)\}$. Note that the $\mathbf{S}'_E$-instance I'' that corresponds to I is a strict superset of I': it contains additional facts such as $R_q(c', b', a')$, see Figure 1.

The translation in [24] consists of two steps. The first step is to transform the given Boolean MDDLog program Π into a Boolean MDDLog program Π_S over a suitable aggregation schema $\mathbf{S}'_E$. Π_S is of a restricted syntactic form, called *simple*, which means that each of its rules contains at most one EDB atom, that this atom contains all variables of the rule body, each variable exactly once, and that rules without an EDB atom contain at most a single variable. To achieve this, Π is first saturated by adding all rules that can be obtained from a rule in Π by identifying variables; then Π is rewritten in an equivalence-preserving way so

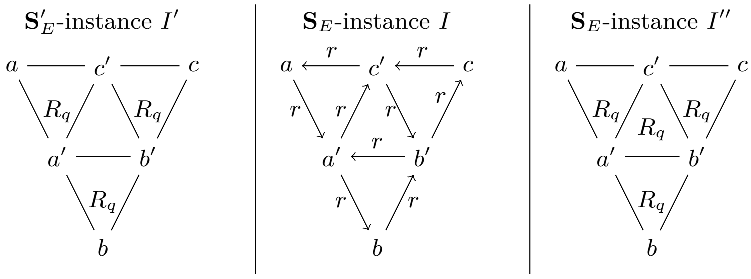
$\mathbf{S}'_E$-instance I'
a
c'
c
R_q
a'
b'
b
$\mathbf{S}_E$-instance I
r
$\mathbf{S}_E$-instance I''

Figure 1 Translating an $\mathbf{S}'_E$-instance into an $\mathbf{S}_E$-instance and vice versa

that all rule bodies are biconnected, introducing fresh unary and nullary IDBs as needed. Finally, for each rule body the conjunction $q(\mathbf{x})$ of its EDB atoms is replaced with a single EDB atom $R_{q(\mathbf{x})}(\mathbf{x})$, additionally taking care of interactions between the new EDB relations that arise, e.g. when we have two relations $R_{q(\mathbf{x})}$ and $R_{p(\mathbf{x})}$ such that $q(\mathbf{x})$ is contained in $p(\mathbf{x})$ (in the sense of query containment).

▶ **Theorem 2** ([24]). *Given a Boolean MDDLog program Π over EDB schema $\mathbf{S}_E$ of diameter k and size n, one can construct a simple Boolean MDDLog program Π_S over a k-aggregation schema $\mathbf{S}'_E$ for $\mathbf{S}_E$ such that*

1. *If I is an $\mathbf{S}_E$-instance and I' the corresponding $\mathbf{S}'_E$-instance, then $I \models \Pi$ iff $I' \models \Pi_S$;*
2. *If I' is an $\mathbf{S}'_E$-instance and I the corresponding $\mathbf{S}_E$-instance, then*
   - **a.** *$I' \models \Pi_S$ implies $I \models \Pi$;*
   - **b.** *$I \models \Pi$ implies $I' \models \Pi_S$ if the girth of I' exceeds k.*

The size of Π_s and the cardinality of $\mathbf{S}'_E$ are bounded by $2^{p(k \cdot \log n)}$, p a polynomial. The construction takes time polynomial in the size of Π_S.

Note that Π_S is equivalent to Π only on instances whose girth exceeds k, the maximal arity of a relation symbol in $\mathbf{S}'_E$.

In the second step, the simple MDDLog program Π_S is translated into a generalized CSP whose complement is equivalent to Π_S. Informally, one introduces one template for every 0-type (set of nullary IDBs), each template contains one constant for every 1-type (set of at most unary IDBs) that is compatible with the 0-type and interprets the EDB relations in a maximal way so that all rules in Π are satisfied (when interpreting the IDBs true as suggested by the 1-types).

▶ **Theorem 3** ([24]). *Let Π be a simple Boolean MDDLog program over EDB schema $\mathbf{S}_E$ and with IDB schema $\mathbf{S}_I$, m the maximum arity of relations in $\mathbf{S}_E$. Then there exists a set of templates S_Π over $\mathbf{S}_E$ such that*

1. *Π is equivalent to coCSP(S_Π);*
2. *$|S_\Pi| \leq 2^{|\mathbf{S}_I|}$ and $|T| \leq |\mathbf{S}_E| \cdot 2^{m|\mathbf{S}_I|}$ for each $T \in S_\Pi$;*

The construction takes time polynomial in $\sum_{T \in S_\Pi} |T|$.

4 FO- and MDLog-Rewritability of Boolean MDDLog Programs

We exploit the translation described in the previous section to lift the decidability of FO-rewritability and of MDLog-rewritability from coCSPs to Boolean MDDLog, and thus also to

MMSNP. In the case of FO, we obtain tight 2NExpTime complexity bounds. For MDLog, the exact complexity remains open (as in the CSP case), between 2NExpTime and 3ExpTime.

We start with observing that FO-rewritability and MDLog-rewritability are more closely related than one might think at first glance. In fact, every MDLog-rewriting can be viewed as an infinitary UCQ-rewriting and, by Rossman's homomorphism preservation theorem [38], FO-rewritability of a Boolean MDDLog program coincides with (finitary) UCQ-rewritability. The latter is true also in the non-Boolean case.

▶ **Proposition 4.** *An MDDLog program Π is FO-rewritable iff it is UCQ-rewritable.*

For utilizing the translation of Boolean MDDLog programs to generalized CSPs in the intended way, the interesting aspect is to deal with the translation of a Boolean MDDLog program Π into a simple program Π_S stated in Theorem 2, since it is not equivalence preserving. The following lemma relates rewritings of Π to rewritings of Π_S.

