Flag & Check: Data-Tractable Expressive Queries for Intelligent Databases (Extended Technical Report)

Technical Report 3030, Institute AIFB, Karlsruhe Institute of Technology

Sebastian Rudolph
Institute AIFB
Karlsruhe Institute of Technology, DE
sebastian.rudolph@kit.edu

Markus Krötzsch
Department of Computer Science
University of Oxford, UK
markus.kroetzsch@cs.ox.ac.uk

ABSTRACT

We propose two novel querying formalisms: *monadically defined queries* (MODEQs) and the more expressive *nested monadically defined queries* (NEMODEQs). Both subsume and go beyond conjunctive queries, conjunctive two-way regular path queries, and monadic Datalog queries. Moreover, MODEQs and NEMODEQs can be expressed as Datalog queries but also in monadic second-order logic, but, unlike those formalisms, they have a decidable query subsumption problem and favorable query answering complexities: they both exhibit P data complexity, whereas the combined complexity is NP for MODEQs and PSpace for NEMODEQs.

We show that MODEQ answering remains decidable in the presence of tuple-generating dependencies (TGDs) of a certain type known as *bounded-treewidth sets*.

We then investigate the topic of query rewriting under dependencies. To this end, we extend the notion of first-order rewritability to (NE)MODEQ rewritability and show that this extended notion can cope with a large variety of TGDs. We devise methods for rewriting rule sets to queries in this new formalism. Finally, we show that rewriting techniques can also be applied partially, and NEMODEQ answering is still decidable, if the non-rewritable part of the TGDs allows for decidable NEMODEQ answering on other grounds.

1. INTRODUCTION

Query languages are fundamental to the design of database systems. A good query language should be able to express a wide range of common information needs, and allow queries to be answered efficiently with limited computational resources. Moreover, databases are often considered in combination with dependencies, e.g., in the form of *tuple-generating dependencies* (TGDs), which are also playing an important role in data exchange, information integration, and database integrity checking [1]. While query answering under dependencies is undecidable in general, there are many decidable cases, and a query language should be robustly applicable to such extensions [41]. Another important task in database management and optimization is to check *query subsumption*, that is, to determine whether the answers of one query are subsumed by the answers of another query over arbitrary databases, possibly under the additional assumption that certain constraints are satisfied. A query language should therefore allow for such checks.

Unfortunately, these basic requirements are in conflict. Very simple query languages like *conjunctive queries* (CQs, [19]) allow for efficient query answering (NP combined/AC_0 data[1]) and subsumption checking, but have very limited expressivity. First-order logic (FOL) queries extend expressivity, but are still restricted to "local" queries, excluding, e.g., the transitive closure of a relation. Query subsumption is undecidable for FOL, and query answering becomes PSpace-complete for combined complexity [45], but remains AC_0 for data [33]. Another extension of CQs is *Datalog*, which introduces rule-based recursion. The price are higher complexities (ExpTime combined/P data), and undecidability of query subsumption [24, 44]. FOL and Datalog are incomparable; both are subsumed by second-order logic (SO), which is more expressive but also more complex (ExpSpace combined/PH data) [33, 46].

To find more tractable query languages, various smaller fragments of Datalog have been considered. *Linear Datalog* allows only one inferred predicate per rule body, which significantly reduces query complexity (PSpace combined/NLogSpace data) [30]. However, query subsumption remains undecidable. Two query languages for which subsumption is decidable are *monadic Datalog* and *conjunctive 2-way regular path queries* (C2RPQs) [28, 17]. The query complexity of C2RPQs (NP combined, NLogSpace data) is slightly lower than that of monadic Datalog (NP combined, P data), but the expressivity of the languages is incomparable. In particular, monadic Datalog cannot express transitive closure. Two well-known query languages that subsume monadic Datalog and C2RPQs are Datalog and *monadic second-order logic* (MSO) [38]. Query subsumption is decidable for neither of these. Moreover, both languages are incomparable, even regarding query complexities (MSO has PSpace combined/PH data [45, 47, 38]), their common upper bound being SO.

This reveals a glaring gap in the landscape of known query languages: no formalism that captures monadic Datalog and C2RPQs ensures tractable data complexity and decidable query subsumption. To address this, we propose *monadically defined queries* (MODEQs) as a novel query formalism that combines these desirable properties. Its relationship to the aforementioned languages is illustrated in Fig. 1.

The contribution of this paper can be split in two parts. In the first part, we introduce the new querying formalism and clarify its

[1]As usual, query complexities refer to the problem of deciding whether a query has a particular certain answer. *Combined complexity* is the complexity for arbitrary queries and databases; *data complexity* is the complexity if the query is fixed or bounded.

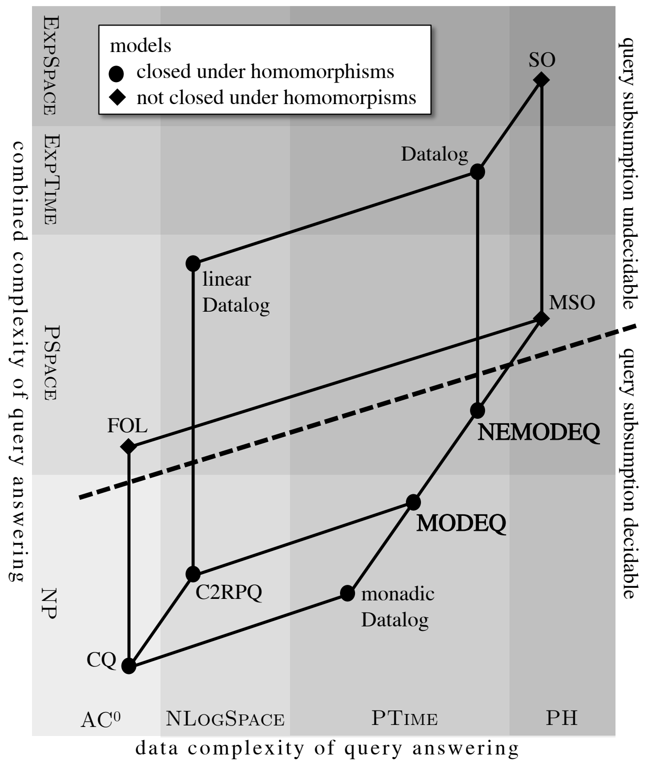
models
● closed under homomorphisms
◆ not closed under homomorpisms
ExpSpace
ExpTime
PSpace
NP
AC0
NLogSpace
PTime
PH
combined complexity of query answering
data complexity of query answering
query subsumption undecidable
query subsumption decidable
SO
Datalog
linear Datalog
MSO
FOL
NEMODEQ
MODEQ
C2RPQ
monadic Datalog
CQ

Figure 1: Overview of complexities and relations of monadically defined queries to other query formalisms as established in this paper; information indicated for conjunctive queries (CQ) and conjunctive 2-way regular path queries (C2RPQ) also hold when allowing unions (UCQ and UC2RPQ); all other formalisms are closed under unions

relations to established query notions and the complexity for query answering. More precisely:

- We define MODEQs, discuss the underlying intuition, and provide examples to demonstrate their expressivity.
- We show that MODEQs subsume (unions of) conjunctive queries, conjunctive 2-way regular path queries, and monadic Datalog queries.
- We show that MODEQ answering is NP-complete for combined complexity (i.e., on a level with conjunctive queries) and P-complete for data complexity, ensuring data-tractability as one of the central desiderata for querying large data sets.

We then extend MODEQs with nested subqueries, leading to a broader class of NEMODEQs that generalize MODEQs.

- We show that NEMODEQs (and thus MODEQs) can be expressed both by Datalog queries and by formulae in MSO.
- We show that NEMODEQ answering is PSpace-complete for combined complexity (i.e., on a level with FOL-based query languages like SQL) and P-complete for data complexity.
- We show that the query subsumption problem for (NE)MODEQs is decidable, in contrast to Datalog and MSO queries.

In the second part of the paper, we study (NE)MODEQs in the context of dependencies and ontology-based data access. An important tool in this context are (finite or infinite) *universal models*, which represent solutions to data exchange and constraint repair problems in the presence of TGDs [25]. Since models of (NE)MODEQ are closed under homomorphisms, we find that universal models can be used to answer such queries, which makes them very robust to a broad class of TGDs.

- We immediately obtain decidability of query answering under all TGDs that admit a finite universal model, which is undecidable, but can be approximated by various notions of *acyclicity* [26, 27, 25, 40, 31, 36].
- More generally, we show that MODEQ answering is decidable in the presence of rules that give rise to (possibly infinite) universal models of *bounded tree-width*. This is the case for many lightweight ontology languages as well as guarded TGDs and generalizations thereof [11, 6, 36, 7].
- In analogy to the practically relevant notion of first-order rewritability, we introduce (NE)MODEQ rewritability, and we identify basic criteria for rewriting Datalog rules.
- Finally, we show that query answering is decidable under any set of TGDs that can be decomposed into one that is (NE)MODEQ-rewritable and one with the bounded-treewidth-model property.

Proofs are omitted from the main paper and given in the Appendix.

2. PRELIMINARIES

We consider a standard language of first-order predicate logic, based on a finite set of *constant symbols* **C**, a finite set of *predicate symbols* **P**, and an infinite set of first-order *variables* **V**. Each predicate $p \in \mathbf{P}$ is associated with a natural number $\mathsf{ar}(p)$ called the *arity* of p. The list of predicates and constants forms the language's *signature* (or *schema*) $\mathscr{S} = \langle \mathbf{P}, \mathbf{C} \rangle$. We generally assume $\mathscr{S} = \langle \mathbf{P}, \mathbf{C} \rangle$ to be fixed, and only refer to it explicitly if needed.

Databases, Rules, and Queries. A *term* is a variable $x \in \mathbf{V}$ or a constant $c \in \mathbf{C}$. We use symbols s, t to denote terms, x, y, z, v, w to denote variables, a, b, c to denote constants. Expressions like $\mathbf{t}$, $\mathbf{x}$, $\mathbf{c}$ denote finite lists of such entities. We use the standard predicate logic definitions of *atom* and *formula*, using symbols φ, ψ for the latter. A formula is *ground* if it contains no variables. A *database*, usually denoted by D, is a finite set of ground atoms. We write $\varphi[\mathbf{x}]$ to emphasize that a formula φ has free variables $\mathbf{x}$; we write $\varphi[\mathbf{c}/\mathbf{x}]$ for the formula obtained from φ by replacing each variable in $\mathbf{x}$ by the respective constant in $\mathbf{c}$ (both lists must have the same length). A formula without free variables is called a *sentence*.

A *conjunctive query* (CQ) is a formula $Q[\mathbf{x}] = \exists \mathbf{y}.\psi[\mathbf{x}, \mathbf{y}]$ where $\psi[\mathbf{x}, \mathbf{y}]$ is a conjunction of atoms. A *tuple-generating dependency* (TGD) is a formula of the form $\forall \mathbf{x}, \mathbf{y}.\varphi[\mathbf{x}, \mathbf{y}] \to \exists \mathbf{z}.\psi[\mathbf{x}, \mathbf{z}]$ where φ and ψ are conjunctions of atoms, called the *body* and *head* of the TGD, respectively. TGDs never have free variables, so we usually omit the universal quantifier when writing them. We use the symbol Σ, possibly with subscripts, to denote sets of TGDs. A *Datalog rule* is a TGD without existentially quantified variables; sets of Datalog rules will be denoted by symbols $\mathbb{P}, \mathbb{R}, \mathbb{S}$.

We use the standard semantics of first-order logic (FOL). An *interpretation* $\mathcal{I}$ consists of a (possibly infinite) set $\Delta^{\mathcal{I}}$ called *domain* and a function $\cdot^{\mathcal{I}}$ that maps constants c to domain elements $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ and predicate symbols p to relations $p^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^{\mathsf{ar}(p)}$, thereby $p^{\mathcal{I}}$ is called the *extension* of p. A *variable assignment* for $\mathcal{I}$ is a function $\mathcal{Z} : \mathbf{V} \to \Delta^{\mathcal{I}}$. Conditions for $\mathcal{I}$ and $\mathcal{Z}$ to satisfy a FOL formula φ (i.e., to be a *model* of φ, written $\mathcal{I}, \mathcal{Z} \models \varphi$) are defined as usual. If φ is a sentence, then $\mathcal{Z}$ is irrelevant for satisfaction and can be omitted. An *answer* to a CQ $Q[\mathbf{x}]$ over a database D and set Σ of TGDs is a list of constants $\mathbf{c}$ for which $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$.

Given an interpretation $\mathcal{I}$ and a formula $\varphi[\mathbf{x}]$ with free variables $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$, the *extension* of $\varphi[\mathbf{x}]$ is the subset of $(\Delta^{\mathcal{I}})^m$ containing all those tuples $\langle \delta_1, \ldots, \delta_m \rangle$ for which $\mathcal{I}, \{x_i \mapsto \delta_i \mid 1 \leq i \leq m\} \models \varphi[\mathbf{x}]$. Two formulae $\varphi[\mathbf{x}]$ and $\psi[\mathbf{x}]$ with the same free variables $\mathbf{x}$ are called *equivalent* if for every $\mathcal{I}$ their extensions coincide (which is the case iff $\forall \mathbf{x}.(\varphi[\mathbf{x}] \leftrightarrow \psi[\mathbf{x}])$ is a tautology).

Homomorphisms and Universal Models. Given interpretations $\mathcal{I}$, $\mathcal{J}$, a *homomorphism* π from $\mathcal{I}$ to $\mathcal{J}$ is a function $\pi : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ such that: (i) for all constants c, we have $\pi(c^{\mathcal{I}}) = c^{\mathcal{J}}$, and (ii) for all predicate symbols p and list of domain elements $\boldsymbol{\delta}$, we have $\boldsymbol{\delta} \in p^{\mathcal{I}}$ implies $\pi(\boldsymbol{\delta}) \in p^{\mathcal{J}}$.

Finding query answers is facilitated in practice since one may focus on universal models. A *universal model* of a set of sentences Ψ is an interpretation $\mathcal{I}$ such that (i) $\mathcal{I} \models \Psi$, and (ii) for every interpretation $\mathcal{J}$ with $\mathcal{J} \models \Psi$, there is a homomorphism from $\mathcal{I}$ to $\mathcal{J}$. For TGDs (and for plain databases), there is always a universal model if there is any model at all. It can be defined by a (possibly infinite) construction process called the *chase* [25]. In particular, we let $\mathcal{I}(D \cup \Sigma)$ denote the universal model, for which every homomorphism in any other model of $D \cup \Sigma$ is injective. $\mathcal{I}(D \cup \Sigma)$ always exists for satisfiable $D \cup \Sigma$, and it is unique up to isomorphism.

For a wide class of query languages, entailment of query answers can be reduced to model checking in the universal model:

Fact 1 (Entailment via model checking). *If $Q[\mathbf{x}]$ is a query for which the models of $\exists \mathbf{x}.Q[\mathbf{x}]$ are closed under homomorphism, then, for every database database D and set Σ of TGDs, $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ if and only if either $D \cup \Sigma$ is inconsistent or $\mathcal{I}(D \cup \Sigma) \models Q[\mathbf{c}/\mathbf{x}]$.*

This applies to CQs, but also to all other query languages studied in this paper. The special case $\Sigma = \emptyset$ shows that one can equivalently use models $I(D)$ instead of sets of facts D to represent databases. Our perspective is more natural when using TGDs.

3. MONADICALLY DEFINED QUERIES

We now introduce *monadically defined queries (MODEQs)* as a new query formalism, and state complexity results on query answering in this language. To deepen our understanding for the expressivity of MODEQs, we show that they strictly generalizes the well-known query formalisms of *conjunctive 2-way regular path queries* (Section 3.1) and *monadic Datalog queries* (Section 3.2).

The heart of our query formalism is a mechanism for defining new predicates based on existing ones. This mechanism – which we refer to as *"flag & check"* – defines new predicates by giving a procedure for testing if a particular tuple $\boldsymbol{\delta} = \langle \delta_1, \ldots, \delta_m \rangle \in (\Delta^{\mathcal{I}})^m$ is in the predicate or not. For this purpose, the candidate tuple is first "flagged" by associating each δ_i with a new, auxiliary constant name λ_i that represents this element. The "check" is performed by running a Datalog program with this fixed interpretation of the constants λ_i. The check succeeds if a special fact `hit` is derived.

Example 1. *To illustrate this idea, we consider a typical transitive closure query. Suppose that the binary predicate certifiedBy represents the direct certification of one entity by another, e.g., in a security application. We are interested in certification chains, which could be expressed using Datalog as follows:*

$$\mathit{certifiedBy}(x, y) \to \mathit{certChain}(x, y) \qquad (1)$$

$$\mathit{certChain}(x, y) \land \mathit{certifiedBy}(y, z) \to \mathit{certChain}(x, z) \qquad (2)$$

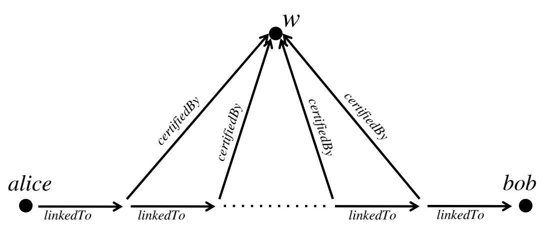

Figure 2: Structure recognized by MODEQ $\mathfrak{Q}_2$ in Example 2

Corresponding Datalog rules for "flag & check" are as follows:

$$\mathit{certifiedBy}(\lambda_1, y) \to \mathtt{U}_1(y) \qquad (3)$$

$$\mathtt{U}_1(y) \land \mathit{certifiedBy}(y, z) \to \mathtt{U}_1(z) \qquad (4)$$

$$\mathtt{U}_1(\lambda_2) \to \mathtt{hit} \qquad (5)$$

One can now define certChain to contain all pairs $\langle \delta_1, \delta_2 \rangle$ for which this Datalog entails `hit` *when interpreting λ_1 as δ_1 and λ_2 as δ_2.*

As in Example 1, the Datalog rules that we consider for the checking phase only use `hit` or new, unary predicates $\mathtt{U}_i$ in rule heads. Such unary predicates can be imagined as "colors" that are assigned to elements of the domain, and the check thus is a deterministic, recursive procedure of coloring the domain, starting from the flagged candidate elements. This idea is defined formally as follows.

