Dichotomies in Ontology-Mediated Querying with the Guarded Fragment

André Hernich
University of Liverpool
United Kingdom
andre.hernich@liverpool.ac.uk

Carsten Lutz
University of Bremen
Germany
clu@informatik.uni-bremen.de

Fabio Papacchini
University of Liverpool
United Kingdom
fabio.papacchini@liverpool.ac.uk

Frank Wolter
University of Liverpool
United Kingdom
wolter@liverpool.ac.uk

ABSTRACT

We study the complexity of ontology-mediated querying when ontologies are formulated in the guarded fragment of first-order logic (GF). Our general aim is to classify the data complexity on the level of ontologies where query evaluation w.r.t. an ontology $\mathcal{O}$ is considered to be in PTIME if all (unions of conjunctive) queries can be evaluated in PTIME w.r.t. $\mathcal{O}$ and CONP-hard if at least one query is CONP-hard w.r.t. $\mathcal{O}$. We identify several large and relevant fragments of GF that enjoy a dichotomy between PTIME and CONP, some of them additionally admitting a form of counting. In fact, almost all ontologies in the BioPortal repository fall into these fragments or can easily be rewritten to do so. We then establish a variation of Ladner's Theorem on the existence of NP-intermediate problems and use this result to show that for other fragments, there is provably no such dichotomy. Again for other fragments (such as full GF), establishing a dichotomy implies the Feder-Vardi conjecture on the complexity of constraint satisfaction problems. We also link these results to Datalog-rewritability and study the decidability of whether a given ontology enjoys PTIME query evaluation, presenting both positive and negative results.

Keywords

Ontology-Based Data Access; Query Answering; Dichotomies

1. INTRODUCTION

In *Ontology-Mediated Querying*, incomplete data is enriched with an ontology that provides domain knowledge, enabling more complete answers to queries [47, 10, 34]. This paradigm has recently received a lot of interest, a significant fraction of the research being concerned with the (data) complexity of querying [46, 14] and, closely related, with the rewritability of ontology-mediated queries into more conventional database query languages [16, 26, 28, 31, 27]. A particular emphasis has been put on designing ontology languages that result in PTIME data complexity, and in delineating these from the CONP-hard cases. This question and related ones have given rise to a considerable array of ontology languages, including many description logics (DLs) [3, 37] and a growing number of classes of tuple-generating dependencies (TGDs), also known as Datalog$^{\pm}$ and as existential rules [13, 45]. A general and uniform framework is provided by the *guarded fragment (GF)* of first-order logic and extensions thereof, which subsume many of the mentioned ontology languages [5, 6].

In practical applications, ontologies often need to use language features that are only available in computationally expensive ontology languages, but do so in a way such that one may hope for hardness to be avoided. This observation has led to a more fine-grained study of data complexity than on the level of ontology languages, initiated in [42], where the aim is to classify the complexity of individual ontologies while quantifying over the actual query: query evaluation w.r.t. an ontology $\mathcal{O}$ is in PTIME if every CQ can be evaluated in PTIME w.r.t. $\mathcal{O}$ and it is CONP-hard if there is at least one CQ that is CONP-hard to evaluate w.r.t. $\mathcal{O}$. In this way, one can identify tractable ontologies within ontology languages that are, in general, computationally hard. Note that an even more fine-grained approach is taken in [11], where one aims to classify the complexity of each pair $(\mathcal{O}, q)$ with $\mathcal{O}$ an ontology and q an actual query. Both approaches are reasonable, the first one being preferable when the queries to be answered are not fixed at the design time of the ontology; this is actually often the case because ontologies are typically viewed as general purpose artifacts to be used in more than a single application. In this paper, we follow the former approach.

The main aim of this paper is to *identify fragments of GF (and of extensions of GF with different forms of counting) that result in a dichotomy between* PTIME *and* CONP *when used as an ontology language and that cover as many real-world ontologies as possible, considering conjunctive queries (CQs) and unions thereof (UCQs) as the actual query language.* We also aim to provide insight into which fragments of GF (with and without counting) do *not* admit such a dichotomy, to understand the relation between PTIME data complexity and rewritability into Datalog (with inequality in rule bodies, in case we start from GF with equality or counting), and to clarify whether it is decidable whether a given ontology has PTIME data complexity. Note that we concentrate on *fragments* of GF because for the full guarded fragment, proving a dichotomy between PTIME and CONP implies the long-standing Feder-Vardi conjecture on constraint satisfaction problems [23] which indicates that it is very difficult to obtain (if it holds at all). In particular, we concentrate on the fragment of GF that is invariant under disjoint unions, which we call uGF, and on fragments thereof and their ex-

PODS'17, May 14 – 19, 2017, Chicago, IL, USA.

ISBN 978-1-4503-4198-1/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3034786.3056108

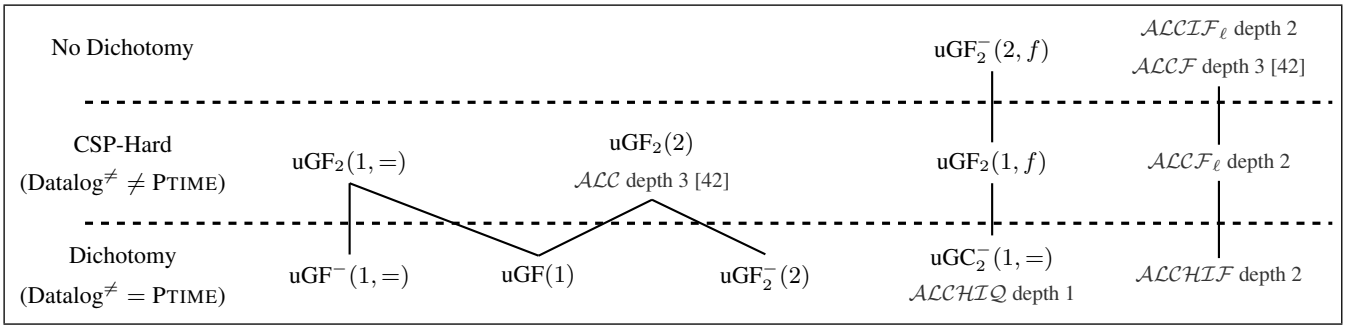


Figure 1: Summary of the results—Number in brackets indicates depth, f presence of partial functions, $\cdot_2$ restriction to two variables, $\cdot^-$ restricts outermost guards to be equality, $\mathcal{F}$ globally function roles, $\mathcal{F}_\ell$ concepts $(\leq 1\, R)$.

tension with forms of counting. Invariance under disjoint unions is a fairly mild restriction that is shared by many relevant ontology languages, and it admits a very natural syntactic characterization.

Our results are summarized in Figure 1. We first explain the fragments shown in the figure and then survey the obtained results. A uGF ontology is a set of sentences of the form $\forall \vec{x}(R(\vec{x}) \rightarrow \varphi(\vec{x}))$ where $R(\vec{x})$ is a guard (possibly equality) and $\varphi(\vec{x})$ is a GF formula that does not contain any sentences as subformulas and in which equality is not used as a guard. The *depth* of such a sentence is the quantifier depth of $\varphi(\vec{x})$ (and thus the outermost universal quantifier is not counted). A main parameter that we vary is the depth, which is typically very small in real world ontologies. In Figure 1, the depth is the first parameter displayed in brackets. As usual, the subscript $\cdot_2$ indicates the restriction to two variables while a superscript $\cdot^-$ means that the guard $R(\vec{x})$ in the outermost universal quantifier can only be equality, $=$ means that equality is allowed (in non-guard positions), f indicates the ability to declare binary relation symbols to be interpreted as partial functions, and GC_2 denotes the two variable guarded fragment extended with counting quantifiers, see [32, 48]. While guarded fragments are displayed in black, description logics (DLs) are shown in grey and smaller font size. We use standard DL names except that '$\mathcal{F}$' denotes globally functional roles while '$\mathcal{F}_\ell$' refers to counting concepts of the form $(\leq 1\, R)$. We do not explain DL names here, but refer to the standard literature [4].