▶ **Lemma 5.** *Let Π be a Boolean MDDLog program of diameter k, Π_S as in Theorem 2, and $\mathcal{Q} \in \{UCQ, MDLog, DLog\}$. Then*

1. *every $\mathcal{Q}$-rewriting of Π_S can effectively be converted into a $\mathcal{Q}$-rewriting of Π;*
2. *every $\mathcal{Q}$-rewriting of Π can effectively be converted into a $\mathcal{Q}$-rewriting of Π_S that is (i) sound on instances of girth exceeding k and (ii) complete.*

Proof.(sketch) We only give the constructions for the case $\mathcal{Q} = \text{UCQ}$ and refer to the long version for the other cases and for correctness proofs. For Point 1, let q_{Π_S} be a UCQ-rewriting of Π_S. Then we obtain a UCQ-rewriting of Π by replacing every atom $R_{q(\mathbf{x})}(\mathbf{y})$ with $q[\mathbf{y}/\mathbf{x}]$, that is, with the result of replacing the variables $\mathbf{x}$ in $q(\mathbf{x})$ with the variables $\mathbf{y}$.

For Point 2, let q_Π be a UCQ-rewriting of Π. We obtain a UCQ-rewriting of Π_S by taking the UCQ that consists of all CQs which can be obtained as follows:

1. choose a CQ $q(\mathbf{x})$ from q_Π, identify variables in q to obtain a CQ $q'(\mathbf{x}')$, and choose a partition $q_1(\mathbf{x}_1), \ldots, q_n(\mathbf{x}_n)$ of $q'(\mathbf{x}')$;
2. for each $i \in \{1, \ldots, n\}$, choose a relation $R_{p(\mathbf{z})}$ from the EDB schema of Π_S and a vector $\mathbf{y}$ of $|\mathbf{z}|$ variables (repeated occurrences allowed) that are either from $\mathbf{x}_i$ or do not occur in $\mathbf{x}'$ such that $q_i(\mathbf{x}_i) \subseteq p[\mathbf{y}/\mathbf{z}]$; then replace $q_i(\mathbf{x}_i)$ in $q'(\mathbf{x}')$ with the single atom $R_{p(\mathbf{z})}(\mathbf{y})$.

❏

Point 2 of Lemma 5 only yields a rewriting of Π_S on $\mathbf{S}'_E$-instances of high girth. We next show that for CSPs, the existence of a $\mathcal{Q}$-rewriting on instances of high girth, $\mathcal{Q} \in \{\text{UCQ}, \text{MDLog}\}$, implies the existence of a $\mathcal{Q}$-rewriting that works on instances of unrestricted girth. Whether the same is true for $\mathcal{Q} = \text{Datalog}$ remains as an open problem.

▶ **Lemma 6.** *Let S be a set of templates over schema $\mathbf{S}_E$, $g \geq 0$, and $\mathcal{Q} \in \{UCQ, MDLog\}$. If coCSP($S$) is $\mathcal{Q}$-rewritable on instances of girth exceeding g, then it is $\mathcal{Q}$-rewritable.*

The proof of Lemma 6 uses a well-known combinatorial lemma that goes back to Erdös and was adapted to CSPs by Feder and Vardi.Putting together Theorem 2 and 3, Proposition 4, and Lemmas 5 and 6, we obtain the following reductions of rewritability of Boolean MDDLog programs to CSP rewritability.

▶ **Proposition 7.** *Every Boolean MDDLog program Π can be converted into a set of templates S_Π such that*

1. *Π is $\mathcal{Q}$-rewritable iff S_Π is $\mathcal{Q}$-rewritable for every $\mathcal{Q} \in \{FO, UCQ, MDLog\}$;*

2. *every $\mathcal{Q}$-rewriting of* Π *can be effectively translated into a $\mathcal{Q}$-rewriting of* S_Π *and vice versa, for every* $\mathcal{Q} \in \{UCQ, MDLog\}$*.*
3. $|S_\Pi| \leq 2^{2^{p(n)}}$ *and* $|T| \leq 2^{2^{p(n)}}$ *for each* $T \in S_\Pi$*,* n *the size of* Π *and* p *a polynomial.*

The construction takes time polynomial in $\sum_{T \in S_\Pi} |T|$*.*

FO-rewritability of CSPs is NP-complete [29] and it was observed in [11] that the upper bound lifts to generalized CSPs. MDLog-rewritability of coCSPs is NP-hard and in ExpTime [19]. We show in the long version that this upper bound lifts to generalized coCSPs. Together with Proposition 7 and the lower bounds from [15], we obtain the following:

▶ **Theorem 8.** *For Boolean MDDLog programs and the complement of MMSNP sentences,*

1. *FO-rewritability (equivalently: UCQ-rewritability) is 2*NExpTime*-complete;*
2. *MDLog-rewritability is in 3*ExpTime *(and 2*NExpTime*-hard).*

5 Shape of Rewritings and Obstructions for MMSNP sentences

An important first step towards the design of practical algorithms that compute rewritings (when they exist) is to analyze the shape of the rewritings. In the case of CSPs, both UCQ- and MDLog-rewritings are known to be of a rather restricted shape, far from exploiting the full expressive power of the target languages: any FO-rewritable CSP has a UCQ-rewriting that consists of tree-shaped CQs and any MDLog-rewritable CSP has an MDLog-rewriting in which each rule has at mosrt a single EDB atom. In this section, we establish corresponding results for Boolean MDDLog.

Exploiting the results concerning the shape of UCQ- and MDLog- rewritings for CSPs and the constructions from the proof of Point 2 of Lemma 5, one can show the following.

▶ **Theorem 9.** *Let* Π *be a Boolean MDDLog program of diameter* k*. Then*

1. *if* Π *is FO-rewritable, then it has a UCQ-rewriting in which each CQ has treewidth* $(1, k)$*;*
2. *if* Π *is MDLog-rewritable, then it has an MDLog-rewriting of diameter* k*.*

In a sense, the concrete bound k in Points 1 and 2 of Theorem 9 is quite remarkable. Point 2 says, for example, that when eliminating disjunction from a Boolean MDDLog program, it never pays off to increase the diameter.