Definition 1 (MODEQ syntax and semantics). *Given a signature $\mathscr{S}$, a* monadically defined predicate (MODEP) *of arity m is based on a signature $\mathscr{S}'$ that extends $\mathscr{S}$ with m fresh constant symbols $\lambda_1, \ldots, \lambda_m$, a fresh nullary predicate* `hit`*, and $k \geq 0$ fresh unary predicates $\mathtt{U}_1, \ldots, \mathtt{U}_k$. A MODEP is a set $\mathbb{P}$ of Datalog rules over $\mathscr{S}'$ where only $\mathtt{U}_1, \ldots, \mathtt{U}_k$, and* `hit` *occur in rule heads.*

Consider an interpretation $\mathcal{I}$ over $\mathscr{S}$. The extension $\mathbb{P}^{\mathcal{I}}$ of $\mathbb{P}$ is the set of all tuples $\langle \delta_1, \ldots, \delta_m \rangle \in (\Delta^{\mathcal{I}})^m$ for which $\mathcal{I}' \models \mathbb{P}$ implies $\mathcal{I}' \models$ `hit`*, for all interpretations $\mathcal{I}'$ that extend $\mathcal{I}$ to the additional symbols in $\mathscr{S}'$ such that $\langle \lambda_1^{\mathcal{I}'}, \ldots, \lambda_m^{\mathcal{I}'} \rangle = \langle \delta_1, \ldots, \delta_m \rangle$.*

A monadically defined query (MODEQ) *is a conjunctive query that uses both normal predicates and monadically defined predicates in its atoms. The semantics of MODEQs is defined in the obvious way based on the semantics of MODEPs.*

Example 2. *The rules* (3)–(5) *above define a binary MODEP, which we denote $\mathbb{P}_1$. The query from Example 1 could thus be written as a MODEQ $\mathfrak{Q}_1[v, w] = \mathbb{P}_1(v, w)$. For another example, consider a routing problem where a message has to be securely passed through a network from Alice to Bob. Assume there are entities certifying the security of message handling in certain nodes. We are interested in the entities y are able to (directly) certify secure treatment of the message at all intermediate nodes on some path from Alice to Bob, as illustrated in Fig. 2. This can be expressed by the MODEQ $\mathfrak{Q}_2[w] = \mathbb{P}_2(w)$, where $\mathbb{P}_2$ consists of the following rules:*

$$\mathit{certifiedBy}(x, \lambda_1) \to \mathtt{U}_3(x) \qquad (6)$$

$$\mathit{linkedTo}(\mathit{alice}, x) \to \mathtt{U}_2(x) \qquad (7)$$

$$\mathtt{U}_2(x) \land \mathtt{U}_3(x) \land \mathit{linkedTo}(x, x') \to \mathtt{U}_2(x') \qquad (8)$$

$$\mathtt{U}_2(\mathit{bob}) \to \mathtt{hit} \qquad (9)$$

This new query notion is rather powerful. It is easy to see that it subsumes conjunctive queries (CQs) as well as unions of CQs. Indeed, given k CQs $\exists \mathbf{y_i}.Q_i[\mathbf{x}, \mathbf{y_i}]$ with $i \in \{1, \ldots, k\}$, their union is expressed as the MODEQ $\{Q_i[\boldsymbol{\lambda}, \mathbf{y_i}] \to \mathtt{hit} \mid 1 \leq i \leq k\}(\mathbf{x})$. Before showing that MODEQs also subsume more powerful query languages, we observe closure under homomorphism (see Fact 1) and state a basic complexity result.

Theorem 2. *For every MODEQ $\mathfrak{Q}[\mathbf{x}]$, the set of models of $\exists\mathbf{x}.\mathfrak{Q}[\mathbf{x}]$ is closed under homomorphism.*

Theorem 3 (MODEQ answering complexity). *Checking if* $\mathbf{c}$ *is an answer to a MODEQ $\mathfrak{Q}[\mathbf{x}]$ over a database D is* P*-complete in the size of D, and* NP*-complete in the size of D and $\mathfrak{Q}$.*

Hardness follows from the fact that MODEQs subsume monadic Datalog, shown in Section 3.2 below. P membership for data complexity is a consequence of the fact that Datalog subsumes MODEQs, demonstrated in Section 4.1. Membership in NP for combined complexity is established directly by showing that every query match is witnessed by a proof of polynomial size that can be guessed and verified in polynomial time.

3.1 MODEQs Capture Regular Path Queries

We now show that MODEQs subsume *conjunctive two-way regular path queries (C2RPQs)*, which generalize CQs by regular expressions over binary predicates [28, 17]. Variants of this type of queries are used, e.g., by the XPath query language for querying semi-structured XML data. Recent versions of the SPARQL 1.1 query language for RDF also support some of regular expressions that can be evaluated under a similar semantics.[2]

The definition of C2RPQs is similar to that of MODEQs, but with MODEPs replaced by another form of defined predicates, that are based on regular expressions over binary predicates and inverted binary predicates:

Definition 2 (C2RPQ syntax and semantics). *A* conjunctive two-way regular path predicate (C2RPP) *is a regular expression over the alphabet $\Gamma = \{p, p^- \mid \mathsf{ar}(p) = 2\}$ of normal and inverse binary predicate symbols. All C2RPPs are of arity 2. Consider an interpretation $\mathcal{I}$. For inverse predicates p^-, we define $(p^-)^{\mathcal{I}} := \{\langle\delta_2, \delta_1\rangle \mid \langle\delta_1, \delta_2\rangle \in p^{\mathcal{I}}\}$. For a C2RPP P, we set $\langle\delta, \delta'\rangle \in P^{\mathcal{I}}$ if there is a word $\gamma_1 \ldots \gamma_n$ matching the regular expression P, and a sequence $\delta_0 \ldots \delta_n$ of domain elements such that $\delta_0 = \delta$, $\delta_n = \delta'$, and $\langle\delta_i, \delta_{i+1}\rangle \in \gamma_i^{\mathcal{I}}$ for every $i \in \{0, \ldots, n-1\}$.*

A conjunctive two-way regular path query (C2RPQ) *is a conjunctive query that uses both normal predicates and C2RPPs in its atoms. The semantics of C2RPQs is defined in the obvious way based on the semantics of C2RPPs.*

Example 3. *The query from Example 1 can be expressed as a C2RPQ certifiedBy$^*(x, y)$. An example with inverse predicates is*

$$mountain(x) \wedge continent(y) \wedge (locatedIn|hasPart^-)^*(x, y). \quad (10)$$

Query answering for C2RPQs is NP-complete regarding the size of the database and query, which is the same as for CQs. In terms of data complexity, C2RPQs are NLogSpace-complete, and thus harder than CQs (AC$_0$). One can show hardness via graph reachability, and membership via a translation to linear Datalog [15].

[2]The original proposal for SPARQL 1.1 used a slightly different semantics that led to computational problems [4]; this was fixed before the finalization of SPARQL 1.1.

Definition 3 (C2RPQ to MODEQ translation). *Consider a C2RPP P and a finite automaton $\mathcal{A}_P = \langle\Gamma, S, I, F, T\rangle$ that recognizes P. The binary MODEP* $\mathsf{modep}(P)$ *to consist of the rules*

$$\begin{aligned}
&\to \mathtt{U}_s(\lambda_1) && \textit{for every initial state } s \in I,\\
\mathtt{U}_s(z) \wedge p(z, z') &\to \mathtt{U}_{s'}(z') && \textit{for every transition } \langle s, p, s'\rangle \in T,\\
\mathtt{U}_s(z) \wedge p(z', z) &\to \mathtt{U}_{s'}(z') && \textit{for every transition } \langle s, p^-, s'\rangle \in T,\\
\mathtt{U}_s(\lambda_2) &\to \mathtt{hit} && \textit{for every final state } s \in F.
\end{aligned}$$

Given a C2RPQ Q, a MODEQ $\mathsf{modeq}(Q)$ *is obtained by replacing every C2RPP P in Q by* $\mathsf{modep}(P)$.

The intuition behind the translation of C2RPQs to MODEQs is to find the possible bindings to x and y in $P(x, y)$ by simulating all possible runs of the automaton corresponding to a C2RPQ. Colors $\mathtt{U}_s$ are associated to states s of the automaton, so as to keep track which domain elements can be reached in which states when starting in an initial state at x. The success criterion is satisfied if y is colored by a final state. One can thus show the following result.

Theorem 4 (MODEQs capture C2RPQs). *For any C2RPQ Q, the MODEQ* $\mathsf{modeq}(Q)$ *can be constructed in linear time, and is equivalent to Q, i.e., $\forall\mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathsf{modeq}(Q)[\mathbf{x}]$ is a tautology. In particular, the answers for Q and* $\mathsf{modeq}(Q)$ *coincide.*

Example 4. *Let Q be the regular path query* (10). *The query* $\mathsf{modeq}(Q)$ *is $mountain(x) \wedge continent(y) \wedge \mathbb{P}(x, y)$ where $\mathbb{P}$ denotes* $\mathsf{modep}((locatedIn|hasPart^-)^*)$*, which consists of the rules:*

$$\to \mathtt{U}(\lambda_1) \quad (11)$$

$$\mathtt{U}(z) \wedge locatedIn(z, z') \to \mathtt{U}(z') \quad (12)$$

$$\mathtt{U}(z) \wedge hasPart(z', z) \to \mathtt{U}(z') \quad (13)$$

$$\mathtt{U}(\lambda_2) \to \mathtt{hit}. \quad (14)$$

Here we only need one "color" $\mathtt{U}$ *that is propagated over locatedIn and inversely over hasPart. The pairs in $\mathbb{P}$ are those for which this process, started at the first element, will eventually color the second argument.*

However, the expressivity of MODEQs goes well beyond that of C2RPQs, even when considering only binary predicates. This follows from the easy observation that for every C2RPQ Q there is an integer n, such that whenever Q matches into a graph G, it also matches into a graph G' where all vertices have degree $\leq n$ and from which there is a homomorphism into G. It is easy to see that the MODEQ $\mathfrak{Q}_2$ from Example 2 does not have this property.

3.2 MODEQs Capture Monadic Datalog

Monadic Datalog queries are another type of query language that enjoys favorable computational properties. They are used, e.g., for information extraction from the Web [29]. We now show that MODEQs can express monadic Datalog queries, which is another way to see that they are strictly more general than C2RPQs.

Definition 4 (Monadic Datalog query). *Given a signature $\mathscr{S}$, a* Datalog query *is based on a signature $\mathscr{S}'$ that extends $\mathscr{S}$ with additional predicates, called* intentional database (IDB) predicates. *A Datalog query is a pair $\langle\mathtt{goal}, \mathbb{S}\rangle$, where* $\mathtt{goal}$ *is an IDB predicate, and $\mathbb{S}$ is a set of Datalog rules over $\mathscr{S}'$ where only IDB predicates occur in rule heads and* $\mathtt{goal}$ *does not occur in rule bodies. A* monadic Datalog query *is a query where all IDB predicates other than* $\mathtt{goal}$ *have arity* 1.

Given an interpretation $\mathcal{I}$, the extension *of $\langle\mathtt{goal}, \mathbb{S}\rangle$ is the set of tuples $\boldsymbol{\delta}$ over $\Delta^{\mathcal{I}}$ for which every extension $\mathcal{I}'$ of $\mathcal{I}$ to $\mathscr{S}'$ which satisfies $\mathbb{S}$ must also satisfy $\boldsymbol{\delta} \in \mathtt{goal}^{\mathcal{I}'}$.*

Allowing `goal` in monadic Datalog queries to have arbitrary arity is essentially equivalent to considering a conjunctive query that uses both normal predicates and monadic IDB predicates. This would be closer to our earlier definitions, but less concise to define. In any case, the additional expressivity of non-unary `goal` predicates does not affect the complexity of the formalism.

Definition 5 (Monadic Datalog to MODEQ translation). *Given a monadic Datalog query $Q = \langle \mathtt{goal}, \mathbb{S} \rangle$, we let* $\mathsf{modeq}(Q)$ *denote the MODEQ* $\mathbb{P}(\mathbf{x})$ *where* $\mathbb{P}$ *is obtained from* $\mathbb{S}$ *by*

- *replacing each rule* $\varphi[\mathbf{x}, \mathbf{y}] \to \mathtt{goal}(\mathbf{x})$ *by* $\varphi[\lambda, \mathbf{y}] \to \mathtt{hit}$,
- *replacing each IDB predicate, uniformly and injectively, by a predicate* $\mathtt{U}_j$.

Theorem 5 (MODEQs capture monadic Datalog). *For every monadic Datalog query Q, the MODEQ* $\mathsf{modeq}(Q)$ *can be constructed in linear time, and is equivalent to Q, i.e.,* $\forall \mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathsf{modeq}(Q)[\mathbf{x}]$ *is a tautology. In particular, the answers for Q and* $\mathsf{modeq}(Q)$ *coincide.*

From this established correspondence follows that the lower complexity bounds of monadic Datalog carry over and MODEQ answering on databases must thus be P-hard for data complexity and NP-hard for combined complexity [29], showing one direction of Theorem 3. MODEQs are strictly more expressive than monadic Datalog queries, shown by the fact that even a simple connectedness query like the one discussed in Example 1 cannot be expressed in monadic Datalog.

We would like to specifically note that, although the rules used in the definitions of MODEPs are in fact monadic Datalog rules, the query evaluation schemes underlying monadic Datalog and MODEQs are fundamentally different: while in the case of monadic Datalog, all elements of the extension can be obtained at once by a forward chaining saturation process on the given interpretation (or database), the *flag & check* strategy that underlies the semantics definition of MODEPs crucially hinges on each potential extension element being verified in a *separate* saturation process. In Section 4.1, we will see that this idea can be captured by Datalog queries, but not by monadic ones.

4. NESTED MODEQS

Query nesting is the process of using an *n*-ary subquery instead of an *n*-ary predicate symbol within a query, with the obvious semantics. In this section, we use this mechanism to extend MODEQs, leading to the more general language of *nested monadically defined queries (NEMODEQs)*. We then show that queries of this type can be expressed in Datalog (Section 4.1) and monadic second-order logic (Section 4.2). These results extend to MODEQs as a special case of NEMODEPs, and help to establish some additional upper bounds for complexity.

It is interesting to ask if nesting of queries actually leads to a new query language or not. A query language is *closed under nesting* if every query with nested subqueries can also be expressed by some non-nested query. Many query languages are trivially closed under nestings since they allow nesting as part of their syntax (e.g, CQs, FOL, MSO, and SO), other languages allow for more or less complex reformulations to eliminate nested queries (e.g., UCQs, Datalog, and monadic Datalog), while others are not closed under nestings (e.g., linear Datalog). We show that MODEQ are not closed under nestings in Example 6 below. This motivates the following definition.

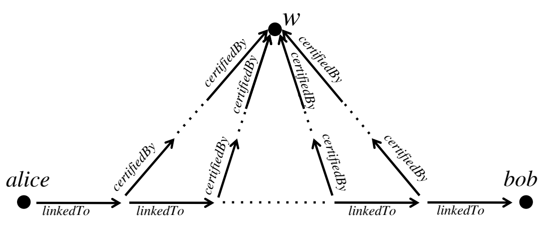

Figure 3: Structure recognized by $\mathfrak{Q}_3$ in Example 5

Definition 6 (NEMODEQ syntax & semantics). *Let* $\mathscr{S}$ *be the underlying signature. A* nested monadically defined predicate (NEMODEP) *of degree* 1 *over* $\mathscr{S}$ *is a MODEP over* $\mathscr{S}$*. Consider a finite set* $\mathbb{P}_i$ $(i = 1, \ldots, k)$ *of NEMODEPs of degree* $\leq d$ *over* $\mathscr{S}$*. A NEMODEP of degree* $d + 1$ *is a MODEP over a signature* $\mathscr{S}'$ *that extends* $\mathscr{S}$ *with additional predicate names* $\mathbb{P}_i$ *that have the same arity as the respective queries.*

The semantics of NEMODEPs of degree $d > 1$ *is defined as for MODEPs, based on the (recursively defined) semantics of NEMODEPs of degree* $< d$*. A* nested monadically defined query (NEMODEQ) *is a conjunctive query that uses both normal predicates and nested monadically defined predicates in its atoms. The semantics of NEMODEQs is defined in the obvious way.*

Note that auxiliary symbols of the form λ_i, $\mathtt{U}_j$, and $\mathtt{hit}$ do not need to be distinct in different subqueries, or in queries and their subqueries. This does not lead to semantic interactions.

Example 5. *Consider again Example 2, and assume that we are now also interested in entities that can certify the security of a communication indirectly, i.e., through a chain of certifications, as shown in Fig. 3. This can be expressed by nesting MODEP* $\mathbb{P}_1$ *in* $\mathfrak{Q}_2$*, leading to a NEMODEQ of degree* 2 $\mathfrak{Q}_3 = \mathbb{P}_3(w)$*, where* $\mathbb{P}_3$ *coincides with* $\mathbb{P}_2$ *except that rule* (6) *is replaced by* $\mathbb{P}_1(x, \lambda_1) \to \mathtt{U}_3(x)$.

However, even if a query is more easily expressed as a NEMODEQ, it might still be possible to do so in a MODEQ. The following example shows that this is the case for $\mathfrak{Q}_3$ above, and presents a query that cannot be expressed by any MODEQ.

Example 6. $\mathfrak{Q}_3$ *of Example 5 can equivalently be expressed by the MODEQ* $\mathfrak{Q}_4[w] = \mathbb{P}_4(w)$*, where* $\mathbb{P}_4$ *consists of the following rules:*

$$\mathit{certifiedBy}(x, \lambda_1) \to \mathtt{U}_3(x) \tag{15}$$

$$\mathtt{U}_3(y) \wedge \mathit{certifiedBy}(x, y) \to \mathtt{U}_3(x) \tag{16}$$

$$\mathit{linkedTo}(\mathit{alice}, x) \to \mathtt{U}_2(x) \tag{17}$$

$$\mathtt{U}_2(x) \wedge \mathtt{U}_3(x) \wedge \mathit{linkedTo}(x, x') \to \mathtt{U}_2(x') \tag{18}$$

$$\mathtt{U}_2(\mathit{bob}) \to \mathtt{hit} \tag{19}$$

To define a NEMODEQ that cannot be expressed as a MODEQ, we first modify this example to ask for all pairs of persons who are connected by a communication chain that is certified by a single entity, i.e., we query for the possible pairs of alice and bob. Let $\mathbb{P}_5$ *be the ternary MODEP that consists of the following rules:*

$$\mathit{certifiedBy}(x, \lambda_1) \to \mathtt{U}_3(x) \tag{20}$$

$$\mathtt{U}_3(y) \wedge \mathit{certifiedBy}(x, y) \to \mathtt{U}_3(x) \tag{21}$$

$$\mathit{linkedTo}(\lambda_2, x) \to \mathtt{U}_2(x) \tag{22}$$

$$\mathtt{U}_2(x) \wedge \mathtt{U}_3(x) \wedge \mathit{linkedTo}(x, x') \to \mathtt{U}_2(x') \tag{23}$$

$$\mathtt{U}_2(\lambda_3) \to \mathtt{hit} \tag{24}$$

We now form a NEMODEQ that asks for all pairs of persons who can communicate through a chain of such secure channels that goes

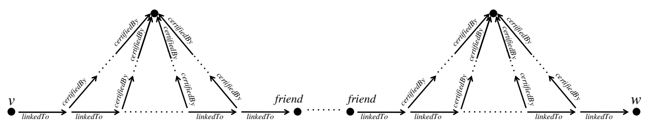

Figure 4: Structure recognized by $\mathfrak{Q}_4$ in Example 6

via multiple people. Moreover, we require that all of these people are "friends," that is, trustworthy in the context of the application. Let $\mathbb{P}_6$ be the binary NEMODEP with the following rules:

$$\rightarrow \mathtt{U}_1(\lambda_1) \tag{25}$$

$$\mathtt{U}_1(y) \wedge \mathbb{P}_5(x, y, z) \wedge \mathit{friend}(z) \rightarrow \mathtt{U}_1(z) \tag{26}$$

$$\mathtt{U}_1(y) \wedge \mathbb{P}_5(x, y, \lambda_2) \rightarrow \mathtt{hit} \tag{27}$$

The NEMODEQ $\mathfrak{Q}_4 = \mathbb{P}_6(v, w)$ (see Fig. 4) cannot be expressed as a MODEQ. To show this, one assumes the existence of such a MODEQ and constructs a database where it must accept a match that is not accepted by $\mathfrak{Q}_4$. A formal proof is given in Proposition 21 in the Appendix.