The bottommost part of Figure 1 displays fragments for which there is a dichotomy between PTIME and CONP, the middle part shows fragments for which such a dichotomy implies the Feder-Vardi conjecture (from now on called CSP-hardness), and the topmost part is for fragments that provably have no dichotomy (unless PTIME = NP). The vertical lines indicate that the linked results are closely related, often indicating a fundamental difficulty in further generalizing an upper bound. For example, uGF$^-(1,=)$ enjoys a dichotomy while uGF$_2(1,=)$ is CSP-hard, which demonstrates that generalizing the former result by dropping the restriction that the outermost quantifier has to be equality (indicated by $\cdot^-$) is very challenging (if it is possible at all).[1] Our positive results are thus optimal in many ways. All results hold both when CQs and when UCQs are used as the actual query; in this context, it is interesting to note that there is a GF ontology (which is not an uGF ontology) for which CQ answering is in PTIME while UCQ-answering is CONP-hard. In the cases which enjoy a dichotomy, we also show that PTIME query evaluation coincides with rewritability into Datalog (with inequality in the rule bodies if we start from a fragment with equality or counting). In contrast, for all fragments that are CSP-hard or have no dichotomy, these two properties do provably not coincide. This is of course independent of whether or not the Feder-Vardi conjecture holds.

For $\mathcal{ALCHIQ}$ ontologies of depth 1, we also show that it is decidable and EXPTIME-complete whether a given ontology admits PTIME query evaluation (equivalently: rewritability into Datalog$^{\neq}$). For uGC$_2^-(1,=)$, we show a NEXPTIME upper bound. For $\mathcal{ALC}$ ontologies of depth 2, we establish NEXPTIME-hardness. The proof indicates that more sophisticated techniques are needed to establish decidability, if the problem is decidable at all (which we leave open).

To understand the practical relevance of our results, we have analyzed 411 ontologies from the BioPortal repository [52]. After removing all constructors that do not fall within $\mathcal{ALCHIF}$, an impressive 405 ontologies turned out to have depth 2 and thus belong to a fragment with dichotomy (sometimes modulo an easy complexity-preserving rewriting). For $\mathcal{ALCHIQ}$, still 385 ontologies had depth 1 and so belonged to a fragment with dichotomy. As a concrete and simple example, consider the two uGC$_2^-(1)$-ontologies

$$\mathcal{O}_1 = \{\forall x\, (\mathsf{Hand}(x) \rightarrow \exists^{=5} y\, \mathsf{hasFinger}(x,y))\}$$
$$\mathcal{O}_2 = \{\forall x\, (\mathsf{Hand}(x) \rightarrow \exists y\, (\mathsf{hasFinger}(x,y) \wedge \mathsf{Thumb}(y)))\}$$

which both enjoy PTIME query evaluation (and thus rewritability into Datalog$^{\neq}$), but where query evaluation w.r.t. the union $\mathcal{O}_1 \cup \mathcal{O}_2$ is CONP-hard. Note that such subtle differences cannot be captured when data complexity is studied on the level of ontology languages, at least when basic compositionality conditions are desired.

We briefly highlight some of the techniques used to establish our results. An important role is played by the notions of materializability and unraveling tolerance of an ontology $\mathcal{O}$. Materializability means that for every instance $\mathfrak{D}$, there is a universal model of $\mathfrak{D}$ and $\mathcal{O}$, defined in terms of query answers rather than in terms of homomorphisms (which, as we show, need not coincide in our context). Unraveling tolerance means that the ontology cannot distinguish between an instance and its unraveling into a structure of bounded treewidth. While non-materializability of $\mathcal{O}$ implies that query evaluation w.r.t. $\mathcal{O}$ is CONP-hard, unraveling tolerance of $\mathcal{O}$ implies that query evaluation w.r.t. $\mathcal{O}$ is in PTIME (in fact, even rewritable into Datalog). To establish dichotomies, we prove for the relevant fragments that materializability implies unraveling tolerance which, depending on the fragment, can be technically rather subtle. To prove CSP-hardness or non-dichotomies, very informally speaking, we need to express properties in the ontology that a (positive existential) query cannot 'see'. This is often very subtle and can often be achieved only partially. While the latter is not

[1] A tentative proof of the Feder-Vardi conjecture has very recently been announced in [49], along with an invitation to the research community to verify its validity.

a major problem for CSP-hardness (where we need to deal with CSPs that 'admit precoloring' and are known to behave essentially in the same way as traditional CSPs), it poses serious challenges when proving non-dichotomy. To tackle this problem, we establish a variation of Ladner's theorem on NP-intermediate problems such that instead of the word problem for NP Turing machines, it speaks about the run fitting problem, which is to decide whether a given partially described run of a Turing machine (which corresponds to a precoloring in the CSP case) can be extended to a full run that is accepting. Also our proofs of decidability of whether an ontology admits PTIME query evaluation are rather subtle and technical, involving e.g. mosaic techniques.

Due to space constraints, throughout the paper we defer proof details to the long version this paper available at http://cgi.csc.liv.ac.uk/~frank/publ/publ.html.

Related Work. Ontology-mediated querying has first been considered in [40, 15]; other important papers include [16, 12, 5]. It is a form of reasoning under integrity constraints, a traditional topic in database theory, see e.g. [9, 8] and references therein, and it is also related to deductive databases, see e.g. the monograph [44]. Moreover, ontology-mediated querying has drawn inspiration from query answering under views [17, 18]. In recent years, there has been significant interest in complete classification of the complexity of hard querying problems. In the context of ontology-mediated querying, relevant references include [42, 11, 41]. In fact, this paper closes a number of open problems from [42] such as that $\mathcal{ALCI}$ ontologies of depth two enjoy a dichotomy and that materializability (and thus PTIME complexity and Datalog-rewritability) is decidable in many relevant cases. Other areas of database theory where complete complexity classifications are sought include consistent query answering [36, 35, 24, 21], probabilistic databases [51], and deletion propagation [33, 25].

2. PRELIMINARIES

We assume an infinite set Δ_D of data constants, an infinite set Δ_N of labeled nulls disjoint from Δ_D, and a set Σ of relation symbols containing infinitely many relation symbols of any arity ≥ 1. A *(database) instance* $\mathfrak{D}$ is a non-empty set of *facts* $R(a_1, \ldots, a_k)$, where $R \in \Sigma$, k is the arity of R, and $a_1, \ldots, a_k \in \Delta_D$. We generally assume that instances are finite, unless otherwise specified. An *interpretation* $\mathfrak{A}$ is a non-empty set of *atoms* $R(a_1, \ldots, a_k)$, where $R \in \Sigma$, k is the arity of R, and $a_1, \ldots, a_k \in \Delta_D \cup \Delta_N$. We use $\mathsf{sig}(\mathfrak{A})$ and $\mathsf{dom}(\mathfrak{A})$ to denote the set of relation symbols and, respectively, constants and labelled nulls in $\mathfrak{A}$. We always assume that $\mathsf{sig}(\mathfrak{A})$ is finite while $\mathsf{dom}(\mathfrak{A})$ can be infinite. Whenever convenient, interpretations $\mathfrak{A}$ are presented in the form $(A, (R^{\mathfrak{A}})_{R \in \mathsf{sig}(\mathfrak{A})})$ where $A = \mathsf{dom}(\mathfrak{A})$ and $R^{\mathfrak{A}}$ is a k-ary relation on A for each $R \in \mathsf{sig}(\mathfrak{A})$ of arity k. An interpretation $\mathfrak{A}$ is a *model* of an instance $\mathfrak{D}$, written $\mathfrak{A} \models \mathfrak{D}$, if $\mathfrak{D} \subseteq \mathfrak{A}$. We thus make a strong open world assumption (interpretations can make true additional facts and contain additional constants and nulls) and also assume *standard names* (every constant in $\mathfrak{D}$ is interpreted as itself in $\mathfrak{A}$). Note that every instance is also an interpretation.

Assume $\mathfrak{A}$ and $\mathfrak{B}$ are interpretations. A *homomorphism* h from $\mathfrak{A}$ to $\mathfrak{B}$ is a mapping from $\mathsf{dom}(\mathfrak{A})$ to $\mathsf{dom}(\mathfrak{B})$ such that $R(a_1, \ldots, a_k) \in \mathfrak{A}$ implies $R(h(a_1), \ldots, h(a_k)) \in \mathfrak{B}$ for all $a_1, \ldots, a_k \in \mathsf{dom}(\mathfrak{A})$ and $R \in \Sigma$ of arity k. We say that h *preserves a set* D of constants and labelled nulls if $h(a) = a$ for all $a \in D$ and that h is an *isomorphic embedding* if it is injective and $R(h(a_1), \ldots, h(a_k)) \in \mathfrak{B}$ entails $R(a_1, \ldots, a_k) \in \mathfrak{A}$. An interpretation $\mathfrak{A} \subseteq \mathfrak{B}$ is a *subinterpretation* of $\mathfrak{B}$ if $R(a_1, \ldots, a_k) \in \mathfrak{B}$ and $a_1, \ldots, a_k \in \mathsf{dom}(\mathfrak{A})$ implies $R(a_1, \ldots, a_k) \in \mathfrak{A}$; if $\mathsf{dom}(\mathfrak{A}) = A$, we denote $\mathfrak{A}$ by $\mathfrak{B}_{|A}$ and call it *the subinterpretation of* $\mathfrak{B}$ *induced by* A.