For CSPs, FO- and MDLog-rewritability is closely related to the theory of obstructions: an *obstruction set* $\mathcal{O}$ for a CSP template T over schema $\mathbf{S}_E$ is a set of instances over the same schema, called *obstructions*, such that for any $\mathbf{S}_E$-instance I, we have $I \not\to T$ iff $O \to I$ for some $O \in \mathcal{O}$. A lot is known about CSP obstructions. For example, T is FO-rewritable if and only if it has a finite obstruction set [3] if and only if it has a finite obstruction set that consists of finite trees [35], and T is MDLog-rewritable if and only if it has a (potentially infinite) obstruction set that consists of finite trees [24].

Here we establish similar results for the case of MMSNP. Obstruction sets for MMSNP are defined in the obvious way: an *obstruction set* $\mathcal{O}$ for an MMSNP sentence θ over schema $\mathbf{S}_E$ is a set of instances over the same schema such that for any $\mathbf{S}_E$-instance I, we have $I \not\models \theta$ iff $O \to I$ for some $O \in \mathcal{O}$. The following result, which essentially is a consequence of Point 1 of Theorem 9, characterizes FO-rewritability of MMSNP sentences in terms of obstruction sets.

▶ **Corollary 10.** *For every MMSNP sentence* θ*, the following are equivalent:*

1. θ *is FO-rewritable;*

2. *θ has a finite obstruction set;*
3. *θ has a finite set of finite obstructions of treewidth $(1, k)$.*

We now turn to MDLog-rewritability.

▶ **Proposition 11.** *Let θ be an MMSNP sentence of diameter k. Then $\neg\theta$ is MDLog-rewritable iff θ has a set of obstructions (equivalently: finite obstructions) that are of treewidth $(1, k)$.*

We remark that the results in [13] almost give Proposition 11, but do not seem to deliver a concrete bound on the parameter k of the treewidth of obstruction sets.

6 Datalog-Rewritability of Boolean MDDLog Programs and Canonical Datalog Programs

We study the Datalog-rewritability of Boolean MDDLog programs. In contrast to the case of FO- or MDLog-rewritings, we obtain a procedure that is sound, but whose completeness remains an open problem. We can show, however, that the procedure is complete for MDDLog programs that have equality, a condition that is defined in detail below. We also give a new and direct construction of canonical Datalog-rewritings of Boolean MDDLog programs (equivalently: the complements of MMSNP sentences), bypassing the construction of infinite templates which involves the application of a non-trivial construction due to Cherlin, Shelah, and Shi [13, 20].

6.1 Datalog-Rewritability of Boolean MDDLog Programs

We say that an MDDLog program Π *has equality* if its EDB schema includes the distinguished binary relation eq, Π contains the rules $P(x) \wedge \mathsf{eq}(x, y) \rightarrow P(y)$ and $P(y) \wedge \mathsf{eq}(x, y) \rightarrow P(x)$ for each IDB relation P, and these are the only rules that mention eq. For an MDDLog program Π that does not have equality, we use $\Pi^=$ to denote the extension of Π with the fresh EDB relation eq and the above rules. If Π has equality, then $\Pi^=$ simply denotes Π. Clearly, a DLog-rewriting of $\Pi^=$ can be converted into a DLog-rewriting of Π by dropping all rules that use the relation eq. This gives the following lemma.

▶ **Lemma 12.** *For any MDDLog program Π, DLog-rewritability of $\Pi^=$ implies DLog-rewitability of Π.*

It remains an interesting open question whether the converse of Lemma 12 holds.

A CSP template T *has equality* if its schema includes the distinguished binary relation eq and T interprets eq as the relation $\{(a, a) \mid a \in \mathsf{dom}(T)\}$. It can be verified that when an MDDLog program that has equality is converted into a generalized CSP based on a set of templates S_Π according to Theorems 2 and 3 (using the concrete constructions in the long version), then all templates in S_Π have equality. The interesting aspect of having equality is that it allows us to establish a counterpart of Lemma 6 also for Datalog-rewritability.

▶ **Lemma 13.** *Let S be a set of templates over schema $\mathbf{S}_E$ that have equality, and let $g \geq 0$. If coCSP(S) is DLog-rewritable on instances of girth exceeding g, then it is DLog-rewritable.*

Proof.(sketch) With every $\mathbf{S}_E$-instance I and $g \geq 0$, we associate an $\mathbf{S}_E$-instance I^g of girth exceeding g such that for any template T over $\mathbf{S}_E$ that has equality, $I^g \rightarrow T$ iff $I \rightarrow T$. In fact, I^g is obtained from I by duplicating domain elements, and introducing chains of equality atoms. Then, for every DLog rewriting Γ of coCSP(S) on instances of girth exceeding g, it is possible to construct a DLog program Γ' such that $I \models \Gamma'$ iff $I^g \models \Gamma$. Intuitively, when executed over I, Γ' mimics the execution of Γ over I^g. Clearly, Γ' is then a DLog-rewriting of coCSP(S) on unrestricted instances. ❏

DLog-rewritability of CSPs is NP-complete [4,19] and it was observed in [11] that this result lifts to generalized CSPs. It thus follows from Theorems 2 and 3 and Lemma 13 that DLog-rewritability of Boolean MDDLog programs that have equality is decidable in 2NExpTime. It is straightforward to verify that the 2NExpTime lower bound for DLog-rewritability of MDDLog programs from [15] applies also to programs that have equality.

▶ **Theorem 14.** *For Boolean MDDLog programs that have equality, DLog-rewritability is 2*NExpTime*-complete.*

MDDLog programs obtained from OMQs typically do not have equality. Due to Lemma 12, though, we obtain a sound but possibly incomplete algorithm for deciding DLog-rewritability of an unrestricted MDDLog program Π by first replacing it with $\Pi^=$ and then deciding DLog-rewritability as per Theorem 14. We speculate that this algorithm is actually complete. Note that for CSPs, it is known that adding equality preseves DLog-rewritability [30], and completeness of our algorithm is equivalent to an analogous result holding for MDDLog.