4.1 Expressing NEMODEQs in Datalog

We now show that NEMODEQs of arbitrary degree can be expressed as Datalog queries. To this end, the auxiliary predicates have to be "contextualized," which increases their arity. Hence the translation does usually not lead to monadic Datalog queries.

Definition 7 (NEMODEQ to Datalog translation). *Given a MODEP $\mathbb{P}$ of arity m, the set* $\mathsf{datalog}(\mathbb{P})$ *of Datalog rules over an extended signature contains, for each rule in $\mathbb{P}$, a new rule obtained by replacing*

- *each constant λ_i with a variable x_{λ_i},*
- *each atom $\mathtt{U}_i(z)$ with the atom $\hat{U}_i(z, x_{\lambda_1}, \ldots, x_{\lambda_m})$ where $\hat{U}_i$ is a fresh predicate of arity $m + 1$,*
- *each atom $\mathtt{hit}$ with the atom $p_{\mathbb{P}}(x_{\lambda_1}, \ldots, x_{\lambda_m})$ where $p_{\mathbb{P}}$ is a fresh predicate symbol of arity m.*

For a NEMODEP $\mathbb{P}$ of degree $d > 1$, let $\mathbb{P}'$ be the MODEP obtained by replacing each direct sub-NEMODEP $\mathbb{Q}$ of $\mathbb{P}$ with the predicate $p_{\mathbb{Q}}$. The Datalog translation of $\mathbb{P}$ is recursively defined as:

$$\mathsf{datalog}(\mathbb{P}) := \mathsf{datalog}(\mathbb{P}') \cup \bigcup\nolimits_{\mathbb{Q} \text{ a direct sub-NEMODEP of } \mathbb{P}} \mathsf{datalog}(\mathbb{Q}).$$

For a NEMODEQ $\mathfrak{Q}[\mathbf{x}] = \exists \mathbf{y}.\varphi[\mathbf{x}, \mathbf{y}]$, the translation $\mathsf{datalog}(\mathfrak{Q})$ *is defined as* $\mathsf{datalog}(\{\varphi[\boldsymbol{\lambda}, \mathbf{y}] \rightarrow \mathtt{hit}\})$*, where the predicate used to replace* $\mathtt{hit}$ *will be denoted by $p_{\mathfrak{Q}}$.*

Note that the predicates $\hat{U}_i$ must be globally fresh, even if multiple subqueries use the same $\mathtt{U}_i$. The rules in $\mathsf{datalog}(\mathfrak{Q})$ might be unsafe, i.e., they may contain universally quantified variables in the head that do not occur in the body. This is no problem with the logical semantics we consider.

Theorem 6 (Datalog expressibility of NEMODEQs). *For every NEMODEQ $\mathfrak{Q}$,* $\mathsf{datalog}(\mathfrak{Q})$ *can be constructed in linear time. Moreover $\mathfrak{Q}[\mathbf{x}]$ and $\langle p_{\mathfrak{Q}}, \mathsf{datalog}(\mathfrak{Q})\rangle$ are equivalent. In particular, the answers for $\mathfrak{Q}[\mathbf{x}]$ and $\langle p_{\mathfrak{Q}}, \mathsf{datalog}(\mathfrak{Q})\rangle$ coincide.*

Example 7. *The Datalog translation for the MODEQ $\mathfrak{Q}_3$ from Example 5 is as follows:*

$\mathsf{datalog}(\mathfrak{Q}_3)$

$\mathsf{datalog}(\mathbb{P}_3)$

$\mathsf{datalog}(\mathbb{P}_1)$

$$\mathit{certifiedBy}(x_{\lambda_1}, y) \rightarrow \hat{U}_1(y, x_{\lambda_1}, x_{\lambda_2})$$
$$\hat{U}_1(y, x_{\lambda_1}, x_{\lambda_2}) \wedge \mathit{certifiedBy}(y, z) \rightarrow \hat{U}_1(z, x_{\lambda_1}, x_{\lambda_2})$$
$$\hat{U}_1(x_{\lambda_2}, x_{\lambda_1}, x_{\lambda_2}) \rightarrow p_{\mathbb{P}_1}(x_{\lambda_1}, x_{\lambda_2})$$

$$p_{\mathbb{P}_1}(x, x_{\lambda_1}) \rightarrow \hat{U}_3(x, x_{\lambda_1})$$
$$\mathit{linkedTo}(\mathit{alice}, x) \rightarrow \hat{U}_2(x, x_{\lambda_1})$$
$$\hat{U}_2(x, x_{\lambda_1}) \wedge \hat{U}_3(x, x_{\lambda_1}) \wedge \mathit{linkedTo}(x, x') \rightarrow \hat{U}_2(x', x_{\lambda_1})$$
$$\hat{U}_2(\mathit{bob}, x_{\lambda_1}) \rightarrow p_{\mathbb{P}_3}(x_{\lambda_1})$$

$$p_{\mathbb{P}_3}(x_{\lambda_1}) \rightarrow p_{\mathfrak{Q}_3}(x_{\lambda_1})$$

Using backward-chaining, the goal $p_{\mathfrak{Q}}(\mathbf{x})$ can be expanded under the rules $\mathsf{datalog}(\mathfrak{Q})$ to obtain a (possibly infinite) set of CQs that do not contain auxiliary predicates $p_{\mathfrak{Q}'}$. Thus $\mathfrak{Q}$ can be considered as a union of (possibly infinitely many) conjunctive queries – an observation that will be useful in Section 5 below.

The linear translation of NEMODEQs (and thus also MODEQs) to Datalog leads to various results. First, NEMODEQs inherit Datalog's polynomial time upper bound for data complexity of query answering [24]. From the results of Section 3.2 we conclude that this bound is tight. Second, we find that the models of NEMODEQs are closed under homomorphisms, since Datalog has this property. Again, this shows that query entailment coincides with model checking (Fact 1).

Theorem 7. *For any NEMODEQ $\mathfrak{Q}[\mathbf{x}]$, the set of models of $\exists \mathbf{x}.\mathfrak{Q}$ is closed under homomorphisms.*

4.2 Expressing NEMODEQs in MSO Logic

In this section, we show that NEMODEQs can also be expressed in *monadic second-order logic (MSO)*, the extension of first-order logic with *set variables*, used like predicates of arity 1. To distinguish them from object variables x, y, z, we denote set variables by the uppercase letter U, possibly with subscripts, hinting at their close relation to the unary coloring predicates $\mathtt{U}$. We adhere to the standard semantics of MSO that we will not repeat here.

To simplify the presentation of the next definition, we henceforth assume that every variable x, constant λ_i, or monadic predicate $\mathtt{U}_j$ is used in at most one (sub-)predicate $\mathbb{P}$ of any NEMODEQ or NEMODEP we consider. This can always be achieved by a suitable renaming of variables and predicates.

Definition 8 (NEMODEQ to MSO translation). *Consider a MODEP $\mathbb{P}$ or arity m with auxiliary unary predicates $\mathtt{U}_1, \ldots, \mathtt{U}_k$, Given a list of terms $\mathbf{t} = \langle t_1, \ldots, t_m\rangle$, we define an MSO formula*

$$\mathsf{mso}(\mathbb{P}(\mathbf{t})) := \forall U_1, \ldots, U_k.\neg \bigwedge_{\rho \in \mathbb{P}} \mathsf{mso}(\rho, \mathbf{t})$$

where $\mathsf{mso}(\rho, \mathbf{t})$ *is the rule obtained from rule ρ by replacing each occurrence of a constant λ_i by t_i, each occurrence of* $\mathtt{hit}$ *by $\bot$ (the falsity atom), and each occurrence of a unary predicate $\mathtt{U}_i$ by a set variable U_i. We extend* mso *to NEMODEPs of higher degree by applying it recursively to NEMODEP atoms. For a NEMODEQ $\mathfrak{Q}$, we obtain* $\mathsf{mso}(\mathfrak{Q})$ *by replacing every NEMODEP atom $\mathbb{P}(\mathbf{t})$ in $\mathfrak{Q}$ by* $\mathsf{mso}(\mathbb{P}(\mathbf{t}))$.

By replacing $\mathtt{hit}$ with $\bot$, the derivation of a query match becomes the derivation of an inconsistency. The formula $\mathsf{mso}(\mathbb{P}(\mathbf{t}))$ evaluates to true if this occurs for all possible interpretations of the predicates $\mathtt{U}_j$, expressed here by universal quantification over the set variables U_j. The interpretation of $\mathbf{t}$ corresponds to the flagged tuple $\boldsymbol{\lambda}$ that is to be checked. It is thus easy to see that the translation captures the semantic conditions of Definitions 1 and 6.

Theorem 8 (MSO expressibility of NEMODEQs). *For every NEMODEQ $\mathfrak{Q}[\mathbf{x}]$, $\mathsf{mso}(\mathfrak{Q}[\mathbf{x}])$ can be constructed in linear time and is equivalent to $\mathfrak{Q}[\mathbf{x}]$.*

Example 8. *For the NEMODEQ $\mathfrak{Q}_3$ from Example 5, $\mathsf{mso}(\mathfrak{Q}_3)$ is the following MSO formula with free variable w*

$$\forall U_2,U_3.\neg\left(\begin{array}{lr} \forall v.\left(\boxed{\forall U_1.\neg\left(\begin{array}{lr} & \forall y.(\mathit{certBy}(v,y)\to U_1(y)) \\ \wedge\forall y,z.(U_1(y)\wedge \mathit{certBy}(y,z)\to U_1(z)) & \\ \wedge & (U_1(w)\to\bot) \end{array}\right)}\to U_3(x)\right) & \\ \wedge & \forall x.(\mathit{linkedTo}(\mathit{alice},x)\to U_2(x)) \\ \wedge & \forall x,x'.(U_2(x)\wedge U_3(x)\wedge \mathit{linkedTo}(x,x')\to U_2(x')) \\ \wedge & (U_2(\mathit{bob})\to\bot) \end{array}\right)$$

where we use certBy to abbreviate certifiedBy. The framed subformula is $\mathsf{mso}(\mathbb{P}_1(v,w))$.

Expressibility of NEMODEQs (and thus also MODEQs) in MSO is a useful feature, which we will further exploit below. For the moment, we just note the direct consequence that the PSpace combined complexity of model checking in MSO directly gives us PSpace-membership of query answering for NEMODEQs and MODEQs.

The following theorem closes the open gaps w.r.t. the combined complexities of query answering by showing PSpace hardness for NEMODEQs by a reduction from the validity problem of quantified Boolean formulae and NP membership for MODEQs by showing that every query match gives rise to a proof the correctness of which can be checked in polynomial time.

Theorem 9 (NEMODEQ answering complexity). *Checking if $\mathbf{c}$ is an answer to a NEMODEQ $\mathfrak{Q}[\mathbf{x}]$ over a database D is* P-*complete in the size of D, and* PSpace-*complete in the size of D and $\mathfrak{Q}$.*

Figure 1 gives an overview of the relationships established so far, regarding both expressivity and complexity. MODEQs feature the same complexities as monadic Datalog, while providing a significant extension of expressivity. The step to NEMODEQs leads to increased combined complexity. Nevertheless, combined complexity is still lower than for Datalog and data complexity is still lower than for MSO. Moreover, in the next sections, we will show that NEMODEQs (and MODEQs) are also more well-behaved than these two when it comes to subsumption checking or interaction with rule sets that give rise to infinite structures.

5. DECIDING NEMODEQ SUBSUMPTION

Checking query subsumption is an essential task in database management, facilitating query optimization, information integration and exchange, and database integrity checking. The *subsumption* or *containment problem* of two queries $\mathfrak{P}$ and $\mathfrak{Q}$ is the question whether the answers of $\mathfrak{Q}$ are contained in the answers of $\mathfrak{P}$ for any underlying database or rule set. Formally, this is the case if the formula $\forall\mathbf{x}.\mathfrak{Q}[\mathbf{x}]\to\mathfrak{P}[\mathbf{x}]$ is valid. In this section, we show that this problem is decidable for NEMODEQs.

An important tool for obtaining this result is the following notion of *treewidth* of an interpretation.

Definition 9 (tree decomposition, treewidth). *Given an interpretation $\mathcal{I}$, a* tree decomposition *of $\mathcal{I}$ is an undirected tree where each node n is associated with a set $\nu(n)\subseteq\Delta^{\mathcal{I}}$ of domain elements such that:*

- *for every tuple $\langle\delta_1,\ldots,\delta_m\rangle\in p^{\mathcal{I}}$ for some predicate p, there is a node n with $\delta_1,\ldots,\delta_m\in\nu(n)$;*
- *for every $\delta\in\Delta^{\mathcal{I}}$, the set of nodes $\{n\mid\delta\in\nu(n)\}$ is connected.*

The width *of a tree decomposition is the maximal cardinality of a set $\nu(n)$. The* treewidth *of $\mathcal{I}$ is the smallest width of any of its tree decompositions.*

We will make use of the following powerful and generic decidability result, implicit in the works of Courcelle [21, 23].

Theorem 10 (Courcelle: MSO decidable over bounded tw). *Satisfiability of monadic second-order logic on countable interpretations of bounded treewidth is decidable.*

An early version of this result has been shown in [21] for a slightly different notion of *width*. A modern account of the relevant proof techniques that uses our above notion of treewidth is given in [23] for the case of finite graphs. Formulating the proof of [21] in these terms, one can show Theorem 10 [22]. We omit the details of this extensive argumentation which is well beyond the scope of the present work.

A set of Datalog rules can be viewed as a (possibly infinite) collection of conjunctive queries that are obtained by expanding rules by repeated backward-chaining. The following definition endows expansions with a useful tree structure.

Definition 10 (Expansion). *Let Σ be a set of Datalog rules with at most one atom in the head. An* expansion tree *is a tree structure where each node is labeled with an atom. Every rule $\rho\in\Sigma$ is associated with an expansion tree $T(\rho)$: The root of $T(\rho)$ is labeled with the head atom of ρ. For each body atom A of ρ, the root of $T(\rho)$ has a direct child node with label A.*

Let κ be an atom. The set $T(\Sigma,\kappa)$ of expansion trees for Σ and κ *is defined inductively:*

1. *The tree that consists of a single node labeled with κ is in $T(\Sigma,\kappa)$.*
2. *Let $T\in T(\Sigma,\kappa)$ with l a leaf node of T labeled $p(\mathbf{t})$ and let $\rho=\forall\mathbf{x}.\varphi\to p(\mathbf{t}')$ be a variant of a rule in Σ where all variables have been renamed to be distinct from variables in T. If θ is the most general unifier of $p(\mathbf{t})$ and $p(\mathbf{t}')$, then an expansion tree T' is obtained from T by replacing l with $T(\rho)$, and by applying the unifier θ to all node labels.*

A partial expansion *of Σ and κ is the conjunction of all leaf labels of some expansion tree of Σ and κ. An* expansion *is a partial expansion that does not contain head predicates of Σ. An expansion of a NEMODEQ $\mathfrak{Q}$ is an expansion of $\mathsf{datalog}(\mathfrak{Q})$ of Definition 7 with atom $\kappa=p_{\mathfrak{Q}}$.*

It is well known that a set of Datalog rules is equivalent to the (possibly infinite) disjunction of its partial expansions, i.e., that an atom $p(\mathbf{c})$ follows from a database D and rules Σ if and only if there is an expansion $\varphi[\mathbf{x},\mathbf{y}]$ for Σ and $p(\mathbf{x})$ such that the query $\exists\mathbf{y}.\varphi[\mathbf{c}/\mathbf{x},\mathbf{y}]$ matches D.

Every expansion $\varphi[\mathbf{x}]$ with variables $\mathbf{x}$ can naturally be associated with an interpretation structure $\mathcal{I}(\varphi)$: its domain $\Delta^{\mathcal{I}(\varphi)}$ is $\mathbf{x}\cup\mathbf{C}$, for each constant $c\in\mathbf{C}$ we set $c^{\mathcal{I}(\varphi)}:=c$, and we have $\mathbf{t}\in p^{\mathcal{I}(\varphi)}$ exactly if $p(\mathbf{t})$ occurs in φ. The treewidth of $\mathcal{I}(\varphi)$ is bounded by the sum $|\mathbf{C}|+n_v$ of the number $|\mathbf{C}|$ of constant symbols and the maximal number n_v of variables in individual rules of Σ.[3] Indeed, the expansion tree can be turned into a tree decomposition by associating

[3]A similar observation has been made by Afrati et al. [2].

each node n with the set of all constant symbols and the variables that occur in the labels of n or any of its direct children. It is easy to verify that this is a tree decomposition. We use $\mathsf{tw}(\Sigma, \kappa) := |\mathbf{C}| + n_v$ to denote the uniform treewidth bound that is thus obtained for expansions of Σ and κ.

This finding can be exploited to show decidability of NEMODEQ subsumption, as it ensures that whenever a non-subsumption witness exists (i.e., a database D with $D \models \mathfrak{Q}$ but $D \not\models \mathfrak{P}$), there is also one with treewidth $\mathsf{tw}(\mathsf{datalog}(\mathfrak{Q}), p_{\mathfrak{Q}})$. Hence, the existence of such a witness is decidable thanks to Theorem 10, since by Theorem 8 the non-subsumption property can be cast into the MSO formula $\exists \mathbf{x}.(\mathsf{mso}(\mathfrak{Q}[x]) \wedge \neg \mathsf{mso}(\mathfrak{P}[x]))$. Thus we obtain the desired decidability result.