Conjunctive queries (CQs) q of arity k take the form $q(\vec{x}) \leftarrow \phi$, where $\vec{x} = x_1, \ldots, x_k$ is the tuple of *answer variables* of q, and ϕ is a conjunction of *atomic formulas* $R(y_1, \ldots, y_n)$ with $R \in \Sigma$ of arity n and $y_1, \ldots, y_n$ variables. As usual, all variables in $\vec{x}$ must occur in some atom of ϕ. Any CQ $q(\vec{x}) \leftarrow \phi$ can be regarded as an instance $\mathfrak{D}_q$, often called the *canonical database of* q, in which each variable y of ϕ is represented by a unique data constant a_y, and that for each atom $R(y_1, \ldots, y_k)$ in ϕ contains the atom $R(a_{y_1}, \ldots, a_{y_k})$. A tuple $\vec{a} = (a_1, \ldots, a_k)$ of constants is an *answer to* $q(x_1, \ldots, x_k)$ *in* $\mathfrak{A}$, in symbols $\mathfrak{A} \models q(\vec{a})$, if there is a homomorphism h from $\mathfrak{D}_q$ to $\mathfrak{A}$ with $h(a_{x_1}, \ldots, a_{x_k}) = \vec{a}$. A *union of conjunctive queries (UCQ)* q takes the form $q_1(\vec{x}), \ldots, q_n(\vec{x})$, where each $q_i(\vec{x})$ is a CQ. The q_i are called *disjuncts* of q. A tuple $\vec{a}$ of constants is an *answer* to q in $\mathfrak{A}$, denoted by $\mathfrak{A} \models q(\vec{a})$, if $\vec{a}$ is an answer to some disjunct of q in $\mathfrak{A}$.

We now introduce the fundamentals of ontology-mediated querying. An *ontology language* $\mathcal{L}$ is a set of first-order sentences over signature Σ (that is, function symbols are not allowed) and an *$\mathcal{L}$-ontology* $\mathcal{O}$ is a finite set of sentences from $\mathcal{L}$. We introduce various concrete ontology languages throughout the paper, including fragments of the guarded fragment and descriptions logics. An interpretation $\mathfrak{A}$ is a *model of an ontology* $\mathcal{O}$, in symbols $\mathfrak{A} \models \mathcal{O}$, if it satisfies all its sentences. An instance $\mathfrak{D}$ is *consistent w.r.t.* $\mathcal{O}$ if there is a model of $\mathfrak{D}$ and $\mathcal{O}$.

An *ontology-mediated query (OMQ)* is a pair $(\mathcal{O}, q)$, where $\mathcal{O}$ is an ontology and q a UCQ. The semantics of an ontology-mediated query is given in terms of *certain answers*, defined next. Assume that q has arity k and $\mathfrak{D}$ is an instance. Then a tuple $\vec{a}$ of length k in $\mathsf{dom}(\mathfrak{D})$ is a *certain answer to q on an instance $\mathfrak{D}$ given $\mathcal{O}$*, in symbols $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$, if $\mathfrak{A} \models q(\vec{a})$ for all models $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$. The *query evaluation problem for an OMQ* $(\mathcal{O}, q(\vec{x}))$ is to decide, given an instance $\mathfrak{D}$ and a tuple $\vec{a}$ in $\mathfrak{D}$, whether $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$.

We use standard notation for Datalog programs (a brief introduction is given in the appendix). An OMQ $(\mathcal{O}, q(\vec{x}))$ is called *Datalog-rewritable* if there is a Datalog program Π such that for all instances $\mathfrak{D}$ and $\vec{a} \in \mathsf{dom}(\mathfrak{D})$, $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ iff $\mathfrak{D} \models \Pi(\vec{a})$. *Datalog$^{\neq}$-rewritability* is defined accordingly, but allows the use of inequality in the body of Datalog rules. We are mainly interested in the following properties of ontologies.

Definition 1 *Let $\mathcal{O}$ be an ontology and $\mathcal{Q}$ a class of queries. Then*

- *$\mathcal{Q}$-evaluation w.r.t. $\mathcal{O}$ is in* PTIME *if for every $q \in \mathcal{Q}$, the query evaluation problem for $(\mathcal{O}, q)$ is in* PTIME.
- *$\mathcal{Q}$-evaluation w.r.t. $\mathcal{O}$ is* Datalog-rewritable *(resp.* Datalog$^{\neq}$-rewritable*) if for every $q \in \mathcal{Q}$, the query evaluation problem for $(\mathcal{O}, q)$ is Datalog-rewritable (resp. Datalog$^{\neq}$-rewritable).*
- *$\mathcal{Q}$-evaluation w.r.t. $\mathcal{O}$ is* CONP-hard *if there is a $q \in \mathcal{Q}$ such that the query evaluation problem for $(\mathcal{O}, q)$ is* CONP-*hard.*

2.1 Ontology Languages

As ontology languages, we consider fragments of the guarded fragment (GF) of FO, the two-variable guarded fragment of FO with counting, and DLs. Recall that GF formulas [1] are obtained by starting from atomic formulas $R(\vec{x})$ over Σ and equalities $x = y$ and then using the boolean connectives and *guarded quantifiers* of the form

$$\forall \vec{y}(\alpha(\vec{x}, \vec{y}) \rightarrow \varphi(\vec{x}, \vec{y})), \quad \exists \vec{y}(\alpha(\vec{x}, \vec{y}) \wedge \varphi(\vec{x}, \vec{y}))$$

where $\varphi(\vec{x},\vec{y})$ is a guarded formula with free variables among $\vec{x},\vec{y}$ and $\alpha(\vec{x},\vec{y})$ is an atomic formula or an equality $x = y$ that contains all variables in $\vec{x},\vec{y}$. The formula α is called the *guard of the quantifier*.

In ontologies, we only allow GF sentences φ that are *invariant under disjoint unions*, that is, for all families $\mathfrak{B}_i$, $i \in I$, of interpretations with mutually disjoint domains, the following holds: $\mathfrak{B}_i \models \varphi$ for all $i \in I$ if, and only if, $\bigcup_{i\in I} \mathfrak{B}_i \models \varphi$. We give a syntactic characterization of GF sentences that are invariant under disjoint unions. Denote by *openGF* the fragment of GF that consists of all (open) formulas whose subformulas are all open and in which equality is not used as a guard. The fragment *uGF* of GF is the set of sentences obtained from openGF by a *single guarded universal quantifier*: if $\varphi(\vec{y})$ is in openGF, then $\forall\vec{y}(\alpha(\vec{y}) \to \varphi(\vec{y}))$ is in uGF, where $\alpha(\vec{y})$ is an atomic formula or an equality $y = y$ that contains all variables in $\vec{y}$. We often omit equality guards in uGF sentences of the form $\forall y(y = y \to \varphi(y))$ and simply write $\forall y\varphi$. A *uGF ontology* is a finite set of sentences in uGF.

Theorem 1 *A GF sentence is invariant under disjoint unions iff it is equivalent to a uGF sentence.*

PROOF. The direction from right to left is straightforward. For the converse direction, observe that every GF sentence is equivalent to a Boolean combination of uGF sentences. Now assume that φ is a GF sentence and invariant under disjoint unions. Let $\text{cons}(\varphi)$ be the set of all sentences χ in uGF with $\varphi \models \chi$. By compactness of FO it is sufficient to show that $\text{cons}(\varphi) \models \varphi$. If this is not the case, take a model $\mathfrak{A}_0$ of $\text{cons}(\varphi)$ refuting φ and take for any sentence ψ in uGF that is not in $\text{cons}(\varphi)$ an interpretation $\mathfrak{A}_{\neg\psi}$ satisfying φ and refuting ψ. Let $\mathfrak{A}_1$ be the disjoint union of all $\mathfrak{A}_{\neg\psi}$. By preservation of φ under disjoint unions, $\mathfrak{A}_1$ satisfies φ. By reflection of φ for disjoint unions, the disjoint union $\mathfrak{A}$ of $\mathfrak{A}_0$ and $\mathfrak{A}_1$ does not satisfy φ. Thus $\mathfrak{A}_1$ satisfies φ and $\mathfrak{A}$ does not satisfy φ but by construction $\mathfrak{A}$ and $\mathfrak{A}_1$ satisfy the same sentences in uGF. This is impossible since φ is equivalent to a Boolean combination of uGF sentences. □

The following example shows that some very simple Boolean combinations of uGF sentences are not invariant under disjoint unions.