6.2 Canonical Datalog-Rewritings

For constructing actual DLog-rewritings instead of only deciding their existence, *canonical Datalog programs* play an important role. Feder and Vardi show that for every CSP template T and all $\ell, k > 0$, once can construct an (ℓ, k)-Datalog program that is canonical for T in the sense that if there is any (ℓ, k)-Datalog program which is equivalent to the complement of T, then the canonical one is [24]. In this section, we show that there are similarly simple canonical Datalog programs for Boolean MDDLog. Note that the existence of canonical Datalog programs for MMSNP (and thus for Boolean MDDLog) is already known from [13]. However, the construction given there is rather complex, proceeding via an infinite template and exploiting that it is ω-categorial. This makes it hard to analyze the exact structure and size of the resulting canonical programs. Here, we define canonical Datalog programs for Boolean MDDLog programs in a more elementary way.

Let $0 \leq \ell < k$, and let Π be a Boolean MDDLog program over EDB schema $\mathbf{S}_E$ and with IDB relations from $\mathbf{S}_I$. We first convert Π into a DDLog program Π' that is equivalent to Π on instances of treewidth (ℓ, k). Unlike Π, the new program Π' is no longer monadic. We start with a preliminary. With every DDLog rule $p(\mathbf{y}) \leftarrow q(\mathbf{x})$ where $q(\mathbf{x})$ is of treewidth (ℓ, k) and every (ℓ, k)-tree decomposition $(T, (B_v)_{v \in V})$ of $q(\mathbf{x})$, we associate a set of *rewritten rules* constructed as follows. Choose a root v_0 of T, thus inducing a direction on the undirected tree T. We write $v \prec v'$ if v' is a successor of v in T and use $\mathbf{x}_{v'}$ to denote $B_v \cap B_{v'}$. For all $v \in V \setminus \{v_0\}$ such that $|\mathbf{x}_v| = m$, introduce a fresh m-ary IDB relation Q_v; note that $m \leq \ell$. Now, the set of rewritten rules contains one rule for each $v \in V$. For $v \neq v_0$, the rule is

$$p_v(\mathbf{y}_v) \vee Q_v(\mathbf{x}_v) \leftarrow q(\mathbf{x})|_{B_v} \wedge \bigwedge_{v \prec v'} Q_{v'}(\mathbf{x}_{v'})$$

where $p_v(\mathbf{y}_v)$ is the sub-disjunction of $p(\mathbf{y})$ that contains all disjuncts $P(\mathbf{z})$ with $\mathbf{z} \subseteq B_v$. For v_0, we include the same rule, but use only $p_v(\mathbf{y}_v)$ as the head. The set of rewritten rules associated with $p(\mathbf{y}) \leftarrow q(\mathbf{x})$ is obtained by taking the union of the rewritten rules associated with $p(\mathbf{y}) \leftarrow q(\mathbf{x})$ and any $(T, (B_v)_{v \in V})$.

The DDLog program Π' is constructed from Π as follows:

1. first extend Π with all rules that can be obtained from a rule in Π by identifying variables;

2. then delete all rules with $q(\mathbf{x})$ not of treewidth (ℓ, k) and replace every rule $p(\mathbf{y}) \leftarrow q(\mathbf{x})$ with $q(\mathbf{x})$ of treewidth (ℓ, k) with the rewritten rules associated with it.

It can be verified that Π' satisfies the following conditions:

(I) Π' is sound for Π, that is, for all $\mathbf{S}_E$-instances I, $I \models \Pi'$ implies $I \models \Pi$;

(II) Π' is complete for Π on $\mathbf{S}_E$-instances of treewidth (ℓ, k), that is, for all such instances I, $I \models \Pi$ implies $I \models \Pi'$.

Let $\mathbf{S}'_I$ denote the additional IDB relations in Π'. We now construct the canonical (ℓ, k)-DLog program Γ^c for Π. Fix constants $a_1, \dots, a_\ell$. For $\ell' \leq \ell$, we use $\mathfrak{I}_{\ell'}$ to denote the set of all $\mathbf{S}_I \cup \mathbf{S}'_I$-instances with domain $\mathbf{a}_{\ell'} := a_1, \dots, a_{\ell'}$. The program uses ℓ'-ary IDB relations P_M, for all $\ell' \leq \ell$ and all $M \subseteq \mathfrak{I}_{\ell'}$. It contains all rules $q(\mathbf{x}) \rightarrow P_M(\mathbf{y})$, $M \subseteq \mathfrak{I}_{\ell'}$, that satisfy the following conditions:

1. $q(\mathbf{x})$ contains at most k variables;
2. for every extension J of the $\mathbf{S}_E$-instance $I_q|_{\mathbf{S}_E}$ with $\mathbf{S}_I \cup \mathbf{S}'_I$-facts such that

   a. J satisfies all rules of Π' and does not contain $\mathsf{goal}()$ and
   b. for each $P_N(\mathbf{z}) \in q$, $N \subseteq \mathfrak{I}_{\ell''}$, there is an $L \in N$ such that $L[\mathbf{z}/\mathbf{a}_{\ell''}] = J|_{\mathbf{S}_I \cup \mathbf{S}'_I, \mathbf{z}}$

   there is an $L \in M$ such that $L[\mathbf{y}/\mathbf{a}_{\ell'}] = J|_{\mathbf{S}_I \cup \mathbf{S}'_I, \mathbf{y}}$

where $L[\mathbf{x}/\mathbf{a}]$ denotes the result of replacing the constants in $\mathbf{a}$ with the variables in $\mathbf{x}$ (possibly resulting in identifications) and where $J|_{\mathbf{S}_I \cup \mathbf{S}'_I, \mathbf{x}}$ denotes the simultaneous restriction of J to schema $\mathbf{S}_I \cup \mathbf{S}'_I$ and constants $\mathbf{x}$.[2] We also include all rules of the form $P_\emptyset(\mathbf{x}) \rightarrow \mathsf{goal}()$, $P_\emptyset$ of any arity from 0 to ℓ.