Theorem 11 (Decidability of NEMODEQ subsumption). *The query subsumption problem for NEMODEQs is decidable.*

The complexity of NEMODEQ subsumption remains to be determined. An obvious lower bound is the recently shown 2ExpTime hardness of subsumption for monadic Datalog queries [9].

6. QUERING UNDER DEPENDENCIES

Dependencies play an important role in many database applications, be it to formulate constraints, to specify views for data integration, or to define relationships in data exchange. Dependencies can be viewed as logical implications, like the TGDs introduced in Section 2, and one is generally interested in answering queries w.r.t. to their logical entailments. Universal models (Section 2) can be viewed as solutions to data exchange problems or as minimal ways of repairing constraint violations over a database. Querying under dependencies thus corresponds to finding *certain answers*, as is common, e.g., in data integration scenarios.

Example 9. *Consider the following set of TGDs:*

$$\begin{aligned} \mathit{hasAuthor}(x, y) &\rightarrow \mathit{publication}(x) \\ \mathit{cites}(x, y) &\rightarrow \mathit{publication}(x) \wedge \mathit{publication}(y) \\ \mathit{publication}(x) &\rightarrow \exists y.\mathit{hasAuthor}(x, y) \end{aligned}$$

For a database $\{\mathit{hasAuthor}(a, c), \mathit{cites}(a, b)\}$, the conjunctive query $\exists z.(\mathit{hasAuthor}(x, z))$ has $x \mapsto a$ as its only answer. When taking the above dependencies into account, the certain answers additionally contain $x \mapsto b$.

The extension of TGDs with equality is known as *embedded dependencies*, a special case of which are *equality-generating dependencies* [1]. We will not focus on equality here, and state most of our results for TGDs.

The problem of computing CQ answers under TGDs is undecidable in general [18, 8]. A practical approach for computing certain answers is to compute a finite universal model, if possible. All combinations of TGDs and databases have universal models, but it is undecidable whether any such model is finite; the *core chase* is a known semi-decision procedure that computes a finite universal model whenever it exists [25]. Many other variants of the chase have been proposed to compute (finite) universal models in certain cases.

One can compute certain answers under TGDs even in cases where no finite universal model exists, as long as there is a universal model that is sufficiently "regular" to allow for a finite representation. The following property captures one of the most general known criteria for decidability of query answering under TGDs.

Definition 11 (Bounded treewidth set). *A rule set Σ is called* bounded treewidth set (bts) *if for every database D, there is a universal model $\mathcal{I}(D \cup \Sigma)$ of bounded treewidth.*

Note that the treewidth bound in each case depends on Σ and D. Recognizing whether a rule set has this property is undecidable in general [6], but a plethora of sufficient criteria ensuring this property have been identified. Every finite model has an obvious bound on its treewidth – its size – so all TGD sets that guarantee the existence of a finite universal model are in bts. This is trivially fulfilled for TGDs without existential quantifiers, called *full dependencies* or Datalog rules [1]. More elaborate conditions are various notions of *acyclicity* that are based on analysing the interaction of TGDs [26, 27, 39, 40, 6, 36].

Cases where $\mathcal{I}(D \cup \Sigma)$ may be infinite but treewidth-bounded are also manifold. A basic case are *guarded TGDs*, inspired by the guarded fragment of first-order logic [3]. These have been generalized to *weakly guarded TGDs* [11] and *frontier-guarded rules* [6], both of which are subsumed by *weakly frontier-guarded TGDs* [6]. The most expressive currently known bts fragments are *greedy bts TGDs* [7] and *glut-guarded TGDs* [36].

Another type of dependencies comes from the area of *Description Logics* (DLs). Originally conceived as ontology languages, DLs have also been applied to express database constraints [16]. DLs use a different syntax, but share a similar first-order semantics that makes them compatible with TGDs. Most DLs considered in database applications are structurally close to Horn-style rule formalisms (and allow for universal models), and can thus be recast into a rule-based representation [12].

Example 10. *The set of TGDs in Example 9 can equivalently be expressed by the DL-Lite ontology*

$$\begin{aligned} \exists \mathit{hasAuthor} &\sqsubseteq \mathit{publication} \\ \exists \mathit{cites} &\sqsubseteq \mathit{publication} \\ \exists \mathit{cites}^{-} &\sqsubseteq \mathit{publication} \\ \mathit{publication} &\sqsubseteq \exists \mathit{hasAuthor}. \end{aligned}$$

While many DLs enjoy tree-model properties, there are also many expressive DLs that are not bounded treewidth sets in our sense. In particular, this applies to DLs that support transitivity or generalizations thereof [37, 42].

As mentioned before, the entailment problem $D, \Sigma \models Q$ is known to be decidable if Σ is bts and Q is a conjunctive query. Our main result of this section is that this extends to NEMODEQs.

Theorem 12 (NEMODEQ answering under bts). *Let D be a database and let Σ be a set of rules for which the treewidth of $\mathcal{I}(D \cup \Sigma)$ is bounded. Let $\mathfrak{Q}[\mathbf{x}]$ be a NEMODEQ.*

1. *$D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ if and only if $D \cup \Sigma \cup \{\neg \mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ has no countable model with bounded treewidth.*
2. *The problem $D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ is decidable.*

The proof of this theorem exploits universality of $\mathcal{I}(D \cup \Sigma)$ and preservation of NEMODEQ matches under homomorphisms (Theorem 7). The second claim follows from the first by applying Theorem 10 and the observation that the MSO theory $D \cup \Sigma \cup \{\neg \mathsf{mso}(\mathfrak{Q}[\mathbf{c}/\mathbf{x}])\}$ is equivalent to $D \cup \Sigma \cup \{\neg \mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ by Theorem 8.

7. MODEQ REWRITABILITY

Query rewriting is an important technique for answering queries under dependencies, and the main alternative to the chase. The general idea is to find "substitute queries" that can be evaluated directly over the database, without taking TGDs into account, and yet deliver the same answer. In this section, we extend this approach to MODEQs and NEMODEQs, and we establish some basic cases where Datalog queries can be rewritten to MODEQs, which we will extend further in Section 8 below.

The most common case that is usually considered is *first-order rewritability*, where a conjunctive query and a set of TGDs is rewritten into a first-order query – typically a union of conjunctive queries [1]. Importantly, the rewriting does not depend on the underlying database but only on the inital query and TGDs.

Example 11. *Given the rule set from Example 9 and the conjunctive query $\exists v, w.(hasAuthor(v, w))$, an appropriate first-order rewriting is*

$$\exists v, w.(hasAuthor(v, w)) \vee \exists v, w.(cites(v, w)) \vee \exists v.(publication(v)).$$

First-order rewritability is a desirable property since there are very efficient implementations for evaluating first-order queries (that is, SQL) over databases. First-order rewritable sets of TGDs have also called *finite unification sets* [6]. It is undecidable in general whether a set of TGDs belongs to this class, but an iterative backward chaining algorithm can be defined that terminates on FO-rewritable rule sets and provides the rewritten FO formula [6]. Moreover, a significant body of research has unveiled a variety of sufficient syntactically checkable criteria for FO-rewritability. Among the known FO-rewritable TGD fragments are *atomic-hypothesis rules* and *domain restricted rules* [6], *linear Datalog+/–* [12], as well as *sticky sets of TGDs* and *sticky-join sets of TGDs* [13, 14]. More recently these criteria have been found to be subsumed by an efficiently checkable condition that gives rise to the class of *weakly recursive TGDs* [20]. Important first-order rewritable description logics include the DL-Lite family of logics [16]. However, many practically useful TGDs can still not be expressed as first-order queries.

Example 12. *First-order logic cannot express transitive closure, so there cannot be a first-order rewriting for the conjunctive query $\exists v, w.s(v) \wedge r(v, w) \wedge s(w)$ under the TGD $r(x, y) \wedge r(y, z) \rightarrow r(x, z)$.*

This motivates the consideration of more expressive query languages in query rewriting.

Example 13. *The TGD and query of Example 12 can be rewritten as a Datalog query*

$$r(x, y) \rightarrow r_{\text{IDB}}(x, y) \tag{28}$$

$$r_{\text{IDB}}(x, y) \wedge r_{\text{IDB}}(y, z) \rightarrow r_{\text{IDB}}(x, z) \tag{29}$$

$$s(v) \wedge r_{\text{IDB}}(v, w) \wedge s(w) \rightarrow \texttt{goal}, \tag{30}$$

but also as a conjunctive regular path query

$$\exists v, w.s(v) \wedge r^*(v, w) \wedge s(w).$$

Arguably, rewriting CQs under TGDs into Datalog queries is of limited interest only, since the evaluation of Datalog queries remains complex. The fact that subsumption of Datalog queries is undecidable also indicates that it is intrinsically difficult to find out if such a query captures a set of TGDs. The use of C2RPQs is therefore more interesting. Yet, to the best of our knowledge, rewriting of conjunctive queries into C2RPQs has been addressed only very implicitly by now [42]. Moreover, as discussed in Section 3.1, C2RPQs are still rather constrained: besides further structural restrictions, they only allow for recursion over binary predicates. In the following, we therefore consider query rewritability under MODEQs and NEMODEQs, which can be defined as follows:

Definition 12 (NEMODEQ rewritability). *Consider a set Σ of TGDs and a conjunctive query $Q[\mathbf{x}]$. A (NE)MODEQ $\mathfrak{Q}_{Q,\Sigma}$ is a* rewriting *of Q under Σ if, for all databases D and potential query answers $\mathbf{c}$, we have $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ iff $D \models \mathfrak{Q}_{Q,\Sigma}[\mathbf{c}/\mathbf{x}]$. The set Σ is called* (NE)MODEQ-rewritable *if every conjunctive query Q admits a (NE)MODEQ rewriting for Σ and Q.*

Rewritability of conjunctive queries entails rewritability of MODEQs, i.e., the conditions of Definition 12 hold even when considering MODEQs instead of CQs. Indeed, CQs that occur in rule bodies in a MODEQ can generally be replaced using a MODEQ for the respective CQ, provided that the existentially quantified variables in the CQ are not used anywhere else in the rule body:

Lemma 13 (Replacement Lemma). *Consider a set Σ of TGDs, a conjunctive query $Q = \exists \mathbf{y}.\psi[\mathbf{x}, \mathbf{y}]$, and a NEMODEQ $\mathfrak{Q}[\mathbf{x}]$ that is a rewriting for Σ and Q. Then Q and $\mathfrak{Q}$ are equivalent in all models of Σ, i.e., $\Sigma \models \forall \mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathfrak{Q}[\mathbf{x}]$.*

Let $\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}]$ be the conjunction of Q with variables $\mathbf{x}$ replaced by terms $\mathbf{t}$ and variables $\mathbf{y}$ replaced by variables $\mathbf{y}'$. We say that $\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}]$ is a match *in a Datalog rule ρ if ρ is of the form $\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}] \wedge \varphi \rightarrow \chi$ where the $\mathbf{y}'$ occur neither in φ nor in χ.*

Given some NEMODEQ $\mathfrak{P}[\mathbf{z}]$ over Σ, let $\mathfrak{P}'[\mathbf{z}]$ denote a NEMODEQ obtained by replacing a match $\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}]$ of Q in some rule of $\mathfrak{P}$ by $\mathfrak{Q}[\mathbf{t}/\mathbf{x}]$, where we assume w.l.o.g. that the bound variables in $\mathfrak{Q}$ do not occur in $\mathfrak{P}$. Then $\mathfrak{P}$ and $\mathfrak{P}'$ are equivalent in all models of Σ, i.e., $\Sigma \models \forall \mathbf{z}.\mathfrak{P}[\mathbf{z}] \leftrightarrow \mathfrak{P}'[\mathbf{z}]$.

Since every (first-order) conjunctive query can be expressed as MODEQ implies that all first-order rewritable rule sets are also MODEQ-rewritable. This covers all of the first-order rewritable classes mentioned above. Moreover, as an easy corollary of Section 3.2, any set of monadic Datalog rules is MODEQ-rewritable. We have observed that MODEQs are strictly more expressive than monadic Datalog queries, so it does not come as a surprise that one can also find larger classes of MODEQ-rewritable TGDs. An appropriate generalisation of monadic Datalog is as follows:

Definition 13 (j-oriented rule set). *A set of Datalog rules Σ is j-*oriented *for the integer j if all head predicates have the same arity n, and $1 \leq j \leq n$, and we have: if a rule's body contains an atom $p(\mathbf{t})$ for some head predicate p and the rule's head contains an atom $q(\mathbf{t}')$, then $\mathbf{t}$ and $\mathbf{t}'$ agree on all positions other than possibly j.*

Intuitively speaking, recursive derivations in j-oriented rule sets can only modify the content of a single position j while keeping all other arguments fixed in all derived facts.

Example 14. *The following rule set Σ_{family} is 3-oriented. We use atoms $parentsSon(x, y, z)$ and $parentsDghtr(x, y, z)$ to denote that z is the son and daughter of x and y, respectively.*

$$parentsSon(x, y, z) \wedge hasBrother(z, z') \rightarrow parentsSon(x, y, z')$$

$$parentsSon(x, y, z) \wedge hasSister(z, z') \rightarrow parentsDghtr(x, y, z')$$

$$parentsDghtr(x, y, z) \wedge hasBrother(z, z') \rightarrow parentsSon(x, y, z')$$

$$parentsDghtr(x, y, z) \wedge hasSister(z, z') \rightarrow parentsDghtr(x, y, z')$$

Definition 14 (Single predicate rewriting). *For a j-oriented set Σ of Datalog rules and a head predicate p of Σ, a MODEP $\mathbb{P}_{p,\Sigma}$ is defined as follows. Let $\mathtt{U}_q$ be an auxiliary unary predicate for each head predicate q in Σ, let $\mathtt{V}_i$ be a auxiliary unary predicate for each $i \in \{1, \ldots, \mathsf{ar}(p)\}$ with $i \neq j$, and let $\tilde{z}_j$ be an additional variable not occurring in Σ. Then $\mathbb{P}_{p,\Sigma}$ contains the following rules:*

- *a rule $\mathtt{U}_p(\lambda_j) \rightarrow \texttt{hit}$;*

- *for each set variable* $\mathtt{V}_i$*, a rule* $\to \mathtt{V}_i(\lambda_i)$ *with empty body;*
- *for each* $\psi \to q(t_1, \ldots, t_n) \in \Sigma$*, a rule* $\psi' \to \mathtt{U}_q(t_j)$ *where* ψ' *is obtained from* ψ *by replacing each atom of the form* $q'(t_1, \ldots, t'_j, \ldots, t_n)$ *for a head predicate* q' *by* $\mathtt{U}_{q'}(t'_j)$*, and by adding, for each term* t_i *with* $i \neq j$*, a new body atom* $\mathtt{V}_i(t_i)$*;*
- *for each head predicate* q'*, a rule* $q'(\lambda_1, \ldots, \tilde{z}_j, \ldots, \lambda_n) \to \mathtt{U}_{q'}(\tilde{z}_j)$.

For a list $\mathbf{z}$ *of* $\mathsf{ar}(p)$ *variables, the MODEQ* $\mathfrak{Q}_{p,\Sigma}[\mathbf{z}]$ *is defined as* $\mathbb{P}_{p,\Sigma}(\mathbf{z})$.

This operation allows us to express the extension of a predicate p by means of a MODEQ.

Theorem 14 (Single predicate rewriting correctness). *If* Σ *is j-oriented and p is a head predicate, then* $\mathfrak{Q}_{p,\Sigma}[\mathbf{z}]$ *is a rewriting for* Σ *and* $p(\mathbf{z})$.

Example 15. *Considering the* 3*-oriented rule set from Example 14, we obtain* $\mathfrak{Q}_{parentsSon,\Sigma}[z_1, z_2, z_3] = \mathbb{P}_{parentsSon,\Sigma}(z_1, z_2, z_3)$ *with the rules*

$$\begin{aligned}
\mathtt{U}_{parentsSon}(\lambda_3) &\to \mathtt{hit}\\
&\to \mathtt{V}_1(\lambda_1)\\
&\to \mathtt{V}_2(\lambda_2)\\
\mathtt{V}_1(x) \wedge \mathtt{V}_2(y) \wedge \mathtt{U}_{parentsSon}(z) \wedge hasBrother(z, z') &\to \mathtt{U}_{parentsSon}(z')\\
\mathtt{V}_1(x) \wedge \mathtt{V}_2(y) \wedge \mathtt{U}_{parentsSon}(z) \wedge hasSister(z, z') &\to \mathtt{U}_{parentsDghtr}(z')\\
\mathtt{V}_1(x) \wedge \mathtt{V}_2(y) \wedge \mathtt{U}_{parentsDghtr}(z) \wedge hasBrother(z, z') &\to \mathtt{U}_{parentsSon}(z')\\
\mathtt{V}_1(x) \wedge \mathtt{V}_2(y) \wedge \mathtt{U}_{parentsDghtr}(z) \wedge hasSister(z, z') &\to \mathtt{U}_{parentsDghtr}(z')\\
parentsSon(\lambda_1, \lambda_2, \tilde{z}_3) &\to \mathtt{U}_{parentsSon}(\tilde{z}_3)\\
parentsDghtr(\lambda_1, \lambda_2, \tilde{z}_3) &\to \mathtt{U}_{parentsDghtr}(\tilde{z}_3).
\end{aligned}$$

Note that the $\mathtt{V}$ *predicates are not really needed here, since rule bodies do not impose any conditions on the respective variables. In general, one could always replace* $\mathtt{V}$ *by using the respective* λ*s if no constants from* $\mathbf{A}$ *are involved. However, expressions like* $\mathtt{V}_1(c)$ *cannot be handled without introducing an explicit equality predicate that allows us to state* $\lambda_1 \approx a$*. Note that, for the semantics of NEMODEQs to be meaningful, constants* λ_i *must always be allowed to be equal to other constants, even if a unique name assumption is adopted for constants in* $\mathbf{A}$.