Example 1 *Let*

$$\begin{aligned}\mathcal{O}_{\textit{UCQ/CQ}} &= \{(\forall x(A(x) \vee B(x)) \vee \exists x E(x)\}\\ \mathcal{O}_{\textit{Mat/PTime}} &= \{\forall x A(x) \vee \forall x B(x)\}\end{aligned}$$

Then $\mathcal{O}_{\textit{Mat/PTime}}$ is not preserved under disjoint unions since $\mathfrak{D}_1 = \{A(a)\}$ and $\mathfrak{D}_2 = \{B(b)\}$ are models of $\mathcal{O}_{\textit{Mat/PTime}}$ but $\mathfrak{D}_1 \cup \mathfrak{D}_2$ refutes $\mathcal{O}_{\textit{Mat/PTime}}$; $\mathcal{O}_{\textit{UCQ/CQ}}$ does not reflect disjoint unions since the disjoint union of $\mathfrak{D}'_1 = \{E(a)\}$ and $\mathfrak{D}'_2 = \{F(b)\}$ is a model of $\mathcal{O}_{\textit{UCQ/CQ}}$ but $\mathfrak{D}'_2$ refutes $\mathcal{O}_{\textit{UCQ/CQ}}$. We will use these ontologies later to explain why we restrict this study to fragments of GF that are invariant under disjoint unions.

When studying uGF ontologies, we are going to vary several parameters. The *depth* of a formula φ in openGF is the nesting depth of guarded quantifiers in φ. The *depth* of a sentence $\forall\vec{y}(\alpha(\vec{y}) \to \varphi(\vec{y}))$ in uGF is the depth of $\varphi(\vec{y})$, thus the outermost guarded quantifier is not counted. The *depth* of a uGF ontology is the maximum depth of its sentences. We indicate restricted depth in brackets, writing e.g. uGF(2) to denote the set of all uGF sentences of depth at most 2.

Example 2 *The sentence*

$$\forall xy(R(x,y) \to (A(x) \vee \exists z S(y,z)))$$

is in uGF(1) since the openGF formula $A(x) \vee \exists z S(y,z)$ has depth 1.

For every GF sentence φ, one can construct in polynomial time a conservative extension φ' in uGF(1) by converting into Scott normal form [29]. Thus, the satisfiability and CQ-evaluation problems for full GF can be polynomially reduced to the corresponding problem for uGF(1).

We use uGF^- to denote the fragment of uGF where only equality guards are admitted in the outermost universal quantifier applied to an openGF formula. Thus, the sentence in Example 2 (1) is a uGF sentence of depth 1, but not a uGF^- sentence of depth 1. It is, however, equivalent to the following uGF^- sentence of depth 1:

$$\forall x(\exists y((R(y,x) \wedge \neg A(y)) \to \exists z S(x,z)))$$

An example of a uGF sentence of depth 1 that is not equivalent to a uGF^- sentence of depth 1 is given in Example 3 below. Intuitively, uGF sentences of depth 1 can be thought of as uGF^- sentences of 'depth 1.5' because giving up $\cdot^-$ allows an additional level of 'real' quantification (meaning: over guards that are not forced to be equality), but only in a syntactically restricted way.

The two-variable fragment of uGF is denoted with uGF_2. More precisely, in uGF_2 we admit only the two fixed variables x and y and disallow the use of relation symbols of arity exceeding two. We also consider two extensions of uGF_2 with forms of counting. First, $\text{uGF}_2(f)$ denotes the extension of uGF_2 with function symbols, that is, an $\text{uGF}_2(f)$ ontology is a finite set of uGF_2 sentences and of *functionality axioms* $\forall x\forall y_1\forall y_2((R(x,y_1)\wedge R(x,y_2)) \to (y_1 = y_2))$ [29]. Second, we consider the extension uGC_2 of uGF_2 with counting quantifiers. More precisely, the language openGC_2 is defined in the same way as the two-variable fragment of openGF, but in addition admits *guarded counting quantifiers* [48, 32]: if $n \in \mathbb{N}$, $\{z_1,z_2\} = \{x,y\}$, and $\alpha(z_1,z_2) \in \{R(z_1,z_2), R(z_2,z_1)\}$ for some $R \in \Sigma$ and $\varphi(z_1,z_2)$ is in openGC_2, then $\exists^{\geq n} z_1(\alpha(z_1,z_2)\wedge \varphi(z_1,z_2))$ is in openGC_2. The ontology language uGC_2 is then defined in the same way as uGF_2, using openGC_2 instead of openGF_2. The *depth* of formulas in uGC_2 is defined in the expected way, that is, guarded counting quantifiers and guarded quantifiers both contribute to it.

The above restrictions can be freely combined and we use the obvious names to denote such combinations. For example, $\text{uGF}_2^-(1,f)$ denotes the two-variable fragment of uGF with function symbols and where all sentences must have depth 1 and the guard of the outermost quantifier must be equality. Note that uGF admits equality, although in a restricted way (only in non-guard positions, with the possible exception of the guard of the outermost quantifier). We shall also consider fragments of uGF that admit no equality at all except as a guard of the outermost quantifier. To emphasize that the restricted use of equality is allowed, we from now on use the equality symbol in brackets whenever equality is present, as in $\text{uGF}(=)$, $\text{uGF}^-(1,=)$, and $\text{uGC}_2^-(1,=)$. Conversely, uGF, $\text{uGF}^-(1)$, and $\text{uGC}_2^-(1)$ from now on denote the corresponding fragments where equality is only allowed as a guard of the outermost quantifier.

Description logics are a popular family of ontology languages that are related to the guarded fragments of FO introduced above. We briefly review the basic description logic $\mathcal{ALC}$, further details on this and other DLs mentioned in this paper can be found in the appendix and in [4]. DLs generally use relations of arity one and two, only. An *$\mathcal{ALC}$ concept* is formed according to the syntax rule

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

where A ranges over unary relations and R over binary relations. An $\mathcal{ALC}$ *ontology* $\mathcal{O}$ is a finite set of *concept inclusions* $C \sqsubseteq D$, with C and D $\mathcal{ALC}$ concepts. The semantics of $\mathcal{ALC}$ concepts C can be given by translation to openGF formulas $C^*(x)$ with one free variable x and two variables overall. A concept inclusion $C \sqsubseteq D$ then translates to the uGF_2^- sentence $\forall x(C^*(x) \rightarrow D^*(x))$. The *depth* of an $\mathcal{ALC}$ concept is the maximal nesting depth of $\exists R$ and $\forall R$. The *depth* on an $\mathcal{ALC}$ ontology is the maximum depth of concepts that occur in it. Thus, every $\mathcal{ALC}$ ontology of depth n is a uGF_2^- ontology of depth n. When translating into uGF_2 instead of into uGF_2^-, the depth might decrease by one because one can exploit the outermost quantifier (which does not contribute to the depth). A more detailed description of the relationship between DLs and fragments of uGF is given in the appendix.

Example 3 *The $\mathcal{ALC}$ concept inclusion $\exists S.A \sqsubseteq \forall R.\exists S.B$ has depth 2, but is equivalent to the $\text{uGF}_2(1)$ sentence*

$$\forall xy(R(x,y) \rightarrow ((\exists S.A)^*(x) \rightarrow (\exists S.B)^*(y))$$

Note that for any ontology $\mathcal{O}$ in any DL considered in this paper one can construct in a straightforward way in polynomial time a conservative extension $\mathcal{O}^*$ of $\mathcal{O}$ of depth one. In fact, many DL algorithms for satisfiability or query evaluation assume that the ontology is of depth one and normalized.