The following theorem says that the canonical program is indeed canonical in the desired sense. For two Boolean DLog programs Π_1, Π_2 over the same EDB schema $\mathbf{S}_E$, we write $\Pi_1 \subseteq \Pi_2$ if for every $\mathbf{S}_E$-instance I, $I \models \Pi_1$ implies $I \models \Pi_2$.

▶ **Theorem 15.** *Let Π be a Boolean MDDLog program, $0 \leq \ell \leq k$, and Γ^c the canonical (ℓ, k)-DLog program for Π. Then*

1. *$\Gamma \subseteq \Gamma^c$ for every (ℓ, k)-DLog program Γ that is sound for Π;*
2. *Π is (ℓ, k)-DLog-rewritable iff Γ^c is a DLog-rewriting of Π.*

Note that by Point 2 of Theorem 15, the canonical (ℓ, k)-DLog program for an MDDLog program Π is interesting even if Π is not rewritable into an (ℓ, k)-DLog program as it is the strongest sound (ℓ, k)-DLog approximation of Π.

7 Non-Boolean MDDLog Programs

We lift the results about the complexity of rewritability, about canonical DLog programs, and about the shape of rewritings and obstructions from the case of Boolean MDDLog programs to the non-Boolean case. For all of this, a certain extension of (ℓ, k)-Datalog programs with parameters plays a central role. We thus begin by introducing these extended programs.

[2] We could additionally demand that M is minimal so that Condition 2 is satisfied, but this is not strictly required.

7.1 Deciding Rewritability

An (ℓ, k)-*Datalog program with n parameters* is an n-ary $(\ell + n, k + n)$-Datalog program in which all IDBs have arity at least n and where in every rule, all IDB atoms agree on the variables used in the last n positions (both in rule bodies and heads and including the goal IDB). The last n positions of IDBs are called *parameter positions.* To visually separate the parameter positions from the non-distinguished positions, we use "$|$" as a delimiter, writing e.g. $P(x_1, x_2 \,|\, y_1, y_1, y_2) \leftarrow Q(y_1 \,|\, y_1, y_1, y_2) \wedge R(x_1, y_1, y_2, x_2)$ where P, Q are IDB, R is EDB, and there are three parameter positions. Note that, by definition, all variable positions in goal atoms are parameter positions.

▶ **Example 16.** The following is an MDLog program with one parameter that returns all constants which are on an R-cycle, R a binary EDB relation:

$$P(y \,|\, x) \;\leftarrow\; R(x \,|\, y); \qquad P(z \,|\, x) \;\leftarrow\; P(y \,|\, x) \wedge R(y, z); \qquad \mathsf{goal}(x) \;\leftarrow\; P(x \,|\, x)$$

Parameters in Datalog programs play a similar role as parameters to least fixed-point operators in FO(LFP), see for example [5] and references therein. The program in Example 16 is not definable in MDLog without parameters, which shows that adding parameters increases expressive power. But although (ℓ, k)-DLog programs with n parameters are $(\ell + n, k + n)$-DLog programs, one should think of them as a mild generalization of (ℓ, k)-programs.

To lift decidability and complexity results from the Boolean case to the non-Boolean case, we show that rewritability of an n-ary MDDLog program into (ℓ, k)-DLog with n parameters can be Turing reduced to rewritability of Boolean MDDLog programs into (ℓ, k)-DLog (without parameters). Note that the case $\ell = 0$ is about UCQ-rewritability (and thus about FO-rewritability) since 0-DLog programs (with and without parameters) are an alternative presentation of UCQs.

The reduction proceeds in two steps: first, rewritability of an n-ary MDDLog program into (ℓ, k)-DLog with n parameters is Turing reduced to rewritability of Boolean MDDLog programs with constants into (ℓ, k)-DLog with constants; then (ℓ, k)-DLog rewritability of a Boolean MDDLog program with constants is reduced to (ℓ, r)-DLog rewritability of a Boolean MDDLog program with constants, where r is the maximum number of occurrences of variables in a rule body of the program with constants. We can now lift the complexity results from Theorems 8 and 14 to the non-Boolean case.

▶ **Theorem 17.** *In MDDLog,*

1. *FO-rewritability (equivalently: UCQ-rewritability) is 2*NExpTime*-complete;*
2. *rewritability into MDLog with parameters is in 3*ExpTime *(and 2*NExpTime*-hard);*
3. *DLog-rewritability is 2*NExpTime*-complete for programs that have equality.*

In view of Point 2, we remark that for non-Boolean MDDLog programs Π, MDLog with parameters is in a sense a more natural target for rewriting than MDLog without parameters.

▶ **Example 18.** The following MDDLog program is rewritable into the MDLog program with parameters from Example 16, but not into an MDLog program without parameters:

$$P_0(x) \vee P_1(y) \leftarrow R(x, y); \quad \mathsf{goal}(x) \leftarrow P_0(x;) \quad P_1(y) \leftarrow P_1(x) \wedge R(x, y); \quad \mathsf{goal}(x) \leftarrow P_1(x)$$

MDLog with parameters also enjoys similarly nice properties as standard MDLog. For example, containment is decidable. This follows from [14, 39] where generalizations of MDLog with parameters are studied, the actual parameters being represented by constants. We also remark that Theorem 17 remains true even when we admit constants in MDDLog programs and that, as another consequence of our reductions, rewritability of MDDLog programs into DLog programs with parameters is decidable if and only if DLog-rewritability of Boolean MDDLog programs is decidable.

.

7.2 Canonical Datalog-Rewritings

We now turn our attention to canonical DLog-rewritings for non-Boolean MDDLog programs. For any n-ary MDDLog program Π, and $\ell < k$, we construct a *canonical* (ℓ, k)*-DLog program with* n *parameters*. The construction is a refinement of the one from the Boolean case.

We start with some preliminaries. An n*-marked instance* is an instance I endowed with n (not necessarily distinct) distinguished elements $\mathbf{c} = c_1, \dots, c_n$. An (ℓ, k)*-tree decomposition with* n *parameters* of an n-marked instance $(I, \mathbf{c})$ is an $(\ell + m, k + m)$-tree decomposition of I, m the number of distinct constants in $\mathbf{c}$, in which every bag B_v contains all constants from $\mathbf{c}$. An n-marked instance *has treewidth* (ℓ, k) *with* n *parameters* if it admits an (ℓ, k)-tree decomposition with n parameters.