Using the Replacement Lemma 13, we can extend Theorem 14 to arbitrary conjunctive queries:

Theorem 15. *Every j-oriented rule set is MODEQ-rewritable.*

8. REWRITING LAYERS OF TGDS

In the previous section, we have identified a first criterion to guarantee MODEQ-rewritability, and thus decidability of the query entailment problem. However, there are many cases where only part of the given TGDs are rewritable. On the other hand, our results from Section 6 guarantee that NEMODEQ answering is still decidable in the presence of TGDs that are in bts, based on techniques that do not require rewriting. We now show that both results can be combined by applying query rewriting to a subset of the given TGDs that is suitably "layered above" the remaining TGDs. This will also allow us to define a class of *fully oriented rule sets* that generalizes *j*-oriented rule sets to cover the full expressiveness of NEMODEQs. More generally, the combination of bts and NEMODEQ-rewritability captures some of the most expressive ontology languages for which query answering is known to be decidable.

Given a set of TGDs, we must first understand which subsets of TGDs could be rewritten into queries that can be evaluated over the remaining TGDs and databases without loosing results. To this end, we consider a notion of *rule dependency*. Related notions were first described in [5], and introduced independently in [25]. Our presentation is most closely related to [6].

Definition 15 (rule dependency, cut). *Let* $\rho_1 = B_1 \to H_1$ *and* $\rho_2 = B_2 \to H_2$ *be two TGDs. We say that* ρ_2 depends on ρ_1*, written* $\rho_1 \prec \rho_2$*, if there is*

- *a database D,*
- *a substitution* θ *of all variables in* B_1 *with terms in D such that* $\theta(B_1) \subseteq D$*, and*
- *a substitution* θ' *of all variables in* B_2 *with terms in* $D \cup \theta(H_1)$ *such that* $\theta'(B_2) \subseteq D \cup \theta(H_1)$ *but* $\theta'(B_2) \nsubseteq D$.

We say that ρ_2 strongly depends on ρ_1*, written* $\rho_1 \ll \rho_2$*, if* H_1 *contains a predicate that occurs in* B_2.

A (strong) cut *of a set of rules* Σ *is a partition* $\Sigma_1 \cup \Sigma_2$ *of* Σ *such that no rule in* Σ_1 *(strongly) depends on a rule in* Σ_2*. It is denoted* $\Sigma_1 \rhd \Sigma_2$ *(*$\Sigma_1 \mathrel{\rhd\!\!\rhd} \Sigma_2$*).*

The notion of rule dependencies encodes which rule can possibly trigger which other rule. Checking if a rule depends on another is an NP-complete task [6]. We thus introduce the simpler notion of strong dependency that can be checked in polynomial time. Clearly, dependency implies strong dependency, but the converse might not be true.

Example 16. *Consider the following Datalog rules:*

$$A(x) \wedge B(x) \to C(x) \tag{31}$$
$$C(x) \to \exists v.p(x, v), A(v) \tag{32}$$

Obviously rule (32) *strongly depends on* (31) *and vice versa. Moreover,* (32) *depends on* (31)*, where the database of Definition 15 can be chosen as* $D = \{A(c), B(c)\}$ *using* $\theta = \theta' = \{x \mapsto c\}$*. However,* (31) *does not depend on* (32)*. Indeed, a substitution* θ' *can map x to v (introduced as a new term when applying* (32)*), but the required fact* $B(v)$ *is not derived by* (32) *and cannot be in any initial database D (since it is not ground).*

Intuitively speaking, we can evaluate a TGD set Σ of the form $\Sigma_1 \rhd \Sigma_2$ by first applying the rules in Σ_1, and then applying the rules of Σ_2. This is the essence of the following theorem, shown in [6].

Theorem 16 (Baget et al.). *Let* Σ *be a set of rules admitting a cut* $\Sigma_1 \rhd \Sigma_2$*. Then, for every database D and every conjunctive query* $Q[\mathbf{x}]$ *we have that* $D \cup \Sigma \models Q[\mathbf{x}/\mathbf{c}]$ *exactly if there is a Boolean conjunctive query* Q' *such that* $D \cup \Sigma_1 \models Q'$ *and* $Q', \Sigma_2 \models Q[\mathbf{x}/\mathbf{c}]$.

This result can be applied to show that query rewriting can proceed in "layers" based on cuts:

Lemma 17 (Query rewriting with cuts). *Let D be a database and let* $\Sigma_1 \rhd \Sigma_2$ *be two sets of TGDs.*

1. *If* $\mathfrak{Q}_{Q,\Sigma_2}[\mathbf{x}]$ *is a NEMODEQ-rewriting of a conjunctive query* $Q[\mathbf{x}]$*, then:*
$$D \cup \Sigma_1 \cup \Sigma_2 \models Q[\mathbf{c}/\mathbf{x}] \text{ if and only if } D \cup \Sigma_1 \models \mathfrak{Q}_{Q,\Sigma_2}[\mathbf{c}/\mathbf{x}].$$
2. *If* Σ_1 *and* Σ_2 *are NEMODEQ-rewritable, then so is* $\Sigma_1 \cup \Sigma_2$.

This observation has two useful implications. The second item outlines an approach of extending and combining rewriting procedures that we will elaborate on in the remainder of this section. The first item hints at a very general approach for constructing TGD languages for which query answering is decidable, as expressed in the next theorem.

THEOREM 18 (QUERY ANSWERING WITH CUTS). *Consider rule sets $\Sigma_1 \triangleright \Sigma_2$, such that NEMODEQ answering is decidable under Σ_1, and NEMODEQ rewriting is decidable under Σ_2. Then NEMODEQ answering is decidable under $\Sigma_1 \cup \Sigma_2$.*

Using Theorem 12, this result specifically applies in cases where Σ_1 is a bounded treewidth set. We have noted that there are a number of effectively checkable criteria for this general class of TGDs. Much less is known about NEMODEQ rewritability beyond rewritability to unions of CQs. For a more general criterion, we can extend j-oriented rules along the lines of Lemma 17 (2).

DEFINITION 16 (FULLY ORIENTED RULE SET). *Let $\approx_{\ll}$ be the reflexive symmetric transitive closure of $\ll$. The set Σ is* fully oriented *if, for every $\rho \in \Sigma$, the equivalence class $[\rho]_{\ll} = \{\rho' \in \Sigma \mid \rho \approx_{\ll} \rho'\}$ is j-oriented (not necessarily for the same j and predicate arity).*

Given a fully oriented rule set, we can construct NEMODEQ rewritings for individual classes $[\rho]_{\ll}$ as in Definition 14, and combine these rewritings using Lemma 13:

THEOREM 19 (FULLY ORIENTED RULE SETS ARE REWRITABLE). *For a rule set Σ, it can be detected in polynomial time if Σ is fully oriented. Every fully oriented set Σ is MODEQ-rewritable.*

The use of $\ll$ instead of $\prec$ is relevant for deciding full orientedness in polynomial time. Even with this restriction, fully oriented Datalog queries have the same expressivity as nested monadically defined queries. Moreover, every NEMODEQ-rewritable TGD set can equivalently be expressed as a set of rules that can be transformed into a MODEQ using Theorem 19.

THEOREM 20 (NEMODEQS = FULLY ORIENTED DATALOG). *For every NEMODEQ $\mathfrak{Q}$, the rule set $\mathsf{datalog}(\mathfrak{Q})$ of Definition 7 is fully oriented. Moreover, for every NEMODEQ-rewritable set Σ of TGDs, there is a fully oriented set of Datalog rules Σ' such that:*

- *every predicate p in Σ has a corresponding head predicate q_p in Σ' that does not occur in Σ,*
- *for every database D and conjunctive query $Q[\mathbf{x}]$ that do not contain predicates of the form q_p, and for every list of constants $\mathbf{c}$, we have $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ iff $D \cup \Sigma' \models Q'[\mathbf{c}/\mathbf{x}]$ where Q' is obtained from Q by replacing all predicates p with q_p.*

Another interesting criterion for NEMODEQ rewritability has been studied for Description Logics (DLs), where all predicates are of arity one or two. Expressive DL ontologies consist of two kinds of terminological axioms: concept inclusion axioms and role inclusion axioms. A role inclusion axiom is a Datalog rule of the form $R_1(x_0, x_1) \wedge \ldots R_n(x_{n-1}, x_n) \rightarrow R(x_0, x_n)$, which can be viewed as a generalized transitivity statement. Even in relatively inexpressive DLs, role inclusion axioms lead to undecidability of CQ answering [37]; in more expressive DLs they even lead to undecidability of simpler reasoning problems [32]. To overcome this, syntactic restrictions are imposed on role inclusions, to ensure that all role inclusions can be captured using finite automata [32, 35]. This is equivalent to rewriting role inclusions to regular path queries, and indeed this has been exploited to decide CQ answering over expressive DLs [42]. This can be viewed as an implicit application of Theorem 18.[4] Reasoning procedures for DLs are often based on tree-like models, so various approaches to DL CQ answering can indeed be viewed as special combinations of bounded treewidth sets and NEMODEQ rewritable sets of TGDs [37, 42]. This supports the practical relevance of the general relationships observed here, and it motivates the further investigation of criteria for NEMODEQ rewritability. The rewriting methods used in DLs are based on regular languages, so it seems promising to consider their generalizations to graph grammars when dealing with arbitrary TGDs [23].

9. CONCLUSION

Monadically defined queries and their nested version achieve a balance between expressivity and computability. They capture and significantly extend the querying capabilities of (unions of) conjunctive queries as well as (unions of) conjunctive two-way regular path queries and monadic Datalog queries, being the prevailing querying paradigms for structured and semi-structured databases. On the other hand, they are conveniently expressible both in Datalog and monadic second-order logic. Yet, as opposed to these two, they ensure decidability of query subsumption and of query answering in the presence of depdendencies that allow for universal models with bounded treewidth – a property shared by many of the known decidable TGD classes.

The novel notions of MODEQ-rewritability and NEMODEQ-rewritability significantly extend first-order rewritability, which has been proven useful for theoretical considerations and practical realization of query answering alike. This extension allows for capturing much larger classes of TGD sets covering features like transitivity which are considered difficult to handle within the known decision frameworks. Furthermore, (NE)MODEQ-rewritable TGD sets can smoothly be integrated with bounded treewidth TGD sets as long as certain dependency constraints are obeyed. This provides a valuable perspective on rule-based data access as a task that can be solved by combining bottom-up techniques like the *chase* with top-down techniques like query rewriting.

Our work raises a number of interesting questions for future research: How general are (NE)MODEQs? Are there larger, more expressive fragments which jointly satisfy all the established properties? Is every rewriting for a TGD set and a given CQ that is expressible in MSO logic equivalent to a NEMODEQ? What is the precise complexity of deciding query subsumption? Which more general syntactic criteria ensure NEMODEQ-rewritability? Can all fragments of TGDs (including those considered in description logics) for which conjunctive query answering is known to be decidable be captured as a combination of bts and NEMODEQ-rewriting? Answering these questions will not only contribute to our understanding of NEMODEQs, but also provide a more unified view on query answering under dependencies in general.

Acknowledgments. We thank Bruno Courcelle and Detlef Seese for very helpful comments on their work on monadic second-order logic and structures of bounded tree-width. We thank Diego Calvanese for helping to clarify complexity issues of C2RPQs.

10. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1994.

[4]DL concept and role inclusion axioms are not always separated by a cut. There are well known rewriting methods to achieve this [34].

[2] F. Afrati, S. Cosmadakis, and E. Foustoucos. Datalog programs and their persistency numbers. *ACM Transactions on Computational Logic*, 6(3):481–518, 2005.

[3] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[4] M. Arenas, S. Conca, and J. Pérez. Counting beyond a Yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, editors, *Proceedings of the 21st World Wide Web Conference (WWW'12)*, pages 629–638. ACM, 2012.

[5] J.-F. Baget. Improving the forward chaining algorithm for conceptual graphs rules. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *KR*, pages 407–414. AAAI Press, 2004.

[6] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9–10):1620–1654, 2011.

[7] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In Walsh [48], pages 712–717.

[8] C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In *Proceedings of the 8th Colloquium on Automata, Languages and Programming*, pages 73–85. Springer, 1981.

[9] M. Benedikt, P. Bourhis, and P. Senellart. Monadic datalog containment. In A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, editors, *ICALP (2)*, volume 7392 of *LNCS*, pages 79–91. Springer, 2012.

[10] G. Brewka and J. Lang, editors. *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*. AAAI Press, 2008.

[11] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In Brewka and Lang [10], pages 70–80.

[12] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In Paredaens and Su [43], pages 77–86.

[13] A. Calì, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *Proceedings of VLDB 2010*, 3(1):554–565, 2010.

[14] A. Calì, G. Gottlob, and A. Pieris. Query answering under non-guarded rules in Datalog+/-. In P. Hitzler and T. Lukasiewicz, editors, *Web Reasoning and Rule Systems*, volume 6333 of *LNCS*, pages 1–17. Springer, 2010.

[15] D. Calvanese. Personal communication, September 2011.

[16] 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. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[17] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. Reasoning on regular path queries. *SIGMOD Record*, 32(4):83–92, 2003.

[18] A. K. Chandra, H. R. Lewis, and J. A. Makowsky. Embedded implicational dependencies and their inference problem. In *Conference Proceedings of the 13th Annual ACM Symposium on Theory of Computation (STOC'81)*, pages 342–354. ACM, 1981.

[19] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In J. E. Hopcroft, E. P. Friedman, and M. A. Harrison, editors, *Proceedings of the 9th Annual ACM Symposium on Theory of Computing (STOC'77)*, pages 77–90. ACM, 1977.

[20] C. Civili and R. Rosati. A broad class of first-order rewritable tuple-generating dependencies. In *Proceedings of the 2nd Workshop on the Resurgence of Datalog in Academia and Industry (Datalog 2.0, 2012)*, LNCS. Springer, 2012. to appear.

[21] B. Courcelle. The monadic second-order logic of graphs, ii: Infinite graphs of bounded width. *Mathematical Systems Theory*, 21(4):187–221, 1989.

[22] B. Courcelle. Personal communication, August 2011.

[23] B. Courcelle and J. Engelfriet. Graph structure and monadic second-order logic, a language theoretic approach. manuscript, to be published at Cambridge University Press; available at http://www.labri.fr/perso/courcell/Book/TheBook.pdf, April 2011.

[24] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[25] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In M. Lenzerini and D. Lembo, editors, *Proc. 27th Symposium on Principles of Database Systems (PODS'08)*, pages 149–158. ACM, 2008.

[26] A. Deutsch and V. Tannen. Reformulation of XML queries and constraints. In D. Calvanese, M. Lenzerini, and R. Motwani, editors, *Proceedings of the 9th International Conference on Database Theory (ICDT 2003)*, volume 2572 of *LNCS*, pages 225–241. Springer, 2003.

[27] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[28] D. Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *Proceedings of the seventeenth ACM symposium on Principles of database systems*, PODS '98, pages 139–148. ACM, 1998.

[29] G. Gottlob and C. Koch. Monadic datalog and the expressive power of languages for web information extraction. *J. ACM*, 51(1):74–113, 2004.

[30] G. Gottlob and C. H. Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comput.*, 183(1):104–122, 2003.

[31] S. Greco and F. Spezzano. Chase termination: A constraints rewriting approach. *Proceedings of VLDB 2010*, 3(1):93–104, 2010.

[32] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67. AAAI Press, 2006.

[33] N. Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16(4):760–778, 1987.

[34] Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In Brewka and Lang [10], pages 274–284.

[35] Y. Kazakov. An extension of complex role inclusion axioms in the description logic $\mathcal{SROIQ}$. In *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR 2010)*, LNCS. Springer, 2010.

[36] M. Krötzsch and S. Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In Walsh [48], pages 963–968.

[37] M. Krötzsch, S. Rudolph, and P. Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In K. Aberer et al., editor, *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, volume 4825 of *LNCS*, pages 310–323. Springer, 2007.

[38] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[39] B. Marnette. Generalized schema-mappings: from termination to tractability. In Paredaens and Su [43], pages 13–22.

[40] M. Meier, M. Schmidt, and G. Lausen. On chase termination beyond stratification. *Proceedings of VLDB 2009*, 2(1):970–981, 2009.

[41] M.-L. Mugnier. Ontological query answering with existential rules. In S. Rudolph and C. Gutierrez, editors, *Web Reasoning and Rule Systems (RR 2011)*, volume 6902 of *LNCS*, pages 2–23. Springer, 2011.

[42] M. Ortiz, S. Rudolph, and M. Simkus. Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In Walsh [48], pages 1039–1044.

[43] J. Paredaens and J. Su, editors. *Proc. 28th Symposium on Principles of Database Systems (PODS'09)*. ACM, 2009.

[44] O. Shmueli. Equivalence of DATALOG queries is undecidable. *J. Log. Program.*, 15(3):231–241, 1993.

[45] L. J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Massachusetts Institute of Technology, 1974.

[46] L. J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.

[47] M. Y. Vardi. The complexity of relational query languages. In H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. H. Landweber, editors, *STOC*, pages 137–146. ACM, 1982.

[48] T. Walsh, editor. *Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*. AAAI Press/IJCAI, 2011.

APPENDIX

This appendix provides proofs that have been omitted in the main paper for reasons of space.

Theorem 2. *For every MODEQ $\mathfrak{Q}[\mathbf{x}]$, the set of models of $\exists\mathbf{x}.\mathfrak{Q}[\mathbf{x}]$ is closed under homomorphism.*

Proof. As MODEQs are a special case of NEMODEQs, the result follows from Theorem 7. □

Theorem 3 (MODEQ answering complexity). *Checking if* $\mathbf{c}$ *is an answer to a MODEQ* $\mathfrak{Q}[\mathbf{x}]$ *over a database* D *is* P*-complete in the size of* D *(data complexity), and* NP*-complete in the size of* D *and* $\mathfrak{Q}$ *(combined complexity).*

Proof. P membership for data complexity is a consequence of the fact that MODEQs are subsumed by NEMODEQs which in turn can be polynomially translated to Datalog queries for which the problem is known to be in P [24, Theorem 4.4].

While P hardness also follows from the fact that MODEQs subsume monadic Datalog (shown later in Theorem 5), we provide a direct proof by reducing entailment in propositional Horn logic to MODEQ answering. Given a set $\mathcal{H}$ of propositional Horn clauses, we introduce for every propositional atom a occurring therein a constant c_a. We also introduce one additional constant *nil*. Moreover, for every Horn clause $C \in \mathcal{H}$ with $C = a_1 \wedge \ldots \wedge a_n \to a$, we introduce constants $b_{C,1}, \ldots, b_{C,n}$ and ground atoms $\mathit{entails}(b_{C,1}, c_a)$, $\mathit{first}(b_{C,i}, c_{a_i})$ for all $i \in \{1, \ldots, n\}$, $\mathit{rest}(b_{C,n}, \mathit{nil})$, and $\mathit{rest}(b_{C,i}, b_{C,i+1})$ for all $i \in \{1, \ldots, n-1\}$. Then the a propositional atom is entailed by $\mathcal{H}$ exactly if it is an answer for the MODEQ $\mathfrak{Q}_{\mathit{true}}[x] = \mathbb{P}(x)$ with $\mathbb{P}$ consisting of the rules

$$\begin{aligned} &\to \mathtt{U}_1(\mathit{nil})\\ \mathit{first}(y,z) \wedge \mathtt{U}_1(z) \wedge \mathit{rest}(y,z') \wedge \mathtt{U}_1(z') &\to \mathtt{U}_1(y)\\ \mathtt{U}_1(y) \wedge \mathit{entails}(y,z) &\to \mathtt{U}_1(z)\\ \mathtt{U}_1(\lambda_1) &\to \mathtt{hit}. \end{aligned}$$

NP hardness for combined complexity is a straightforward consequence of the fact that MODEQs capture CQs for which the problem is known to be NP-hard.