We also consider the extensions of $\mathcal{ALC}$ with inverse roles R^- (denoted in the name of the DL by the letter $\mathcal{I}$), role inclusions $R \sqsubseteq S$ (denoted by $\mathcal{H}$), qualified number restrictions $(\geq n\ R\ C)$ (denoted by $\mathcal{Q}$), partial functions as defined above (denoted by $\mathcal{F}$), and local functionality expressed by $(\leq 1R)$ (denoted by $\mathcal{F}_\ell$). The depth of ontologies formulated in these DLs is defined in the obvious way. Thus, $\mathcal{ALCHIQ}$ ontologies (which admit all the constructors introduced above) translate into uGC_2^- ontologies, preserving the depth.

For any syntactic object O (such as an ontology or a query), we use $|O|$ to denote the number of symbols needed to write O, counting relation names, variable names, and so on as a single symbol and assuming that numbers in counting quantifiers and DL number restrictions are coded in unary.

2.2 Guarded Tree Decompositions

We introduce guarded tree decompositions and rooted acyclic queries [30]. A set $G \subseteq \text{dom}(\mathfrak{A})$ is *guarded* in the interpretation $\mathfrak{A}$ if G is a singleton or there are $R \in \Sigma$ and $R(a_1, \ldots, a_k) \in \mathfrak{A}$ such that $G = \{a_1, \ldots, a_k\}$. By $S(\mathfrak{A})$, we denote the set of all guarded sets in $\mathfrak{A}$. A tuple $(a_1, \ldots, a_k) \in A^k$ is *guarded* in $\mathfrak{A}$ if $\{a_1, \ldots, a_k\}$ is a subset of some guarded set in $\mathfrak{A}$. A *guarded tree decomposition* of $\mathfrak{A}$ is a triple (T, E, bag) with (T, E) an acyclic undirected graph and bag a function that assigns to every $t \in T$ a set $\text{bag}(t)$ of atoms such that $\mathfrak{A}_{|\text{dom}(\text{bag}(t))} = \text{bag}(t)$ and

1. $\mathfrak{A} = \bigcup_{t \in T} \text{bag}(t)$;
2. $\text{dom}(\text{bag}(t))$ is guarded for every $t \in T$;
3. $\{t \in T \mid a \in \text{dom}(\text{bag}(t))\}$ is connected in (T, E), for every $a \in \text{dom}(\mathfrak{A})$.

We say that $\mathfrak{A}$ is *guarded tree decomposable* if there exists a guarded tree decomposition of $\mathfrak{A}$. We call (T, E, bag) a *connected guarded tree decomposition (cg-tree decomposition)* if, in addition, (T, E) is connected (i.e., a tree) and $\text{dom}(\text{bag}(t)) \cap \text{dom}(\text{bag}(t')) \neq \emptyset$ for all $(t, t') \in E$. In this case, we often assume that (T, E) has a designated root r, which allows us to view (T, E) as a directed tree whenever convenient.

A CQ $q \leftarrow \phi$ is a *rooted acyclic query (rAQ)* if there exists a cg-tree decomposition (T, E, bag) of the instance $\mathfrak{D}_q$ with root r such that $\text{dom}(\text{bag}(r))$ is the set of answer variables of q. Note that, by definition, rAQs are non-Boolean queries.

Example 4 *The CQ*

$$q(x) \leftarrow \phi, \quad \phi = R(x,y) \wedge R(y,z) \wedge R(z,x)$$

is not an rAQ since $\mathfrak{D}_q$ is not guarded tree decomposable. By adding the conjunct $Q(x,y,z)$ to ϕ one obtains an rAQ.

We will frequently use the following construction: let $\mathfrak{D}$ be an instance and $\mathcal{G}$ a set of guarded sets in $\mathfrak{D}$. Assume that $\mathfrak{B}_G$, $G \in \mathcal{G}$, are interpretations such that $\text{dom}(\mathfrak{B}_G) \cap \text{dom}(\mathfrak{D}) = G$ and $\text{dom}(\mathfrak{B}_{G_1}) \cap \text{dom}(\mathfrak{B}_{G_2}) = G_1 \cap G_2$ for any two distinct guarded sets G_1 and G_2 in $\mathcal{G}$. Then the interpretation

$$\mathfrak{B} = \mathfrak{D} \cup \bigcup_{G \in \mathcal{G}} \mathfrak{B}_G$$

is called the *interpretation obtained from $\mathfrak{D}$ by hooking $\mathfrak{B}_G$ to $\mathfrak{D}$ for all $G \in \mathcal{G}$*. If the $\mathfrak{B}_G$ are cg-tree decomposable interpretations with $\text{dom}(\text{bag}(r)) = G$ for the root r of a (fixed) cg-tree decomposition of $\mathfrak{B}_G$, then $\mathfrak{B}$ is called a *forest model of $\mathfrak{D}$ defined using $\mathcal{G}$*. If $\mathcal{G}$ is the set of all maximal guarded sets in $\mathfrak{D}$, then we call $\mathfrak{B}$ simply a *forest model of $\mathfrak{D}$*. The following result can be proved using standard guarded tree unfolding [29, 30].

Lemma 1 *Let $\mathcal{O}$ be a uGF(=) or uGC$_2$(=) ontology, $\mathfrak{D}$ a possibly infinite instance, and $\mathfrak{A}$ a model of $\mathfrak{D}$ and $\mathcal{O}$. Then there exists a forest model $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}$ and a homomorphism h from $\mathfrak{B}$ to $\mathfrak{A}$ that preserves dom($\mathfrak{D}$).*

3. MATERIALIZABILITY

We introduce and study materializability of ontologies as a necessary condition for query evaluation to be in PTIME. In brief, an ontology $\mathcal{O}$ is materializable if for every instance $\mathfrak{D}$, there is a model $\mathfrak{A}$ of $\mathcal{O}$ and $\mathfrak{D}$ such that for all queries, the answers on $\mathfrak{A}$ agree with the certain answers on $\mathfrak{D}$ given $\mathcal{O}$. We show that this sometimes, but not always, coincides with existence of universal models defined in terms of homomorphisms. We then prove that in uGF(=) and uGC$_2$(=), non-materializability implies CONP-hard query answering while this is not the case for GF. Using these results, we further establish that in uGF(=) and uGC$_2$(=), query evalution w.r.t. ontologies to be in PTIME, Datalog$^{\neq}$-rewritable, and CONP-hard does not depend on the query language, that is, all these properties agree for rAQs, CQs, and UCQs. Again, this is not the case for GF.

Definition 2 (Materializability) *Let $\mathcal{O}$ be an FO(=)-ontology, $\mathcal{Q}$ a class of queries, and $\mathcal{M}$ a class of instances. Then*

- *an interpretation $\mathfrak{B}$ is a* $\mathcal{Q}$-materialization *of $\mathcal{O}$ and an instance $\mathfrak{D}$ if it is a model of $\mathcal{O}$ and $\mathfrak{D}$ and for all $q(\vec{x}) \in \mathcal{Q}$ and $\vec{a}$ in dom($\mathfrak{D}$), $\mathfrak{B} \models q(\vec{a})$ iff $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$.*
- *$\mathcal{O}$ is* $\mathcal{Q}$-materializable for $\mathcal{M}$ *if for every instance $\mathfrak{D} \in \mathcal{M}$ that is consistent w.r.t. $\mathcal{O}$, there is a $\mathcal{Q}$-materialization of $\mathcal{O}$ and $\mathfrak{D}$.*

If $\mathcal{M}$ is the class of all instances, we simply speak of $\mathcal{Q}$-materializability of $\mathcal{O}$.

We first observe that the materializability of ontologies does not depend on the query language (although concrete materializations do).

Theorem 2 *Let $\mathcal{O}$ be a uGF(=) or uGC$_2$(=) ontology and $\mathcal{M}$ a class of instances. Then the following conditions are equivalent:*

1. $\mathcal{O}$ is rAQ-materializable for $\mathcal{M}$;

2. *$\mathcal{O}$ is CQ-materializable for $\mathcal{M}$;*

3. *$\mathcal{O}$ is UCQ-materializable for $\mathcal{M}$.*

PROOF. The only non-trivial implication is (1) $\Rightarrow$ (2). It can be proved by using Lemma 1 and showing that if $\mathfrak{A}$ is a rAQ-materialization of an ontology $\mathcal{O}$ and an instance $\mathfrak{D}$, then any forest model $\mathfrak{B}$ of $\mathcal{O}$ and $\mathfrak{D}$ which admits a homomorphism to $\mathfrak{A}$ that preserves $\mathsf{dom}(\mathfrak{D})$ is a CQ-materialization of $\mathcal{O}$ and $\mathfrak{D}$. □

Because of Theorem 2, we from now on speak of *materializability* without reference to a query language and of *materializations* instead of UCQ-materializations (which are then also CQ-materializations and rAQ-materializations).