We first convert Π into a DDLog program Π' that is equivalent to Π on instances of bounded treewidth. The construction is identical to the Boolean case (first variable identification, then rewriting) except that, in the rewriting step,

1. we use treewidth $(\ell + n, k + n)$ in place of treewidth (ℓ, k); consequently, the arity of the freshly introduced IDB relations may also be up to $\ell + n$;
2. for goal rules, all head variables occur in the root bag of the tree decomposition (they can then be treated in the same way as a Boolean goal rule despite the n-ary head relation).

It can be verified that Π' is sound for Π. It is complete for Π only on n-marked instances of treewidth (ℓ, k) with n parameters: for all such instances $(I, \mathbf{c})$, $I \models \Pi[\mathbf{c}]$ implies $I \models \Pi'[\mathbf{c}]$.

Let $\mathbf{S}'_I$ denote the new IDB relations in Π'. We now construct the canonical (ℓ, k)-DLog program with n parameters Γ^c. Fix constants $a_1, \dots, a_\ell$, $b_1, \dots, b_n$ and let $\mathfrak{I}_{\ell'+n}$ denote the set of all $\mathbf{S}_I \cup \mathbf{S}_{I'}$-instances with domain $\mathbf{a}_{\ell'+n} := a_1, \dots, a_{\ell'}$, $b_1, \dots, b_n$. The program uses $\ell' + n$-ary IDB relations P_M, for all $\ell' \leq \ell$ and all $M \subseteq \mathfrak{I}_{\ell'+n}$. It contains all rules $q(\mathbf{x}) \rightarrow P_M(\mathbf{y} \,|\, \mathbf{x}_p)$, $M \subseteq \mathfrak{I}_{\ell'+n}$, that satisfy the following conditions:

1. $q(\mathbf{x})$ contains at most $k + n$ variables;
2. in every extension J of the $\mathbf{S}_E$-instance $I_q|_{\mathbf{S}_E}$ with $\mathbf{S}_I \cup \mathbf{S}'_I$-facts such that
   - **a.** J satisfies all rules of Π' and does not contain $\mathsf{goal}(\mathbf{x}_p)$ and
   - **b.** for each $P_N(\mathbf{z} \,|\, \mathbf{x}_p) \in q$, $N \subseteq \mathfrak{I}_{\ell''+n}$, there is an $L \in N$ such that $L[\mathbf{z}\mathbf{x}_p/\mathbf{a}_{\ell''+n}] = J|_{\mathbf{S}_I \cup \mathbf{S}'_I, \mathbf{z}}$

   there is an $L \in M$ such that $L[\mathbf{y}\mathbf{x}_p/\mathbf{a}_{\ell'+n}] = J|_{\mathbf{S}_I \cup \mathbf{S}'_I, \mathbf{y}}$

We also include all rules of the form $P_\emptyset(\mathbf{y} \,|\, \mathbf{x}_p) \rightarrow \mathsf{goal}(\mathbf{x}_p)$. This finishes the construction. It is straightforward to verify that Γ^c is sound for Π and complete in the same sense as Π'. We obtain the following generalization of Theorem 15.

▶ **Theorem 19.** *Let* Π *be an* n*-ary MDDLog program,* $0 < \ell \leq k$*, and* Γ^c *the canonical* (ℓ, k)*-DLog program with* n *parameters associated with* Π*. Then*

1. $\Gamma \subseteq \Gamma^c$ *for every* (ℓ, k)*-DLog program* Γ *that is sound for* Π*;*
2. Π *is rewritable into* (ℓ, k)*-DLog with* n *parameters iff* Γ^c *is a rewriting of* Π*.*

7.3 Shape of Rewritings and Obstructions

We now analyze the shape of rewritings of non-Boolean MDDLog programs. An (ℓ, k)-*tree decomposition with* n *parameters* of an n-ary CQ q is an $(\ell + n, k + n)$-tree decomposition of q in which every bag B_v contains all answer variables of q. The treewidth with parameters of an n-ary CQ is now defined in the expected way.

▶ **Theorem 20.** *Let* Π *be an* n*-ary MDDLog program of diameter* k*. Then*

1. *if* Π *is FO-rewritable, then it has a UCQ-rewriting in which each CQ has treewidth* $(1, k)$ *with* n *parameters;*
2. *if* Π *is rewritable into MDLog with* n *parameters, then it has an MDLog-rewriting with* n *parameters of diameter* k*.*

As in the Boolean case, rewritings are closely related to obstructions. We define obstruction sets for MMSNP formulas with free variables and summarize the results that we obtain for them. A *set of marked obstructions* $\mathcal{O}$ for an MMSNP formula θ with n free variables over schema $\mathbf{S}_E$ is a set of n-marked instances over the same schema such that for any $\mathbf{S}_E$-instance I, we have $I \not\models \theta[\mathbf{a}]$ iff for some $(O, \mathbf{c}) \in \mathcal{O}$, there is a homomorphism h from O to I with $h(\mathbf{c}) = \mathbf{a}$. We obtain the following corollary from Point 1 of Theorem 20 in exactly the same way in which Corollary 10 is obtained from Point 1 of Theorem 9.

▶ **Corollary 21.** *For* θ *an MMSNP formula with* n *free variables, the following are equivalent:*

1. θ *is FO-rewritable;*
2. θ *has a finite marked obstruction set;*
3. θ *has a finite set of finite marked obstructions of treewidth* $(1, k)$ *with* n *parameters.*

It is interesting to note that this result can be viewed as a generalization of the characterization of obstruction sets for CSP templates with constants in terms of 'c-acyclicity' in [1]; our parameters correspond to constants in that paper. We now turn to MDLog-rewritability.