We prove NP membership by showing that each query match gives rise to a witness which can be verified in polynomial time. For a database D and a MODEQ $\mathfrak{Q}[\mathbf{x}] = \exists\mathbf{y}.\psi[\mathbf{x},\mathbf{y}]$ (with ψ being a conjunction containing normal atoms and MODEP atoms), $\mathbf{c}$ is an answer iff there exists a tuple $\mathbf{d}$ of constants, such that $D \models \psi[\mathbf{c},\mathbf{d}]$. As $\psi[\mathbf{c},\mathbf{d}]$ is a conjunction of (normal and MODEP) ground atoms, we can check this for every conjunct separately. For normal ground atoms $p(\mathbf{e})$, we have $D \models p(\mathbf{e})$ iff $p(\mathbf{e}) \in D$, which can be checked in polynomial time. For MODEP ground atoms $\mathbb{P}(\mathbf{e})$, we find that $D \models \mathbb{P}(\mathbf{e})$ iff there is a Datalog derivation of $\mathtt{hit}$ from D via applying rules from $\mathbb{P}[\mathbf{e}/\boldsymbol{\lambda}]$. Due to the arity restriction of head predicates in $\mathbb{P}$ there are only polynomially many atoms that can be derived via $\mathbb{P}[\mathbf{e}/\boldsymbol{\lambda}]$, therefore there must be a proof for $\mathbb{P}(\mathbf{e})$ of polynomial size.

Hence, we define the witness for $\mathbf{c}$ being an answer for D and $\mathfrak{Q}[\mathbf{x}] = \exists\mathbf{y}.\psi[\mathbf{x},\mathbf{y}]$ to contain the tuple $\mathbf{d}$ providing us with bindings to the existentially quantified variables and a polynomial-size proof for every MODEP atom $\mathbb{P}(\mathbf{e})$ contained in $\psi[\mathbf{c},\mathbf{d}]$. Clearly this witness can be verified in polynomial time. □

Theorem 4 (MODEQs capture C2RPQs). *For any C2RPQ* Q*, the MODEQ* $\mathsf{modeq}(Q)$ *can be constructed in linear time, and is equivalent to* Q*, i.e.,* $\forall\mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathsf{modeq}(Q)[\mathbf{x}]$ *is a tautology. In particular, the answers for* Q *and* $\mathsf{modeq}(Q)$ *coincide.*

Proof. The linear construction time follows directly from the definition (taking into account that for every regular expression, the corresponding automaton can be obtained in linear time). The rest of the claim is an immediate consequence of the fact that P and $\mathsf{modep}(P)$ are equivalent for every conjunctive 2-way regular path predicate P. This can be seen as follows (letting $\mathcal{I}$ be an arbitrary interpretation):

To see that $P^{\mathcal{I}} \subseteq \mathsf{modep}(P)^{\mathcal{I}}$ we recap that $P^{\mathcal{I}}$ contains those pairs $\langle\delta,\delta'\rangle$ where δ' can be reached from δ over a path $\delta = \delta_0 \xrightarrow{\gamma_1} \cdots \xrightarrow{\gamma_n} \delta_n = \delta'$ such that $\gamma_1 \ldots \gamma_n$ satisfies the regular expression P. The latter is the case exactly if the corresponding automaton $\mathcal{A}_P$ accepts $\gamma_1 \ldots \gamma_n$, i.e., there is a sequence $\langle s_0, \ldots, s_n\rangle$ of states with s_0 being an initial and s_n being a final state such that $\langle s_{i-1}, \gamma_i, s_i\rangle \in T$ for all $i \in \{1,\ldots,n\}$. We can now show that $\langle\delta,\delta'\rangle$ must be in $\mathsf{modep}(P)^{\mathcal{I}}$ by noting that in every extension $\mathcal{I}'$ of $\mathcal{I}$ satisfying the rules from $\mathsf{modep}(P)$ as well as $\delta = \lambda_1^{\mathcal{I}'}$ and $\delta' = \lambda_2^{\mathcal{I}'}$ (1) we have $\delta \in \mathtt{U}_{s_0}$ due to the according initial-state-rule, (2) we can thus infer by induction that $\delta_i \in \mathtt{U}_{s_i}$ by the according transition rules (3) we hence arrive at $\delta' \in \mathtt{U}_{s_n}$ and can apply the final-state-rule to prove that $\mathtt{hit}$ must hold.

The proof of $\mathsf{modep}(P)^{\mathcal{I}} \subseteq P^{\mathcal{I}}$ is very similar: starting from a derivation sequence for $\mathtt{hit}$, we can easily construct an according path from δ to δ'. □

Theorem 5 (MODEQs capture monadic Datalog). *For every monadic Datalog query* Q*, the MODEQ* $\mathsf{modeq}(Q)$ *can be constructed in linear time, and is equivalent to* Q*, i.e.,* $\forall\mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathsf{modeq}(Q)[\mathbf{x}]$ *is a tautology. In particular, the answers for* Q *and* $\mathsf{modeq}(Q)$ *coincide.*

Proof. The linear construction time follows directly from the definition. For showing the equivalence claim between the monadic Datalog query $Q = (\mathtt{goal}, \mathbb{S})$ and $\mathsf{modeq}(Q)$, we assume w.l.o.g. that all unary IDB predicates in Q are already of the shape $\mathtt{U}_j$ such that the rule sets of $\mathbb{S}$ and $\mathsf{modeq}(Q)$ coincide on all rules not referring to $\mathtt{goal}$ or $\mathtt{hit}$. Given an interpretation $\mathcal{I}$, let $\mathcal{I}''$ denote the extended interpretation obtained by saturating $\mathcal{I}$ under these rules. Then, a tuple $\boldsymbol{\delta}$ from $\mathcal{I}$ is in the extension of Q if $\mathcal{I}'' \models B[\boldsymbol{\delta}/\mathbf{x}]$ for the body of some rule $B \to \mathtt{goal}[x]$ from $\mathbb{S}$. Yet, since $\mathsf{modeq}(Q)$ contains a rule $B[\boldsymbol{\lambda}/\mathbf{x}] \to \mathtt{hit}$ this is exactly the case if $\boldsymbol{\delta}$ is in the extension of $\mathsf{modeq}(Q)$. □

Proposition 21. *The NEMODEQ* $\mathfrak{Q}_4$ *of Example 6 cannot be expressed as a MODEQ.*

Proof. Suppose for a contradiction that there is a MODEQ $\mathfrak{Q}$ that expresses the query $\mathfrak{Q}_4$. Without loss of generality, assume that no MODEP occurs more than once in $\mathfrak{Q}$ (if it does, we can replace it by a copy that is defined by the same rules). Moreover, we assume that the MODEPs in $\mathfrak{Q}$ use distinct names for their auxiliary unary predicates $\mathtt{U}$ and for their constants λ, again without loss of generality. Let ℓ be the maximal number of variables in any of the rules used (in MODEPs) in $\mathfrak{Q}$, and let m be the total number of constants λ (in MODEPs) in $\mathfrak{Q}$ (i.e., the sum of the arities of all MODEPs in $\mathfrak{Q}$).

We first define a structure S of the shape illustrated in Fig. 3 to contain exactly the following relations:

- S contains a chain of relations $a \xrightarrow{\mathit{linkedTo}} e_1 \xrightarrow{\mathit{linkedTo}} \ldots \xrightarrow{\mathit{linkedTo}} e_{2\ell} \xrightarrow{\mathit{linkedTo}} b$;
- for each e_i ($i = 1, \ldots, 2\ell$), S contains a chain $e_i \xrightarrow{\mathit{certifiedBy}} f_{i1} \xrightarrow{\mathit{certifiedBy}} \ldots \xrightarrow{\mathit{certifiedBy}} f_{i\ell} \xrightarrow{\mathit{certifiedBy}} c$;
- the elements a and b are in the unary relation *friend*.

Thus S has $2\ell(\ell+1)+3$ elements. We now use multiple copies of S to define a structure T of the shape that is illustrated in Fig. 4. The structure T consists of $m+1$ copies of S, connected in a chain. Formally, T is defined as the disjoint union of $m+1$ copies $S_0, \ldots, S_m$ of S, factorized by the equivalence relation $\{b_i \approx a_{i+1} \mid 0 \le i < m\}$, where we use a_i and b_i to denote the elements a and b, respectively, from S_i.

Clearly, $\{v \mapsto a_0, w \mapsto b_m\}$ is an answer to $\mathfrak{Q}_4$ over T. Thus, by assumption, $\{v \mapsto a_0, w \mapsto b_m\}$ is also an answer to $\mathfrak{Q}$ over T. This match is therefore entailed in every model of T. By Theorem 2 and Fact 1, we only need to consider the universal model $\mathcal{I}(T)$, the elements of which are in one-to-one correspondence with the elements of T. Thus, the validity of the answer $\{v \mapsto a_0, w \mapsto b_m\}$ is witnessed by a query match on $\mathcal{I}(T)$, which is based on a particular match for each MODEP in $\mathfrak{Q}$. We can consider these matches as a list of m elements of $\mathcal{I}(T)$ (and thus also of T) which were used to interpret the constants λ in $\mathfrak{Q}$. We call these the *matched elements* of T. Since there are only m constants λ in $\mathfrak{Q}$, there are at most m matched elements. Hence, there is some S_o in T which does not contain any matched elements, other than possibly a_o and b_o (since they also belong to adjacent substructures).

We construct a new structure T' from T as follows:

- For each $(e_i)_o$ in the substructure S_o with $i = 2, \ldots, 2\ell$, add a chain $(e_i)_o \xrightarrow{\textit{certifiedBy}} (f'_{i1})_o \xrightarrow{\textit{certifiedBy}} \ldots \xrightarrow{\textit{certifiedBy}} (f'_{i\ell})_o \xrightarrow{\textit{certifiedBy}} c'_o$, where c'_o and the $(f'_{ij})_o$ are fresh elements distinct from any element in T.
- Delete all elements $(f_{(2\ell)1})_o, \ldots, (f_{(2\ell)\ell})_o$.

T' no longer contains a chain of links from a_o to b_o that is certified by a single authority: only the first $2\ell - 1$ elements are certified by c_o, and only the last $2\ell - 1$ elements are certified by c'_o. Likewise, it is not possible to split the chain between a_o and b_o into multiple chains of links, since none of the intermediate elements belongs to the class *friend*. Thus $\{v \mapsto a_0, w \mapsto b_m\}$ is not an answer to $\mathfrak{Q}_4$.

Nevertheless, we can show that $\{v \mapsto a_0, w \mapsto b_m\}$ is an answer to $\mathfrak{Q}$, which yields the required contradiction. To do this, it suffices to show that the matches of MODEPs in $\mathfrak{Q}$ that witness the answer $\{v \mapsto a_0, w \mapsto b_m\}$ of $\mathfrak{Q}$ over T are still valid. Thus consider some MODEP $\mathbb{P}$ in $\mathfrak{Q}$ with auxiliary constants $\lambda_1, \ldots, \lambda_{\mathsf{ar}(\mathbb{P})}$ that have been matched to elements $\boldsymbol{\delta} = \delta_1, \ldots, \delta_{\mathsf{ar}(\mathbb{P})}$ of T (or, equivalently, $\mathcal{I}(T)$). By construction of T', the elements $\boldsymbol{\delta}$ are also in T' and $\mathcal{I}(T')$. We show that $\boldsymbol{\delta} \in \mathbb{P}^{\mathcal{I}(T')}$.

By our assumption, every extension $\mathcal{I}$ of $\mathcal{I}(T)$ with $\lambda_i^{\mathcal{I}} = \delta_i$ and $\mathcal{I} \models \mathbb{P}$ satisfies $\mathtt{hit}$. Hence there is a sequence of applications of rules from $\mathbb{P}$ by which $\mathtt{hit}$ can be derived (using the pre-defined interpretation of λ_i). The same sequence of rules can be used to derive $\mathtt{hit}$ for extensions $\mathcal{I}'$ of $\mathcal{I}(T')$ with $\lambda_i^{\mathcal{I}'} = \delta_i$. Indeed, rules of $\mathbb{P}$ have at most ℓ variables, hence their applicability does only depend on a substructure of size at most ℓ. It is easy to see that $\mathcal{I}(T)$ and $\mathcal{I}(T')$ have the same substructures of size ℓ, up to renaming of elements that are not referred to by any constants in $\mathbb{P}$. It can be shown by induction that this property is preserved when considering unary predicates derived by any number of rule applications. This shows that $\mathcal{I}' \models \mathbb{P}$ implies $\mathcal{I}' \models \mathtt{hit}$. □

Theorem 6 (Datalog expressibility of NEMODEQs). *For every NEMODEQ $\mathfrak{Q}$, $\mathsf{datalog}(\mathfrak{Q})$ can be constructed in linear time. Moreover $\mathfrak{Q}[\mathbf{x}]$ and $\langle p_{\mathfrak{Q}}, \mathsf{datalog}(\mathfrak{Q})\rangle$ are equivalent. In particular, the answers for $\mathfrak{Q}[\mathbf{x}]$ and $\langle p_{\mathfrak{Q}}, \mathsf{datalog}(\mathfrak{Q})\rangle$ coincide.*

Proof. The claimed linear time bound is immediate from the definition.

The equivalence is a straightforward consequence of the following claim, which we will show in the following: $\forall \mathbf{x}.\mathbb{P}(\mathbf{x}) \leftrightarrow (p_{\mathbb{P}}, \mathsf{datalog}(\mathbb{P}))[\mathbf{x}]$ for all NEMODEPs $\mathbb{P}$ of degree ≥ 1. We show this by induction on the degree of $\mathbb{P}$.

We first show $\forall \mathbf{x}.\mathbb{P}(\mathbf{x}) \rightarrow (p_{\mathbb{P}}, \mathsf{datalog}(\mathbb{P}))[\mathbf{x}]$ for all NEMODEPs $\mathbb{P}$. Consider an interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathsf{datalog}(\mathbb{P})$ and a variable assignment $\mathcal{Z}$ such that $\mathcal{I}, \mathcal{Z} \models \mathbb{P}(x_1, \ldots, x_m)$. By definition, the latter means that all $\mathcal{I}'$ with $\lambda_i^{\mathcal{I}'} = \mathcal{Z}(x_i)$ and $\mathcal{I}' \models \mathbb{P}$ must satisfy $\mathtt{hit}$. $(*)$

We define such an $\mathcal{I}'$ as follows:

- $\mathcal{I}'$ has the same domain as $\mathcal{I}$ and the two interpretations coincide on $\mathcal{I}$'s vocabulary,
- $\lambda_i^{\mathcal{I}'} = \mathcal{Z}(x_i)$ for all $i \in \{1, \ldots, m\}$
- $\mathtt{U}_i^{\mathcal{I}'} = \{\delta \mid \langle \delta, \mathcal{Z}(x_1), \ldots, \mathcal{Z}(x_m)\rangle \in \hat{U}_i^{\mathcal{I}}\}$
- $\mathtt{hit}^{\mathcal{I}'} = \{\langle\rangle\}$ if $\langle \mathcal{Z}(x_1), \ldots, \mathcal{Z}(x_m)\rangle \in p_{\mathbb{P}}^{\mathcal{I}}$ and $\mathtt{hit}^{\mathcal{I}'} = \emptyset$ otherwise.

We now show that indeed $\mathcal{I}' \models \rho$ for all $\rho \in \mathbb{P}$. Let $\rho = \forall \mathbf{y}.B_1 \wedge \ldots \wedge B_\ell \rightarrow H$ and assume for some $\mathcal{Z}'$ that $\mathcal{I}', \mathcal{Z}' \models B$ for all $B \in \{B_1, \ldots, B_\ell\}$. Let now $\rho' = \forall \mathbf{y}'.B'_1 \wedge \ldots \wedge B'_\ell \rightarrow H'$ be the rule from $\mathsf{datalog}(\mathbb{P})$ corresponding to ρ and let $\mathcal{Z}'' = \{x_{\lambda_i} \mapsto \lambda_i^{\mathcal{I}'} \mid 1 \le i \le m\}$. Then we find that $\mathcal{I}, \mathcal{Z}'' \cup \mathcal{Z}' \models B'$ for all $B' \in \{B'_1, \ldots, B'_\ell\}$ because

- for B a normal atom, we have $B' = B[\lambda_1/x_1 \ldots \lambda_m/x_m]$ as well as $\lambda_i^{\mathcal{I}', \mathcal{Z}'} = x_i^{\mathcal{I}, \mathcal{Z}'' \cup \mathcal{Z}'}$ for all $i \in \{1, \ldots, m\}$ and the extensions of the corresponding predicate in $\mathcal{I}'$ and $\mathcal{I}$ coincide by construction,
- for B an atom of degree ≥ 1, the coincidence follows from the induction hypothesis,
- for $B = \mathtt{U}_j(t)$, we have $B' = \hat{U}_j(t, x_{\lambda_1}, \ldots, x_{\lambda_m})$ and satisfaction coincides by construction

Now, since $\mathcal{I} \models \mathsf{datalog}(\mathbb{P})$, we obtain $\mathcal{I}, \mathcal{Z} \cup \mathcal{Z}' \models H'$. Therefrom follows $\mathcal{I}', \mathcal{Z}' \models H$ since

- for $H' = \hat{U}_j(t, x_{\lambda_1}, \ldots, x_{\lambda_m})$ we have $H = \mathtt{U}_j(t)$ and satisfaction coincides by construction,
- for $H' = p_{\mathbb{P}}(x_{\lambda_1}, \ldots, x_{\lambda_m})$ we have $H = \mathtt{hit}$ and satisfaction coincides by construction.

Now, having constructed an $\mathcal{I}'$ with $\mathcal{I}'(\lambda_i) = \mathcal{Z}(x_i)$ and $\mathcal{I}' \models \mathbb{P}$ we can infer $\mathcal{I}' \models \mathtt{hit}$ via $(*)$ and therefore $\mathcal{I}, \mathcal{Z} \models p_{\mathbb{P}}(x_1, \ldots, x_m)$, which finishes the first direction.