A notion closely related to materializations are (homomorphically) universal models as used e.g. in data exchange [22, 20]. A model of an ontology $\mathcal{O}$ and an instance $\mathfrak{D}$ is *hom-universal* if there is a homomorphism preserving $\mathrm{dom}(\mathfrak{D})$ into any model of $\mathcal{O}$ and $\mathfrak{D}$. We say that an ontology $\mathcal{O}$ *admits hom-universal models* if there is a hom-universal model for $\mathcal{O}$ and any instance $\mathfrak{D}$. It is well-known that hom-universal models are closely related to what we call UCQ-materializations. In fact, in many DLs and in $\mathrm{uGC}_2(=)$, materializability of an ontology $\mathcal{O}$ coincides with $\mathcal{O}$ admitting hom-universal models (although for concrete models, being hom-universal is not the same as being a materialization). We show in the long version that this is not the case for ontologies in $\mathrm{uGF}(2)$ (with three variables). The proof also shows that admitting hom-universal models is not a necessary condition for query evaluation to be in PTIME (in contrast to materializability).

Lemma 2 *A $\mathrm{uGC}_2(=)$ ontology is materializable iff it admits hom-universal models. This does not hold for $\mathrm{uGF}(2)$ ontologies.*

The following theorem links materializability to computational complexity, thus providing the main reason for our interest into this notion. The proof is by reduction of 2+2-SAT [50], a variation of a related proof from [42].

Theorem 3 *Let $\mathcal{O}$ be an FO(=)-ontology that is invariant under disjoint unions. If $\mathcal{O}$ is not materializable, then rAQ-evaluation w.r.t. $\mathcal{O}$ is* CONP*-hard.*

We remark that, in the proof of Theorem 3, we use instances and rAQs that use additional fresh (binary) relation symbols, that is, relation symbols that do not occur in $\mathcal{O}$.

The ontology $\mathcal{O}_{\text{Mat/PTime}}$ from Example 1 shows that Theorem 3 does not hold for GF ontologies, even if they are of depth 1 and use only a single variable. In fact, $\mathcal{O}_{\text{Mat/PTime}}$ is not CQ-materializable, but CQ-evaluation is in PTIME (which is both easy to see).

Theorem 4 *For all $\mathrm{uGF}(=)$ and $\mathrm{uGC}_2(=)$ ontologies $\mathcal{O}$, the following are equivalent:*

1. *rAQ-evaluation w.r.t. $\mathcal{O}$ is in* PTIME*;*

2. *CQ-evaluation w.r.t. $\mathcal{O}$ is in* PTIME*;*

3. *UCQ-evaluation w.r.t. $\mathcal{O}$ is in* PTIME*.*

This remains true when 'in PTIME*' is replaced with 'Datalog$^{\neq}$-rewritable' and with '*CONP*-hard' (and with 'Datalog-rewritable' if $\mathcal{O}$ is a uGF ontology).*

PROOF. By Theorem 3, we can concentrate on ontologies that are materializable. For the non-trivial implication of Point 3 by Point 1, we exploit materializability to rewrite UCQs into a finite disjunction of queries $q \wedge \bigwedge_i q_i$ where q is a "core CQ" that only needs to be evaluated over the input instance $\mathfrak{D}$ (ignoring labeled nulls) and each q_i is a rAQ. This is similar to squid decompositions in [12], but more subtle due to the presence of subqueries that are not connected to any answer variable of q. Similar constructions are used also to deal with Datalog$^{\neq}$-rewritability and with CONP-hardness. □

The ontology $\mathcal{O}_{\text{UCQ/CQ}}$ from Example 1 shows that Theorem 4 does not hold for GF ontologies, even if they use only a single variable and are of depth 1 up to an outermost universal quantifier with an equality guard.

Lemma 3 *CQ-evaluation w.r.t. $\mathcal{O}_{\text{UCQ/CQ}}$ is in* PTIME *and UCQ-evaluation w.r.t. $\mathcal{O}_{\text{UCQ/CQ}}$ is* CONP*-hard.*

The lower bound essentially follows the construction in the proof of Theorem 3 and the upper bound is based on a case analysis, depending on which relations occur in the CQ and in the input instance.

4. UNRAVELLING TOLERANCE

While materializability of an ontology is a necessary condition for PTIME query evaluation in $\mathrm{uGF}(=)$ and $\mathrm{uGC}_2(=)$, we now identify a sufficient condition called unravelling tolerance that is based on unravelling instances into cg-tree decomposable instances (which might be infinite). In fact, unravelling tolerance is even a sufficient condition of Datalog$^{\neq}$-rewritability and we will later establish our dichotomy results by showing that, for the ontology languages in question, materializability implies unravelling tolerance.

We start with introducing suitable forms of unravelling (also called guarded tree unfolding, see [30] and references therein). The *uGF-unravelling* $\mathfrak{D}^u$ of an instance $\mathfrak{D}$ is constructed as follows. Let $T(\mathfrak{D})$ be the set of all sequences $t = G_0 G_1 \cdots G_n$ where G_i, $0 \leq i \leq n$, are maximal guarded sets of $\mathfrak{D}$ and

(a) $G_i \neq G_{i+1}$,

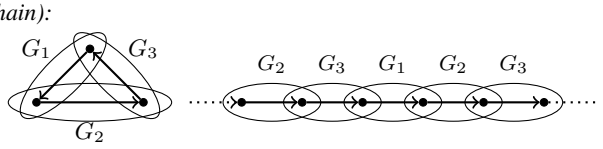
(b) $G_i \cap G_{i+1} \neq \emptyset$, and

(c) $G_{i-1} \neq G_{i+1}$.

In the following, we associate each $t \in T(\mathfrak{D})$ with a set of atoms $\mathsf{bag}(t)$. Then we define $\mathfrak{D}^u$ as $\bigcup_{t \in T(\mathfrak{D})} \mathrm{bag}(t)$ and note that $(T(\mathfrak{D}), E, \mathrm{bag})$ is a cg-tree decomposition of $\mathfrak{D}^u$ where $(t, t') \in E$ if $t' = tG$ for some G.

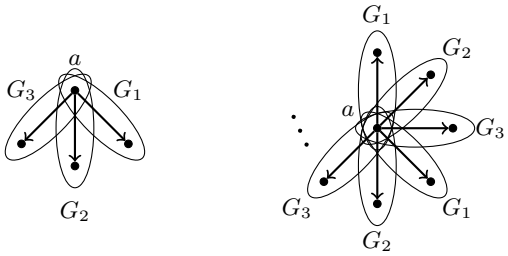
Set $\mathsf{tail}(G_0 \cdots G_n) = G_n$. Take an infinite supply of *copies* of any $d \in \mathrm{dom}(\mathfrak{D})$. We set $e^{\uparrow} = d$ if e is a copy of d. We define $\mathrm{bag}(t)$ (up to isomorphism) proceeding by induction on the length of the sequence t. For any $t = G$, $\mathrm{bag}(t)$ is an instance whose domain is a set of copies of $d \in G$ such that the mapping $e \mapsto e^{\uparrow}$ is an isomorphism from $\mathrm{bag}(G)$ onto the subinstance $\mathfrak{D}_{|G}$ of $\mathfrak{D}$ induced by G. To define $\mathrm{bag}(t')$ for $t' = tG'$ when $\mathsf{tail}(t) = G$, take for any $d \in G' \setminus G$ a fresh copy d' of d and define $\mathrm{bag}(t')$ with domain $\{d' \mid d \in G' \setminus G\} \cup \{e \in \mathrm{bag}(t) \mid e^{\uparrow} \in G' \cap G)\}$ such that the mapping $e \mapsto e^{\uparrow}$ is an isomorphism from $\mathrm{bag}(t')$ onto $\mathfrak{D}_{|G'}$. The following example illustrates the construction of $\mathfrak{D}^u$.