▶ **Proposition 22.** *Let* θ *be an MMSNP formula of diameter* k *with* n *free variables. Then* $\neg\theta$ *is rewritable into an MDLog program with* n *parameters iff* θ *has a set of marked obstructions (equivalently: finite marked obstructions) that are of treewidth* $(1, k)$ *with* n *parameters.*

8 Ontology-Mediated Queries

We study rewritability of ontology-mediated queries, covering several standard description logics as the ontology language. We start with introducing the relevant classes of queries.

An *ontology-mediated query (OMQ)* over a schema $\mathbf{S}_E$ is a triple $(\mathcal{T}, \mathbf{S}_E, q)$ where $\mathcal{T}$ is a TBox formulated in a description logic and q is a query over the schema $\mathbf{S}_E \cup \mathsf{sig}(\mathcal{T})$, $\mathsf{sig}(\mathcal{T})$ the set of relation symbols used in $\mathcal{T}$. The TBox can introduce symbols that are not in $\mathbf{S}_E$, which allows it to enrich the schema of the query q. As the TBox language, we use the description logic $\mathcal{ALC}$, its extension $\mathcal{ALCI}$ with inverse roles, and the further extension $\mathcal{SHI}$ of $\mathcal{ALCI}$ with transitive roles and role hierarchies. Since all these logics admit only unary and binary relations, we assume that these are the only allowed arities in schemas throughout the section. As the actual query language, we use UCQs and CQs. The OMQ languages that these choices give rise to are denoted with $(\mathcal{ALC}, \text{CQ})$, $(\mathcal{SHI}, \text{UCQ})$, and so on. In OMQs $(\mathcal{T}, \mathbf{S}_E, q)$ from $(\mathcal{SHI}, \text{UCQ})$, we disallow superroles of transitive roles in q; it is known that allowing such roles in the query poses serious additional complications, which are outside the scope of this paper, see e.g. [7, 26]. The semantics of an OMQ is given in terms of *certain answers*. More details are provided in the long version of the paper. An OMQ $Q = (\mathcal{T}, \mathbf{S}_E, q)$ is *FO-rewritable* if there is an FO query $\varphi(x)$ over schema $\mathbf{S}_E$ (and possibly involving equality), called an *FO-rewriting* of Q, such that for all $\mathbf{S}_E$-instances I and $\mathbf{a} \subseteq \mathsf{dom}(I)$, we have $I \models Q(\mathbf{a})$ iff $I \models \varphi(\mathbf{a})$. Other notions of rewritability such as UCQ-rewritability are defined accordingly. Note that the TBox $\mathcal{T}$ can be inconsistent with the input instance I, that is, there could be no model of $\mathcal{T}$ and I. It can thus be a sensible alternative to work with *consistent FO-rewritability*, considering only $\mathbf{S}_E$-instances I that are consistent w.r.t. $\mathcal{T}$. As argued in the long version, our results apply to consistent FO-rewritability, too.

▶ **Theorem 23.** *In all OMQ languages between* $(\mathcal{ALC}, UCQ)$ *and* $(\mathcal{SHI}, UCQ)$*, as well as between* $(\mathcal{ALCI}, CQ)$ *and* $(\mathcal{SHI}, UCQ)$*,*

1. *FO-rewritability (equivalently: UCQ-rewritability) is 2*NExpTime*-complete; in fact, there is an algorithm which, given an OMQ* $Q = (\mathcal{T}, \mathbf{S}_E, q)$*, decides in time* $2^{2^{p(n_q \cdot \log n_\mathcal{T})}}$ *whether* Q *is FO-rewritable;*
2. *MDLog-rewritability is in 3*ExpTime *(and 2*NExpTime*-hard); in fact, there is an algorithm which, given an OMQ* $Q = (\mathcal{T}, \mathbf{S}_E, q)$*, decides in time* $2^{2^{2^{p(n_q \cdot \log n_\mathcal{T})}}}$ *whether* Q *is MDLog-rewritable*

where n_q *and* $n_\mathcal{T}$ *are the size of* q *and* $\mathcal{T}$ *and* p *is a polynomial.*

Note that the runtime for deciding FO-rewritability stated in in Theorem 23 is double exponential only in the size of the actual query q (which tends to be very small) while it is only single exponential in the size of the TBox (which can become large) and similarly for MDLog-rewritability, only one exponential higher.

The lower bounds in Theorem 23 are from [15]. We obtain the upper bounds by translating the OMQ into an equivalent MDDLog program and then applying the constructions that we have already established. As shown in [11] and refined in [15], every OMQ from the languages mentioned in Theorem 23 can be converted into an equivalent MDDLog program at the expense of a single or even double exponential blowup, depending on the OMQ language. Thus, we can decide FO- or MDLog-rewritability of an OMQ Q from $(\mathcal{SHI}, \text{UCQ})$ by translating Q into an MDDLog program Π and deciding the same problem for Π. A detailed analysis of all the relevant blowups involved in the composed reductions reveals that, when implemented with sufficent care, we actually obtain a 2NExpTime upper bound.

9 Discussion

We have clarified the decidability status and computational complexity of FO- and MDLog-rewritability in MMSNP, MDDLog, and various OMQ languages based on expressive description logics and conjunctive queries. For Datalog-rewritability, we were only able to obtain partial results, namely a sound algorithm that is complete only on a certain class of inputs and potentially incomplete in general. This raises several natural questions: is our algorithm actually complete in general? Does an analogue of Lemma 6 (that is, rewritability on high girth implies rewritability) hold for Datalog as a target language? What is the complexity of deciding Datalog-rewritability in the afore-mentioned languages? From an OMQ perspective, it would also be important to work towards more practical approaches for computing (FO-, MDLog-, and DLog-) rewritings. Given the high computational complexities involved, such approaches might have to be incomplete to be practically feasible. However, the degree/nature of incompleteness should then be characterized, and we expect the results in this paper to be helpful in such an endeavour.

Acknowledgement. We thank Libor Barto, Manuel Bodirsky, and Florent Madeleine for helpful discussions.

References

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

2 Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *JAIR*, 36:1–69, 2009.