We proceed by showing $\models \forall \mathbf{x}.(p_{\mathbb{P}}, \mathsf{datalog}(\mathbb{P}))[\mathbf{x}] \rightarrow \mathbb{P}(\mathbf{x})$. To this end, consider an interpretation $\mathcal{I}$ and a variable assignment $\mathcal{Z}$ such that $\mathcal{I}, \mathcal{Z} \models (p_{\mathbb{P}}, \mathsf{datalog}(\mathbb{P}))[\mathbf{x}]$. The latter means that in every extension $\mathcal{J}$ of $\mathcal{I}$ that satisfies all rules from $\mathsf{datalog}(\mathbb{P})$ must also hold $\mathcal{J}, \mathcal{Z} \models p_{\mathbb{P}}(\mathbf{x})$. $(**)$

Let now $\mathcal{I}'$ be an arbitrary extension of $\mathcal{I}$ satisfying $\lambda_i^{\mathcal{I}'} = \mathcal{Z}(x_i)$ for all $i \in \{1, \ldots, m\}$ and $\mathcal{I}' \models \rho$ for all $\rho \in \mathbb{P}$.

We now construct an extension $\mathcal{J}'$ of $\mathcal{I}$, as follows

- $\mathcal{J}'$ has the same domain as $\mathcal{I}$ and the two interpretations coincide on the vocabulary of the latter,
- $\hat{U}_i^{\mathcal{J}'} = \{\langle \delta, \mathcal{Z}(x_1), \ldots, \mathcal{Z}(x_m)\rangle \mid \delta \in \mathtt{U}_i^{\mathcal{I}'}\} \cup \{\langle \delta_0, \delta_1, \ldots, \delta_n\rangle \mid \delta_0 \in \Delta^{\mathcal{I}}, \langle \delta_1, \ldots, \delta_m\rangle \neq \langle \mathcal{Z}(x_1), \ldots, \mathcal{Z}(x_m)\rangle\}$
- $p_{\mathbb{P}}^{\mathcal{J}'} = \Delta^{\mathcal{I}}$ if $\mathtt{hit}^{\mathcal{I}'} = \{\langle\rangle\}$ and $p_{\mathbb{P}}^{\mathcal{J}'} = \Delta^{\mathcal{I}} \setminus \{\langle \mathcal{Z}(x_1), \ldots, \mathcal{Z}(x_m)\rangle\}$ otherwise.

By construction, $\mathcal{J}'$ satisfies $\mathsf{datalog}(\mathbb{P})$ and therefore, by $(**)$, follows $\mathcal{J}', \mathcal{Z} \models p_{\mathbb{P}}(\mathbf{x})$, in other words $\langle \mathcal{Z}(x_1), \ldots, \mathcal{Z}(x_m)\rangle \in p_{\mathbb{P}}^{\mathcal{J}'}$. Again by construction of $\mathcal{J}$, this is the case only if $\mathtt{hit}^{\mathcal{I}'} = \{\langle\rangle\}$, whence we have shown $\mathcal{I}' \models \mathtt{hit}$, which finishes the second direction. □

Theorem 7. *For any NEMODEQ $\mathfrak{Q}[\mathbf{x}]$, the set of models of $\exists\mathbf{x}.\mathfrak{Q}$ is closed under homomorphisms.*

Proof. As established in Theorem 6, NEMODEQs are expressible as Datalog queries. However, for the latter, preservation under homomorphisms is well known and easy to show. □

Theorem 8 (MSO expressibility of NEMODEQs). *For every NEMODEQ $\mathfrak{Q}[\mathbf{x}]$, $\mathsf{mso}(\mathfrak{Q}[\mathbf{x}])$ can be constructed in linear time and is equivalent to $\mathfrak{Q}[\mathbf{x}]$.*

Proof. We show the equivalence for NEMODEPs by an induction on their degree. From this, the general claim is a straightforward consequence.

Let $\mathcal{I}$ be an arbitrary interpretation and $\mathcal{Z}$ a variable assignment for $\mathbf{x}$. Then we can establish the following sequence of equivalent propositions showing the claimed correspondence:

- $\mathcal{I}, \mathcal{Z} \models \mathbb{P}(\mathbf{x})$
- every extension $\mathcal{I}'$ of $\mathcal{I}$ with $\lambda_i^{\mathcal{I}'} = \mathcal{Z}(x_i)$ and $\mathcal{I}' \models \mathbb{P}$ must satisfy $\mathtt{hit}$
- every extension $\mathcal{I}'$ of $\mathcal{I}$ with $\lambda_i^{\mathcal{I}'} = \mathcal{Z}(x_i)$ and $\mathcal{I}' \models \mathbb{P}'$ with $\mathbb{P}' = \mathbb{P}[\bot/\mathtt{hit}]$ must satisfy $\bot$; in other words, no such extension exists
- (via induction hypothesis) no extension $\mathcal{I}'$ of $\mathcal{I}$ with $\lambda_i^{\mathcal{I}'} = \mathcal{Z}(x_i)$ simultaneously satisfies $\mathsf{mso}(\rho)$ for all $\rho \in \mathbb{P}$
- no extension $\mathcal{I}'$ of $\mathcal{I}$ with $\lambda_i^{\mathcal{I}'} = \mathcal{Z}(x_i)$ satisfies $\bigwedge_{\rho\in\mathbb{P}} \mathsf{mso}(\rho)$
- all extensions $\mathcal{I}'$ of $\mathcal{I}$ with $\lambda_i^{\mathcal{I}'} = \mathcal{Z}(x_i)$ satisfy $\neg \bigwedge_{\rho\in\mathbb{P}} \mathsf{mso}(\rho)$
- letting $\mathcal{I}_\lambda$ denote $\mathcal{I}$ extended by $\{\lambda_i \mapsto \mathcal{Z}(x_i) \mid 1 \le i \le m\}$, $\mathcal{I}_\lambda, \Xi \models \neg \bigwedge_{\rho\in\mathbb{P}} \mathsf{mso}(\rho)$ holds for every set-variable assignment $\Xi : \{U_1, \ldots, U_k\} \to 2^{\Delta^{\mathcal{I}}}$
- $\mathcal{I}_\lambda \models \forall U_1, \ldots, U_k.\neg \bigwedge_{\rho\in\mathbb{P}} \mathsf{mso}(\rho)$
- $\mathcal{I}, \mathcal{Z} \models (\forall U_1, \ldots, U_k.\neg \bigwedge_{\rho\in\mathbb{P}} \mathsf{mso}(\rho))[\mathbf{x}/\boldsymbol{\lambda}]$
- $\mathcal{I}, \mathcal{Z} \models \mathsf{mso}(\mathbb{P})[\mathbf{x}]$ □

Theorem 9 (NEMODEQ answering complexity). *Checking if $\mathbf{c}$ is an answer to a NEMODEQ $\mathfrak{Q}[\mathbf{x}]$ over a database D is* P-*complete in the size of D, and* PSpace-*complete in the size of D and $\mathfrak{Q}$.*

Proof. PSpace membership for combined complexity is a direct consequence of PSpace completeness of model checking in monadic second-order logic [45, 47], keeping in mind that due to Fact 1, entailment coincides with model checking if the premise is a set of ground facts.

We show PSpace hardness by providing a reduction from the validity problem of quantified Boolean formulae (QBFs). We recap that for any QBF, we can construct in polynomial time an equivalent QBF of the shape $Q_1x_1Q_2x_2\ldots Q_nx_n \bigvee_{L\in\mathcal{L}} \bigwedge_{\ell\in L} \ell$, with $Q_1, \ldots Q_n \in \{\exists, \forall\}$ and $\mathcal{L}$ being a set of sets of literals over the propositional variables $x_1, \ldots, x_n$. In words, we assume our QBF to be in prenex form with the propositional part of the formula in disjunctive normal form. For every $L = \{x_{k_1}, \ldots, x_{k_i}, \neg x_{k_{i+1}}, \ldots, \neg x_{k_j}\}$, we now define the n-ary MODEP $p_L = \{t(\lambda_{k_1}) \wedge \ldots \wedge t(\lambda_{k_i}) \wedge f(\lambda_{k_{i+1}}) \wedge \ldots \wedge f(\lambda_{k_j}) \to \mathtt{hit}\}$. Moreover, we define the n-ary MODEP $p_{\mathcal{L}} = \{p_L(\lambda_1, \ldots, \lambda_n) \to \mathtt{hit} \mid L \in \mathcal{L}\}$. Letting $p_{\mathcal{L}} = p_n$ we now define MODEPs $p_{n-1} \ldots p_0$ in descending order. If $Q_i = \exists$, then the $i{-}1$-ary MODEP p_{i-1} is defined as the singleton rule set $\{p_{i-1}(\lambda_1, \ldots, \lambda_{i-1}, y) \to \mathtt{hit}\}$. In case $Q_i = \exists$, we let $p_{i-1} = \{p_{i-1}(\lambda_1, \ldots, \lambda_{i-1}, y) \to \mathtt{U}(y) \quad \mathtt{U}(y) \wedge f(y) \wedge \mathtt{U}(z) \wedge t(z) \to hit\}$. Now, let D be the database containing the two individuals 0 and 1 and the facts $f(0)$ and $t(1)$. We now show that the considered QBF is true exactly if $D \models p_0()$. To this end, we first note that by construction the extension of p_L contains exactly those n-tuples $\langle\delta_1, \ldots, \delta_n\rangle$ for which the corresponding truth value assignment val, sending x_i to **true** iff $\delta_i = 1$, makes the formula $\bigwedge_{\ell\in L} \ell$ true. In the same way, the extension of $p_{\mathcal{L}}$ represents the set of truth value assignments satisfying $\bigvee_{L\in\mathcal{L}} \bigwedge_{\ell\in L} \ell$. Then, by descending induction, we can show that the extensions of p_i encode the assignments to free propositional variables of the subformula $Q_{i+1}x_{i+1} \ldots Q_nx_n \bigvee_{L\in\mathcal{L}} \bigwedge_{\ell\in L} \ell$ that make this formula true. Consequently, p_0 has a nonempty extension if the entire considered QBF is true.

For data complexity, P membership follows from expressibility in Datalog shown in Theorem 6, for which the problem is known to be in P [24]. P hardness follows from the respective result for MODEQs established in Theorem 3. □

Theorem 11 (Decidability of NEMODEQ subsumption). *The query subsumption problem for NEMODEQs is decidable.*

Proof. Consider two NEMODEQs $\mathfrak{Q}[\mathbf{x}]$ and $\mathfrak{P}[\mathbf{x}]$. $\mathfrak{P}$ subsumes $\mathfrak{Q}$ if $\forall\mathbf{x}.\mathfrak{Q}[\mathbf{x}] \to \mathfrak{P}[\mathbf{x}]$ is a tautology, i.e., if $\exists\mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ is unsatisfiable.

We show that, if $\exists\mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ is satisfiable, then it has a model of treewidth at most $\mathsf{tw}(\mathsf{datalog}(\mathfrak{Q}), p_{\mathfrak{Q}})$. Thus assume that there is an interpretation $\mathcal{I}$ with $\mathcal{I} \models \exists\mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$. By Theorem 6, there must be an expansion $\varphi[\mathbf{x}, \mathbf{y}]$ of $\mathfrak{Q}[\mathbf{x}]$ such that $\mathcal{I} \models \exists\mathbf{x}, \mathbf{y}.\varphi[\mathbf{x}, \mathbf{y}] \wedge \neg\mathfrak{P}[\mathbf{x}]$. Then there is a variable assignment $\mathcal{Z}$ such that $\mathcal{I}, \mathcal{Z} \models \varphi[\mathbf{x}, \mathbf{y}] \wedge \neg\mathfrak{P}[\mathbf{x}]$. In particular, $\mathcal{I}, \mathcal{Z} \models \varphi[\mathbf{x}, \mathbf{y}]$. This holds exactly if there is a homomorphism π from $\mathcal{I}(\varphi)$ to $\mathcal{I}$ that agrees with $\mathcal{Z}$ on variables, where $\mathcal{I}(\varphi)$ is as defined in Section 5.

By construction, $\mathcal{I}(\varphi), \mathcal{Z}_{\mathrm{id}} \models \varphi[\mathbf{x}, \mathbf{y}]$ where $\mathcal{Z}_{\mathrm{id}}(z) = z$ for each variable z in φ. By Theorem 6, $\mathcal{I}(\varphi), \mathcal{Z}_{\mathrm{id}} \models \mathfrak{Q}[\mathbf{x}]$.

We show that $\mathcal{I}(\varphi), \mathcal{Z}_{\mathrm{id}} \models \neg\mathfrak{P}[\mathbf{x}]$. For a contradiction, suppose that $\mathcal{I}(\varphi), \mathcal{Z}_{\mathrm{id}} \models \mathfrak{P}[\mathbf{x}]$. Since there is a homomorphism π from $\mathcal{I}(\varphi)$ to $\mathcal{I}$, Theorem 7 implies $\mathcal{I}, \pi \circ \mathcal{Z}_{\mathrm{id}} \models \mathfrak{P}[\mathbf{x}]$. By construction of π, the assignment $\pi \circ \mathcal{Z}_{\mathrm{id}}$ agrees with $\mathcal{Z}$ on all variables of $\mathbf{x}$, and thus $\mathcal{I}, \mathcal{Z} \models \mathfrak{P}[\mathbf{x}]$; a contradiction.

We have thus shown that $\mathcal{I}(\varphi), \mathcal{Z}_{\mathrm{id}} \models \mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ and thus $\mathcal{I}(\varphi) \models \exists\mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$. $\mathcal{I}(\varphi)$ is finite, hence countable. Moreover, the treewidth of $\mathcal{I}(\varphi)$ is bounded by $\mathsf{tw}(\mathsf{datalog}(\mathfrak{Q}), p_{\mathfrak{Q}})$. Therefore, if $\exists\mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ is satisfiable, then it is satisfied by a countable model of treewidth at most $\mathsf{tw}(\mathsf{datalog}(\mathfrak{Q}), p_{\mathfrak{Q}})$. On the other hand, the models of $\exists\mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ and $\exists\mathbf{x}.\mathsf{mso}(\mathfrak{Q}[\mathbf{x}]) \wedge \neg\mathsf{mso}(\mathfrak{P}[\mathbf{x}])$ coincide due to Theorem 8 and the latter is an MSO formula. Thus, by Theorem 10, the existence of a model of bounded treewidth can be decided, which finishes the proof. □

Theorem 12 (Decidability of MODEQ answering under bts). *Let D be a database and let Σ be a set of rules for which the treewidth of $\mathcal{I}(D \cup \Sigma)$ is bounded. Let $\mathfrak{Q}[\mathbf{x}]$ be a MODEQ.*

1. *$D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ if and only if $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ has no countable model with bounded treewidth.*
2. *The problem $D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ is decidable.*

Proof. We start by proving the first claim. For the "only if" direction, it is obvious that $D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ implies that $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ can have no model (and therefore no model with bounded treewidth).

For proving the "if" direction, we assume that $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ has no countable model with bounded treewidth. Then it must be unsatisfiable for the following reason: toward a contradiction assume it has a model $\mathcal{I}$. Since $\mathcal{I} \models D \cup \Sigma$, there must be a homomorphism from $\mathcal{I}(D \cup \Sigma)$ to $\mathcal{I}$. Thus, by contraposition of the closedness of models of $\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ under homomorphisms (Theorem 7), $\mathcal{I}(D \cup \Sigma)$ itself satisfies $\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ and is therefore a model of $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$. Moreover, since Σ is bts, we obtain that $\mathcal{I}(D \cup \Sigma)$ has bounded treewidth. Obviously it is also countable, which yields the desired contradiction and ensures unsatisfiability of $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$.

For the second claim we start from the previous claim and note that, since D and Σ are first-order theories and $\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ is expressible as an MSO formula, $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ is expressible as an MSO theory. Thus, we can invoke Theorem 10 to obtain the desired result. □

Lemma 13 (Replacement Lemma). *Consider a set Σ of TGDs, a conjunctive query $Q = \exists\mathbf{y}.\psi[\mathbf{x},\mathbf{y}]$, and a NEMODEQ $\mathfrak{Q}[\mathbf{x}]$ that is a rewriting for Σ and Q. Then Q and $\mathfrak{Q}$ are equivalent in all models of Σ, i.e., $\Sigma \models \forall\mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathfrak{Q}[\mathbf{x}]$.*

Let $\psi[\mathbf{t}/\mathbf{x},\mathbf{y}'/\mathbf{y}]$ be the conjunction of Q with variables $\mathbf{x}$ replaced by terms $\mathbf{t}$ and variables $\mathbf{y}$ replaced by variables $\mathbf{y}'$. We say that $\psi[\mathbf{t}/\mathbf{x},\mathbf{y}'/\mathbf{y}]$ is a match *in a Datalog rule ρ if ρ is of the form $\psi[\mathbf{t}/\mathbf{x},\mathbf{y}'/\mathbf{y}] \wedge \varphi \to \chi$ where the $\mathbf{y}'$ occur neither in φ nor in χ.*

Given some NEMODEQ $\mathfrak{P}[\mathbf{z}]$ over Σ, let $\mathfrak{P}'[\mathbf{z}]$ denote a NEMODEQ obtained by replacing a match $\psi[\mathbf{t}/\mathbf{x},\mathbf{y}'/\mathbf{y}]$ of Q in some rule of $\mathfrak{P}$ by $\mathfrak{Q}[\mathbf{t}/\mathbf{x}]$, where we assume w.l.o.g. that the bound variables in $\mathfrak{Q}$ do not occur in $\mathfrak{P}$. Then $\mathfrak{P}$ and $\mathfrak{P}'$ are equivalent in all models of Σ, i.e., $\Sigma \models \forall\mathbf{z}.\mathfrak{P}[\mathbf{z}] \leftrightarrow \mathfrak{P}'[\mathbf{z}]$.

Proof. We first show that $\Sigma \models \forall\mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathfrak{Q}[\mathbf{x}]$. For the one direction, consider a model $\mathcal{I} \models \Sigma$ and a variable assignment $\mathcal{Z}$ such that $\mathcal{I},\mathcal{Z} \models \mathfrak{Q}[\mathbf{x}]$. According to Theorem 6, there is an expansion $\varphi[\mathbf{y}]$ of $\mathsf{datalog}(\mathfrak{Q})$ such that $\mathcal{I},\mathcal{Z} \models \varphi[\mathbf{y}]$ (where we assume w.l.o.g. that $\mathcal{Z}$ assigns the appropriate domain elements to the fresh variables that φ may contain). Using notation as in Section 5, we find a model $\mathcal{I}(\varphi)$ to which φ matches under the variable assignment $\mathcal{Z}_{\mathrm{id}}$ with $\mathcal{Z}_{\mathrm{id}}(y) = y$ for each y in φ. Then $\mathcal{I}(\varphi),\mathcal{Z}_{\mathrm{id}} \models \mathfrak{Q}[\mathbf{x}]$.