Example 5 *(1) Consider the instance $\mathfrak{D}$ depicted below with the maximal guarded sets G_1, G_2, G_3. Then the unravelling $\mathfrak{D}^u$ of $\mathfrak{D}$ consists of three isomorphic chains (we depict only one such chain):*



(2) Next consider the instance $\mathfrak{D}$ depicted below which has the shape of a tree of depth one with root a and has three maximal

guarded sets G_1, G_2, G_3. Then the unravelling $\mathfrak{D}^u$ of $\mathfrak{D}$ consists of three isomorphic trees of depth one of infinite outdegree (again we depict only one):



By construction, the mapping $h : e \mapsto e^{\uparrow}$ is a homomorphism from $\mathfrak{D}^u$ onto $\mathfrak{D}$ and the restriction of h to any guarded set G is an isomorphism. It follows that for any uGF(=) ontology $\mathcal{O}$, UCQ $q(\vec{x})$, and $\vec{a}$ in $\mathfrak{D}^u$, if $\mathcal{O}, \mathfrak{D}^u \models q(\vec{a})$, then $\mathcal{O}, \mathfrak{D} \models q(h(\vec{a}))$. This implication does not hold for ontologies in the guarded fragment with functions or counting. To see this, let

$$\mathcal{O} = \{\forall x(\exists^{\geq 4} y R(x,y) \rightarrow A(x))\}$$

Then $\mathcal{O}, \mathfrak{D}^u \models A(a)$ for the instance $\mathfrak{D}$ from Example 5 (2) but $\mathcal{O}, \mathfrak{D} \not\models A(a)$. For this reason the uGF-unravelling is not appropriate for the guarded fragment with functions or counting. By replacing Condition (c) by the stronger condition

(c′) $G_i \cap G_{i-1} \neq G_i \cap G_{i+1}$,

we obtain an unravelling that we call *uGC$_2$-unravelling* and that we apply whenever all relations have arity at most two. One can show that the uGC$_2$-unravelling of an instance preserves the number of R-successors of constants in $\mathfrak{D}$ and that, in fact, the implication '$\mathcal{O}, \mathfrak{D}^u \models q(\vec{a}) \Rightarrow \mathcal{O}, \mathfrak{D} \models q(h(\vec{a}))$' holds for every uGC$_2$(=) ontology $\mathcal{O}$, UCQ $q(\vec{x})$, and tuple $\vec{a}$ in the uGC$_2$-unravelling $\mathfrak{D}^u$ of $\mathfrak{D}$.

We are now ready to define unravelling tolerance. For a maximal guarded set G in $\mathfrak{D}$, the *copy in bag(G) of a tuple $\vec{a} = (a_1, \ldots, a_k)$ in G* is the unique $\vec{b} = (b_1, \ldots, b_k)$ in dom(bag(G)) such that $b_i^{\uparrow} = a_i$ for $1 \leq i \leq k$.

Definition 3 *A uGF(=) (resp. uGC$_2$(=)) ontology $\mathcal{O}$ is* unravelling tolerant *if for every instance $\mathfrak{D}$, every rAQ $q(\vec{x})$, and every tuple $\vec{a}$ in $\mathfrak{D}$ such that the set G of elements of $\vec{a}$ is maximally guarded in $\mathfrak{D}$ the following are equivalent:*

1. $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$;
2. $\mathcal{O}, \mathfrak{D}^u \models q(\vec{b})$ *where $\vec{b}$ is the copy of $\vec{a}$ in bag(G)*

where $\mathfrak{D}^u$ is the uGF-unravelling (resp. the uGC$_2$-unravelling) of $\mathfrak{D}$.

We have seen above that the implication (2) $\Rightarrow$ (1) in Definition 3 holds for every uGF(=) and uGC$_2$(=) ontology and every UCQ. Note that it is pointless to define unravelling tolerance using the implication (1) $\Rightarrow$ (2) for UCQs or CQs that are not acyclic. The following example shows that (1) $\Rightarrow$ (2) does not always hold for rAQs.

Example 6 *Consider the uGF ontology $\mathcal{O}$ that contains the sentences*

$$\forall x\big(X(x) \rightarrow (\exists y(R(x,y) \wedge X(y)) \rightarrow E(x))\big)$$

with $X \in \{A, \neg A\}$ and

$$\forall x\big(E(x) \rightarrow ((R(x,y) \vee R(y,x)) \rightarrow E(y))\big)$$

For instances $\mathfrak{D}$ not using A, $\mathcal{O}$ states that $E(a)$ is entailed for all $a \in$ dom$(\mathfrak{D})$ that are R-connected to some R-cycle in $\mathfrak{D}$ with an odd number of constants. Thus, for the instance $\mathfrak{D}$ from Example 5 (1) we have $\mathcal{O}, \mathfrak{D} \models E(a)$ for every $a \in$ dom$(\mathfrak{D})$ but $\mathcal{O}, \mathfrak{D}^u \not\models E(a)$ for any $a \in$ dom$(\mathfrak{D}^u)$.

We now show that, as announced, unraveling tolerance implies that query evaluation is Datalog$^{\neq}$-rewritable.

Theorem 5 *For all uGF(=) and uGC$_2$(=) ontologies $\mathcal{O}$, unravelling tolerance of $\mathcal{O}$ implies that rAQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable (and Datalog-rewritable if $\mathcal{O}$ is formulated in uGF).*

PROOF. We sketch the proof for the case that $\mathcal{O}$ is a uGF(=) ontology; similar constructions work for the other cases. Suppose that $\mathcal{O}$ is unravelling tolerant, and that $q(\vec{x})$ is a rAQ. We construct a Datalog$^{\neq}$ program Π that, given an instance $\mathfrak{D}$, computes the certain answers $\vec{a}$ of q on $\mathfrak{D}$ given $\mathcal{O}$, where w.l.o.g. we can restrict our attention to answers $\vec{a}$ such that the set G of elements of $\vec{a}$ is maximally guarded in $\mathfrak{D}$. By unravelling tolerance, it is enough to check if $\mathcal{O}, \mathfrak{D}^u \models q(\vec{b})$, where $\vec{b}$ is the copy of $\vec{a}$ in bag(G) and $\mathfrak{D}^u$ is the uGF-unravelling of $\mathfrak{D}$.

The Datalog$^{\neq}$ program Π assigns to each maximally guarded tuple $\vec{a} = (a_1, \ldots, a_k)$ in $\mathfrak{D}$ a set of *types*. Here, a type is a maximally consistent set of uGF formulas with free variables in $x_1, \ldots, x_k$, where the variable x_i represents the element a_i. It can be shown that we only need to consider types with formulas of the form ϕ or $\neg\phi$, where ϕ is obtained from a subformula of $\mathcal{O}$ or q by substituting a variable in $x_1, \ldots, x_k$ for each of its free variables, or ϕ is an atomic formula in the signature of $\mathcal{O}, q$ with free variables in $x_1, \ldots, x_k$. In particular, the set of all types is finite. We further restrict our attention to types θ that are realizable in some model of $\mathcal{O}$, i.e., there is a model $\mathfrak{B}(\theta)$ of $\mathcal{O}$ containing all elements of $\vec{a}$ that is a model of each formula in θ under the interpretation $x_i \mapsto a_i$. The Datalog$^{\neq}$ program Π ensures the following:

1. for any two maximally guarded tuples $\vec{a} = (a_1, \ldots, a_k)$, $\vec{b} = (b_1, \ldots, b_l)$ in $\mathfrak{D}$ that share an element, and any type θ assigned to $\vec{a}$ there is a type θ' assigned to $\vec{b}$ that is *compatible* to θ (intuitively, the two types agree on all formulas that only talk about elements shared by $\vec{a}$ and $\vec{b}$);
2. a tuple $\vec{a} = (a_1, \ldots, a_k)$ is an answer to Π if all types assigned to $\vec{a}$ contain $q(x_1, \ldots, x_k)$, or some maximally guarded tuple $\vec{b}$ in $\mathfrak{D}$ has no type assigned to it.

It can be shown that $\vec{a}$ is an answer to Π iff $\mathcal{O}, \mathfrak{D}^u \models q(\vec{a})$.