3 Albert Atserias. On digraph coloring problems and treewidth duality. *Eur. J. Comb.*, 29(4):796–820, 2008.

4 Libor Barto. The collapse of the bounded width hierarchy. *J. Log. Comput.*, 26(3):923–943, 2016. URL: `http://dx.doi.org/10.1093/logcom/exu070`, `doi:10.1093/logcom/exu070`.

5 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. A step up in expressiveness of decidable fixpoint logics. In *Proc. of LICS*. IEEE Computer Society, 2016.

6 Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *Proc. of LICS*, pages 293–304. IEEE Computer Society, 2015.

7 Meghyn Bienvenu, Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Simkus. Query answering in the description logic $\mathcal{S}$. In *Proc. of DL2010*, volume 573 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010. URL: `http://ceur-ws.org/Vol-573/paper_20.pdf`.

8 Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First order-rewritability and containment of conjunctive queries in horn description logics. In *Proc. of IJCAI*, 2016.

9 Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. First-order rewritability of atomic queries in horn description logics. In *Proc. of IJCAI*, 2013.

10 Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Proc. of Reasoning Web*, volume 9203 of *LNCS*, pages 218–307. Springer, 2015.

11 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014. URL: `http://doi.acm.org/10.1145/2661643`, `doi:10.1145/2661643`.

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

13 Manuel Bodirsky and Víctor Dalmau. Datalog and constraint satisfaction with infinite templates. *J. Comput. Syst. Sci.*, 79(1):79–100, 2013. URL: `http://dx.doi.org/10.1016/j.jcss.2012.05.012`, `doi:10.1016/j.jcss.2012.05.012`.

14 Pierre Bourhis, Markus Krötzsch, and Sebastian Rudolph. Reasonable highly expressive query languages. In *Proc. of IJCAI2015*, pages 2826–2832. AAAI Press, 2015. URL: `http://ijcai.org/papers15/Abstracts/IJCAI15-400.html`.

15 Pierre Bourhis and Carsten Lutz. Containment in monadic disjunctive Datalog, MMSNP, and expressive description logics. In *Proc. of KR*, 2016.

16 Andrei A. Bulatov, Andrei A. Krokhin, and Benoit Larose. Dualities for constraint satisfaction problems. In *Complexity of Constraints - An Overview of Current Research Themes*, volume 5250 of *LNCS*, pages 93–124. Springer, 2008.

17 Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The DL-Lite approach. In *Proc. of Reasoning Web 2009*, volume 5689 of *LNCS*, pages 255–356. Springer, 2009. URL: `http://dx.doi.org/10.1007/978-3-642-03754-2_7`, `doi:10.1007/978-3-642-03754-2_7`.

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

19 Hubie Chen and Benoit Larose. Asking the metaquestions in constraint tractability. *CoRR*, abs/1604.00932, 2016.

20 Gregory L. Cherlin, Saharon Shelah, and Niandong Shi. Universal graphs with forbidden subgraphs and algebraic closure. *Advances in Applied Mathematics*, 22:454–491, 1999.

21 Víctor Dalmau and Benoit Larose. Maltsev + datalog –> symmetric datalog. In *Proc. of LICS*, pages 297–306. IEEE Computer Society, 2008.

22 László Egri, Benoit Larose, and Pascal Tesson. Symmetric datalog and constraint satisfaction problems in logspace. In *Proc. of LICS*, pages 193–202. IEEE Computer Society, 2007.

23 Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI*. AAAI Press, 2012.

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

25 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

26 Georg Gottlob, Andreas Pieris, and Lidia Tendera. Querying the guarded fragment with transitivity. In *Proc. of ICALP2013*, volume 7966 of *LNCS*, pages 287–298. Springer, 2013. URL: `http://dx.doi.org/10.1007/978-3-642-39212-2_27`, `doi:10.1007/978-3-642-39212-2_27`.

27 Peter Hansen, Carsten Lutz, İnanç Seylan, and Frank Wolter. Efficient query rewriting in the description logic el and beyond. In *Proc. of IJCAI*, 2015.

28 Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. Computing datalog rewritings for disjunctive datalog programs and description logic ontologies. In *Proc. of RR*, pages 76–91, 2014.

29 Benoit Larose, Cynthia Loten, and Claude Tardif. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Computer Science*, 3(4), 2007. URL: `http://dx.doi.org/10.2168/LMCS-3(4:6)2007`, `doi:10.2168/LMCS-3(4:6)2007`.

30 Benoit Larose and László Zádori. Bounded width problems and algebras. *Algebra universalis*, 56(3):439–466, 2007.

31 Florent R. Madelaine. Universal structures and the logic of forbidden patterns. *Logical Methods in Computer Science*, 5(2), 2009. URL: `http://arxiv.org/abs/0904.2521`.

32 Florent R. Madelaine. On the containment of forbidden patterns problems. In *Proc. of CP2010*, volume 6308 of *LNCS*, pages 345–359. Springer, 2010. URL: `http://dx.doi.org/10.1007/978-3-642-15396-9_29`, `doi:10.1007/978-3-642-15396-9_29`.

33 Florent R. Madelaine and Iain A. Stewart. Constraint satisfaction, logic and forbidden patterns. *SIAM J. Comput.*, 37(1):132–163, 2007. URL: `http://dx.doi.org/10.1137/050634840`, `doi:10.1137/050634840`.

34 Jaroslav Nesetril. Many facets of dualities. In *Proc. of Workshop on Combinatorial Optimization*, pages 285–302. Springer, 2008.

35 Jaroslav Nešetřil and Claude Tardif. Duality theorems for finite structures (characterising gaps and good characterisations). *J. Comb. Theory, Ser. B*, 80(1):80–97, 2000.

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

37 Riccardo Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proc. of DL*, pages 451–458, 2007.

38 Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3):15:1–15:53, 2008.

39 Sebastian Rudolph and Markus Krötzsch. Flag & check: data access with monadically defined queries. In *Proc. of PODS*, pages 151–162. ACM, 2013.

40 Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos B. Stamou. Optimising resolution-based rewriting algorithms for OWL ontologies. *J. Web Sem.*, 33:30–49, 2015.