Let $D(\mathcal{I}(\varphi))$ be the model $\mathcal{I}(\varphi)$ considered as a database containing a fact for each of the finitely many relations in $\mathcal{I}(\varphi)$. Introducing finitely many new constants for this purpose is not a problem. Let $\mathbf{c}_\mathbf{x}$ denote the constants in $D(\mathcal{I}(\varphi))$ that correspond to $\mathcal{Z}_{\mathrm{id}}(\mathbf{x})$. Then $D(\mathcal{I}(\varphi)) \models \mathfrak{Q}[\mathbf{c}_\mathbf{x}/\mathbf{x}]$.

Since $\mathfrak{Q}$ is a rewriting of Q under Σ, we have $D(\mathcal{I}(\varphi)),\Sigma \models Q[\mathbf{c}_\mathbf{x}/\mathbf{x}]$. Consider a universal model $\mathcal{J}$ of $D(\mathcal{I}(\varphi)),\Sigma$. Then $\mathcal{J} \models Q[\mathbf{c}_\mathbf{x}/\mathbf{x}]$. Moreover, there is a homomorphism from $\mathcal{J}$ to $\mathcal{I}$. Indeed, the mapping $\mathcal{Z}$ induces a homomorphism π from $\mathcal{I}(\varphi)$ to $\mathcal{I}$. This mapping can be extended to a homomorphism π' from $\mathcal{J}$ to $\mathcal{I}$, since $\mathcal{I}$ is a model of Σ. Due to Theorem 7, the query match $\mathcal{J} \models Q[\mathbf{c}_\mathbf{x}/\mathbf{x}]$ implies $\mathcal{J} \models Q[\pi'(\mathbf{c}_\mathbf{x})/\mathbf{x}]$. Since $\pi'(\mathbf{c}_\mathbf{x}) = \pi(\mathbf{c}_\mathbf{x}) = \mathcal{Z}(\mathbf{x})$, this shows the claim $\mathcal{I},\mathcal{Z} \models Q[\mathbf{x}]$.

The other direction can be shown in a similar way, somewhat simplified due to the fact that one does not need to construct an intermediate model $\mathcal{J}$ of Σ to obtain the match for $\mathfrak{Q}$.

Now the rest of the claim follows from Theorem 6. It remains to show the claimed equivalence for $\mathsf{datalog}(\mathfrak{P})$ and $\mathsf{datalog}(\mathfrak{P}')$. This is a direct consequence of the Replacement Theorem of first-order logic that allows us to replace the sub-formula $\exists\mathbf{y}'.\psi[\mathbf{t}/\mathbf{x},\mathbf{y}'/\mathbf{y}]$ by $p_\mathfrak{Q}(\mathbf{t})$, both of which have just shown to be equivalent. □

Theorem 14 (Single predicate rewriting correctness). *If Σ is j-oriented and p is a head predicate, then $\mathfrak{Q}_{p,\Sigma}[\mathbf{z}]$ is a rewriting for Σ and $p(\mathbf{z})$.*

Proof. We consider the Datalog rewriting $\mathsf{datalog}(\mathfrak{Q}_{p,\Sigma})$ of Definition 7. By Theorem 6, it suffices to show that, for every database D and list of constants $\mathbf{c} = c_1,\ldots,c_n$, we have $D \cup \Sigma \models p(\mathbf{c})$ iff $D,\mathsf{datalog}(\mathfrak{Q}_{p,\Sigma}) \models p_{\mathfrak{Q}_{p,\Sigma}}(\mathbf{c})$.

To establish this, we show that, for every database D, list of constants $\mathbf{c} = c_1,\ldots,c_n$, and predicate $\hat{U}_q$ of $\mathsf{datalog}(\mathfrak{Q}_{p,\Sigma})$, we have $D \cup \Sigma \models q(\mathbf{c})$ iff $D \cup \mathsf{datalog}(\mathfrak{Q}_{p,\Sigma}) \models \hat{U}_q(c_j,c_1,\ldots,c_n)$. The proof is by an easy induction over the derivation of $q(\mathbf{c})$. Clearly, $\mathsf{datalog}(\mathfrak{Q}_{p,\Sigma}) \models \hat{V}_i(\mathbf{d})$ iff $\mathbf{d}$ is of the form $d_i,d_1,\ldots,d_i,\ldots,d_n$. Moreover, whenever there is a rule $\psi \to q(t_1,\ldots,t_n) \in \Sigma$ that has an instance $\psi_c \to q(c_1,\ldots,c_n)$, then $\mathsf{datalog}(\mathfrak{Q}_{p,\Sigma})$ contains a rule $\psi' \to \hat{U}_q(t_j,t_1,\ldots,t_n)$ with an instance $\psi'_c \to \hat{U}_q(c_j,c_1,\ldots,c_n)$. It is easy to verify the claim. □

Theorem 15. *Every j-oriented rule set is MODEQ-rewritable.*

Proof. A rewriting for a j-oriented rule set Σ and a CQ $Q[\mathbf{x}] = \exists\mathbf{y}.p_1(\mathbf{t_1}) \wedge \ldots \wedge p_m(\mathbf{t_m})$ is obtained from $Q[\mathbf{x}]$ by replacing all head atoms $p_i(\mathbf{t_i})$ with the MODEQ $\mathfrak{Q}_{p_i,\Sigma}[\mathbf{t_i}/\mathbf{x}]$ as in Definition 14. We assume a fixed sequence of replacement steps in the construction of the rewriting. Let $\mathfrak{Q}_0$ denote the original query $Q[\mathbf{x}]$, let $\mathfrak{Q}_i$ with $1 \le i$ denote the result after each subsequent replacement step, and let $\mathfrak{Q}$ be the final result.

One can show $\Sigma \models \forall\mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathfrak{Q}_i[\mathbf{x}]$ by induction over i. The base case follows since $\mathfrak{Q}_0$ is clearly equivalent to Q. The induction steps follow from Lemma 13 and Theorem 14.

Hence $\Sigma \models \forall\mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathfrak{Q}[\mathbf{x}]$, i.e., for every interpretation $\mathcal{I} \models \Sigma$ and every variable assignment $\mathcal{Z}$ for $\mathcal{I}$, we have $\mathcal{I},\mathcal{Z} \models Q[\mathbf{x}]$ iff $\mathcal{I},\mathcal{Z} \models \mathfrak{Q}[\mathbf{x}]$ $(*)$.

We show the condition of Definition 12. Thus consider an arbitrary database D and a potential query answer $\mathbf{c}$. Without loss of generality, we assume that D contains no fact of the form $q'(\mathbf{d})$ for some head predicate q' of Σ. Indeed, if it does, we can replace q' by a fresh predicate q'_D and add a rule $\forall\mathbf{x}.q'_D(\mathbf{x}) \to q'(\mathbf{x})$ to Σ. This modification clearly preserves j-orientedness.

If $D \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ then clearly $D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ and thus $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ by $(*)$. Conversely, assume that $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$. Then $D \cup \Sigma$ is satisfiable since Σ contains no constraints (rules with empty head). More precisely, for every interpretation $\mathcal{I} \models D$, there is an interpretation $\mathcal{I}' \models \Sigma, D$ that coincides with $\mathcal{I}$ on all constants and all predicates that are not head predicates in Σ, where we use that D contains no head predicates of Σ. By the assumption $\mathcal{I}' \models Q[\mathbf{c}/\mathbf{x}]$. By $(*)$ $\mathcal{I}' \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$. Besides the auxiliary unary predicates $\mathtt{U}$, $\mathfrak{Q}$ contains only predicates for which $\mathcal{I}$ and $\mathcal{I}'$ agree. Hence $\mathcal{I} \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$. Since $\mathcal{I}$ was arbitrary, this establishes the claim. □

Lemma 17 (Query rewriting with cuts). *Let D be a database and let $\Sigma_1 \rhd \Sigma_2$ be two sets of TGDs.*

1. *If $\mathfrak{Q}_{Q,\Sigma_2}[\mathbf{x}]$ is a NEMODEQ-rewriting of a conjunctive query $Q[\mathbf{x}]$, then:*

$$D \cup \Sigma_1 \cup \Sigma_2 \models Q[\mathbf{c}/\mathbf{x}] \textit{ if and only if } D \cup \Sigma_1 \models \mathfrak{Q}_{Q,\Sigma_2}[\mathbf{c}/\mathbf{x}].$$

2. *If Σ_1 and Σ_2 are NEMODEQ-rewritable, then so is $\Sigma_1 \cup \Sigma_2$.*

Proof. We start by proving the first claim. For the "only if" direction, assume $D \cup \Sigma_1 \cup \Sigma_2 \models Q[\mathbf{c}/\mathbf{x}]$. By Theorem 16 there is a conjunctive query Q' with $D \cup \Sigma_1 \models Q'$ and $Q' \cup \Sigma_2 \models Q[\mathbf{c}/\mathbf{x}]$. By the definition of rewritability, we obtain $Q' \models \mathfrak{Q}_{Q,\Sigma_2}[\mathbf{c}/\mathbf{x}]$. Due to $D \cup \Sigma_1 \models Q'$, we obtain $D \cup \Sigma_1 \models \mathfrak{Q}_{Q,\Sigma_2}[\mathbf{c}/\mathbf{x}]$.

For proving the "if" direction we assume $D \cup \Sigma_1 \models \mathfrak{Q}_{Q,\Sigma_2}[\mathbf{c}/\mathbf{x}]$ and conclude $\mathcal{I}(D \cup \Sigma_1) \models \mathfrak{Q}_{Q,\Sigma_2}[\mathbf{c}/\mathbf{x}]$, which in turn means that there must be an expansion $\varphi[\mathbf{y}]$ of $\mathfrak{Q}_{Q,\Sigma_2}[\mathbf{c}/\mathbf{x}]$ such that $\mathcal{I}(D \cup \Sigma_1) \models \exists\mathbf{y}.\varphi[\mathbf{y}]$. Due to universality of $\mathcal{I}(D \cup \Sigma_1)$ and homomorphism-closedness of models of $\exists\mathbf{y}.\varphi[\mathbf{y}]$, we thus find $D \cup \Sigma_1 \models \exists\mathbf{y}.\varphi[\mathbf{y}]$. Since $\exists\mathbf{y}.\varphi[\mathbf{y}] \models \mathfrak{Q}_{Q,\Sigma_2}[\mathbf{c}/\mathbf{x}]$, we can conclude $\{\exists\mathbf{y}.\varphi[\mathbf{y}]\} \cup \Sigma_2 \models Q[\mathbf{c}/\mathbf{x}]$ by the definition of rewriting. Combining these observations, we obtain $D \cup \Sigma_1 \cup \Sigma_2 \models Q[\mathbf{c}/\mathbf{x}]$.

Consider any conjunctive query $Q[\mathbf{x}]$ and its rewriting $\mathfrak{Q}_{Q,\Sigma_2}$. For every rule ρ in $\mathfrak{Q}_{Q,\Sigma_2}$, the conjunction of all body atoms that do not use any of the auxiliary unary predicates $\mathtt{U}$ can be considered as a conjunctive query Q', that contains no existentially quantified variables. This query constitutes a match for ρ in the sense of Lemma 13. By this lemma, we can thus replace Q' by its rewriting $\mathfrak{Q}_{Q',\Sigma_1}$. The query $\mathfrak{Q}_{Q,\Sigma_1\cup\Sigma_2}$ is obtained by performing this replacement in all rules of $\mathfrak{Q}_{Q,\Sigma_2}$. It is not hard to see that $\mathfrak{Q}_{Q,\Sigma_1\cup\Sigma_2}$ is a rewriting for Q and $\Sigma_1 \cup \Sigma_2$, using a similar argument as in the proof of Theorem 15. □

Theorem 18 (Query answering with cuts). *Consider rule sets $\Sigma_1 \triangleright \Sigma_2$, such that NEMODEQ answering is decidable under Σ_1, and NEMODEQ rewriting is decidable under Σ_2. Then NEMODEQ answering is decidable under $\Sigma_1 \cup \Sigma_2$.*

Proof. This is a direct consequence of Lemma 17 (1). Given any conjunctive query Q, we can compute the rewriting $\mathfrak{Q}_{Q,\Sigma_2}$, and compute its answers over $D \cup \Sigma_1$. □

Theorem 19 (Fully oriented rule sets are rewritable). *For a rule set Σ, it can be detected in polynomial time if Σ is fully oriented. Every fully oriented set Σ is MODEQ-rewritable.*

Proof. Clearly, the relation $\approx_{\ll}$ can be constructed in polynomial time by checking $\rho \ll \rho'$ for each pair of rules and constructing the reflexive symmetric transitive closure. The equivalence classes $[\rho]_{\ll}$ are obtained from this in linear time. It is clear that j-orientedness can be checked for each set of rules in polynomial time.

It remains to show the second part of the claim. We say that $[\rho]_{\ll}$ is *maximal* if, for all $\rho' \in \Sigma$, we have that $\rho \ll \rho'$ implies $\rho' \in [\rho]_{\ll}$. If Σ is fully oriented and $[\rho]_{\ll}$ is maximal, then $\Sigma_1 := \Sigma \setminus [\rho]_{\ll}$ and $\Sigma_2 := [\rho]_{\ll}$ are such that $\Sigma_1 \mathrel{\triangleright\!\!\!\triangleright} \Sigma_2$, and thus $\Sigma_1 \triangleright \Sigma_2$. Moreover, Σ_1 again is fully oriented. Thus, one can apply Theorem 15 and Lemma 17 (2) to obtain the required rewriting. □

Theorem 20 (NEMODEQs = fully oriented Datalog). *For every NEMODEQ $\mathfrak{Q}$, the rule set $\mathsf{datalog}(\mathfrak{Q})$ of Definition 7 is fully oriented. Moreover, for every NEMODEQ-rewritable set Σ of TGDs, there is a fully oriented set of Datalog rules Σ' such that:*

- *every predicate p in Σ has a corresponding head predicate q_p in Σ' that does not occur in Σ,*
- *for every database D and conjunctive query $Q[\mathbf{x}]$ that do not contain predicates of the form q_p, and for every list of constants $\mathbf{c}$, we have $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ iff $D \cup \Sigma' \models Q'[\mathbf{c}/\mathbf{x}]$ where Q' is obtained from Q by replacing all predicates p with q_p.*

Proof. The first part of the claim follows by induction over the degree d of $\mathfrak{Q}$. The case of $d = 1$ follows by observing that the rules with head predicate $p_{\mathfrak{Q}}$ in $\mathsf{datalog}(\mathfrak{Q})$ cannot be $\prec$-smaller than any other rule, and thus form a maximal j-oriented (for any position j in $p_{\mathfrak{Q}}$) subset of $\mathsf{datalog}(\mathfrak{Q})$. Likewise, the set of rules with head of the form $\hat{U}_i(y, \mathbf{z})$ is clearly 1-oriented. To show the claim for $d > 1$, we observe that no rule in $\mathsf{datalog}(\mathfrak{Q})$ is $\prec$-smaller than any rule in $\mathsf{datalog}(\mathfrak{Q}')$ for some subquery $\mathfrak{Q}'$ of $\mathfrak{Q}$. The claim follows by induction.

For the second part of the claim, let $Q_p[\mathbf{y}]$ be the CQ $p(\mathbf{y})$ for each predicate p that is not of the form $q_{p'}$. There is a rewriting $\mathfrak{Q}_p[\mathbf{y}]$ for $Q_p[\mathbf{y}]$ and Σ. By Lemma 13, $\Sigma \models \forall \mathbf{y}.\mathfrak{Q}_p[\mathbf{y}] \leftrightarrow Q_p[\mathbf{y}]$. By Theorem 6, $\models \forall \mathbf{y}.\mathfrak{Q}_p[\mathbf{y}] \leftrightarrow (\mathsf{datalog}(\mathfrak{Q}_p) \to p_{\mathsf{datalog}(\mathfrak{Q}_p)}(\mathbf{y}))$. Thus, $\Sigma \models \forall \mathbf{y}.Q_p[\mathbf{y}] \leftrightarrow (\mathsf{datalog}(\mathfrak{Q}_p) \to p_{\mathsf{datalog}(\mathfrak{Q}_p)}(\mathbf{y}))$.

Using arguments as in the proof of Theorem 15, we find that $D \models \forall \mathbf{y}.(\Sigma \to Q_p[\mathbf{y}]) \leftrightarrow (\mathsf{datalog}(\mathfrak{Q}_p) \to p_{\mathsf{datalog}(\mathfrak{Q}_p)}(\mathbf{y}))$ whenever $D \cup \Sigma$ is consistent. To cover the case that $D \cup \Sigma$ is inconsistent, let $\mathfrak{Q}_\perp$ be a MODEQ without free variables such that, for all databases D', $D' \cup \Sigma$ is inconsistent iff $D' \models \mathfrak{Q}_\perp$. To find such a $\mathfrak{Q}_\perp$, consider $\mathfrak{Q}_p$ for a predicate p that does not occur in Σ and delete from $\mathsf{datalog}(\mathfrak{Q}_p)$ all rules that use the predicate p (these rules check for occurrences of p in the input database). $\mathfrak{Q}_\perp$ can easily be obtained from this.

By the first part of the claim, $\mathsf{datalog}(\mathfrak{Q}_p)$ and $\mathsf{datalog}(\mathfrak{Q}_\perp)$ are fully oriented. We set $q_p := p_{\mathsf{datalog}(\mathfrak{Q}_p)}$. Let $\Sigma_\perp$ be the fully oriented rule set obtained from $\mathsf{datalog}(\mathfrak{Q}_\perp)$ by replacing each rule $\forall \mathbf{y}.\varphi \to$ that has an empty head by new rules $\forall \mathbf{x}, \mathbf{y}.\varphi \to q_p(\mathbf{x})$ for each of the predicates q_p where $\mathbf{x}$ is a list of fresh variables of the appropriate length. Thus, $\Sigma_\perp$ entails all possible facts over predicates q_p from D whenever $D \cup \Sigma$ is inconsistent.

Now we can set $\Sigma' := \Sigma_\perp \cup \bigcup_p \mathsf{datalog}(\mathfrak{Q}_p)$ where we assume w.l.o.g. that any two $\mathsf{datalog}(\mathfrak{Q}_p)$ and $\mathsf{datalog}(\mathfrak{Q}_{p'})$ use mutually disjoint sets of head predicates. It is easy to verify that the claim is satisfied for this choice. □