The interesting part is the "if" part. Suppose $\vec{a} = (a_1, \ldots, a_k)$ is *not* an answer to Π. Then, each maximally guarded tuple $\vec{b}$ in $\mathfrak{D}$ is assigned to at least one type, and for some type θ^* assigned to $\vec{a}$ we have $q(x_1, \ldots, x_k) \notin \theta^*$. We use this type assignment to label each maximally guarded tuple $\vec{b}$ of $\mathfrak{D}^u$ with a type $\theta_{\vec{b}}$ so that (1) for each maximally guarded tuple $\vec{c}$ of $\mathfrak{D}^u$ that shares an element with $\vec{b}$ the two types $\theta_{\vec{b}}$ and $\theta_{\vec{c}}$ are compatible; and (2) $\theta^* = \theta_{\vec{a}^*}$, where $\vec{a}^*$ is the copy of $\vec{a}$ in $G = \{a_1, \ldots, a_k\}$. We can now show that the interpretation $\mathfrak{A}$ obtained from $\mathfrak{D}^u$ by hooking $\mathfrak{B}(\theta_{\vec{b}})$ to $\mathfrak{D}^u$, for all maximally guarded tuples $\vec{b}$ of $\mathfrak{D}^u$, is a model of $\mathcal{O}$ and $\mathfrak{D}^u$ with $\mathfrak{A} \not\models q(\vec{a}^*)$. □

5. DICHOTOMIES

We prove dichotomies between PTIME and CONP for query evaluation in the five ontology languages displayed in the bottom-most part of Figure 1. In fact, the dichotomy is even between

Datalog$^{\neq}$-rewritability and coNP. The proof establishes that for ontologies $\mathcal{O}$ formulated in any of these languages, CQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable iff it is in PTIME iff $\mathcal{O}$ is unravelling tolerant iff $\mathcal{O}$ is materializable for the class of (possibly infinite) cg-tree decomposable instances iff $\mathcal{O}$ is materializable and that, if none of this is the case, CQ-evaluation w.r.t. $\mathcal{O}$ is coNP-hard. The main step towards the dichotomy result is provided by the following theorem.

Theorem 6 *Let $\mathcal{O}$ be an ontology formulated in one of uGF(1), uGF$^-$(1,=), uGF$_2^-$(2), uGC$_2^-$(1,=), or an $\mathcal{ALCHIF}$ ontology of depth 2. If $\mathcal{O}$ is materializable for the class of (possibly infinite) cg-tree decomposable instances $\mathfrak{D}$ with sig($\mathfrak{D}$) $\subseteq$ sig($\mathcal{O}$), then $\mathcal{O}$ is unravelling tolerant.*

PROOF. We sketch the proof for uGF(1) and uGF$_2^-$(2) ontologies $\mathcal{O}$ and then discuss the remaining cases. Assume that $\mathcal{O}$ satisfies the precondition from Theorem 6. Let $\mathfrak{D}$ be an instance and $\mathfrak{D}^u$ its uGF unravelling. Let $\vec{a}$ be a tuple in a maximal guarded set G in D and $\vec{b}$ be the copy in dom(bag(G)) of $\vec{a}$. Further let q be an rAQ such that $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$. We have to show that $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$. Using the condition that $\mathcal{O}$ is materializable for the class of cg-tree decomposable instances $\mathfrak{D}$ with sig($\mathfrak{D}$) $\subseteq$ sig($\mathcal{O}$), it can be shown that there exists a materialization $\mathfrak{B}$ of $\mathcal{O}$ and $\mathfrak{D}^u$.

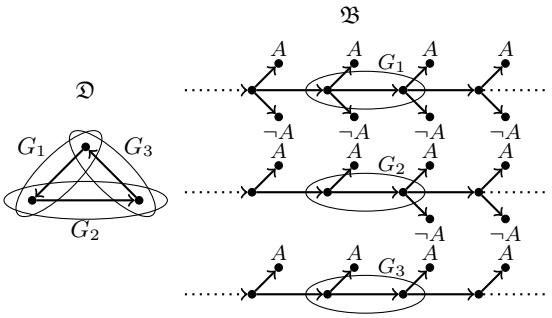
By Lemma 1 we may assume that $\mathfrak{B}$ is a forest model which is obtained from $\mathfrak{D}^u$ by hooking cg-tree decomposable $\mathfrak{B}_{\text{bag}(t)}$ to maximal guarded bag(t) in $\mathfrak{D}^u$. Now we would like to obtain a model of $\mathcal{O}$ and the original $\mathfrak{D}$ by hooking for any maximal guarded G in $\mathfrak{D}$ the interpretation $\mathfrak{B}_{\text{bag}(G)}$ to $\mathfrak{D}$ rather than to $\mathfrak{D}^u$. However, the resulting model is then not guaranteed to be a model of $\mathcal{O}$. The following example illustrates this. Let $\mathcal{O}$ contain

$$\forall x \exists y (S(x,y) \wedge A(y)),$$

and for $\varphi(x) = \exists z (S(x,z) \wedge \neg A(z))$

$$\forall xy (R(x,y) \to (\varphi(x) \to \varphi(y))$$

Thus in every model of $\mathcal{O}$ each node has an S-successor in A and having an S-successor that is not in A is propagated along R. $\mathcal{O}$ is unravelling tolerant. Consider the instance $\mathfrak{D}$ from Example 5 (1) depicted here again with the maximal guarded sets G_1, G_2, G_3.



We have seen that the unravelling $\mathfrak{D}^u$ of $\mathfrak{D}$ consists of three chains. An example of a forest model $\mathfrak{B}$ of $\mathcal{O}$ and $\mathfrak{D}^u$ is given in the figure. Even in this simple example a naive way of hooking the models $\mathfrak{B}_{G_i}$, $i = 1,2,3$, to the original instance $\mathfrak{D}$ will lead to an interpretation not satisfying $\mathcal{O}$ as the propagation condition for S-successors not in A will not be satisfied. To ensure that we obtain a model of $\mathcal{O}$ we first define a new instance $\mathfrak{D}^{u+} \supseteq \mathfrak{D}^u$ by adding to each maximal guarded set in $\mathfrak{D}^u$ a copy of any entailed rAQ. The following facts are needed for this to work:

1. Automorphisms: for any $t, t' \in T(\mathfrak{D})$ with tail(t) = tail(t') there is an automorphism $\hat{h}_{t,t'}$ of $\mathfrak{D}^u$ mapping bag(t) onto bag(t') and such that $\hat{h}_{t,t'}(a)^\uparrow = a^\uparrow$ for all $a \in \text{dom}(\mathfrak{D}^u)$. (This is trivial in the example above.) It is for this property that we need that $\mathfrak{D}^u$ is obtained from $\mathfrak{D}$ using maximal guarded sets only and the assumption that $G_{i-1} \neq G_{i+1}$. It follows that if tail(t) = tail(t') then the same rAQs are entailed at bag(t) and bag(t') in $\mathfrak{D}^u$.

2. Homomorphism preservation: if there is a homomorphism h from instance $\mathfrak{D}$ to instance $\mathfrak{D}'$ then $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ entails $\mathcal{O}, \mathfrak{D}' \models q(h(\vec{a}))$. Ontologies in uGF(1) and uGF$_2^-$(2) have this property as they do not use equality nor counting. Because of homomorphism preservation the answers in $\mathfrak{D}^u$ to rAQs are invariant under moving from $\mathfrak{D}^u$ to $\mathfrak{D}^{u+}$. Note that the remaining ontology languages in Theorem 6 do not have this property.

Now using that $\mathfrak{D}^{u+}$ is materializable w.r.t. $\mathcal{O}$ one can uniformize a materialization $\mathfrak{B}^{u+}$ of $\mathfrak{D}^{u+}$ that is a forest model in such a way that the automorphisms $\hat{h}_{t,t'}$ for tail(t) = tail(t') extend to automorphisms of the resulting model $\mathfrak{B}^{u*}$ which also still satisfies $\mathcal{O}$. In the example, after uniformization all chains will behave in the same way in the sense that every node receives an S-successor not in A. We then obtain a forest model $\mathfrak{B}^*$ of $\mathfrak{D}$ by hooking the interpretations $\mathfrak{B}^{u*}_{\text{bag}(G)}$ to the maximal guarded sets G in $\mathfrak{D}$. $(\mathfrak{B}^*, \vec{a})$ and $(\mathfrak{B}^{u*}, \vec{b})$ are guarded bisimilar. Thus $\mathfrak{B}^*$ is a model of $\mathcal{O}$ and $\mathfrak{B}^* \not\models q(\vec{a})$, as required.

For uGF$^-$(1,=) and uGC$_2^-$(1,=) the intermediate step of constructing $\mathfrak{D}^{u+}$ is not required as sentences have smaller depth and no uniformization is needed to satisfy the ontology in the new model. For $\mathcal{ALCHIF}$ ontologies of depth 2 uniformization by constructing $\mathfrak{D}^{u+}$ is needed and has to be done carefully to preserve functionality when adding copies of entailed rAQs to $\mathfrak{D}^u$. □

We can now prove our main dichotomy result.

Theorem 7 *Let $\mathcal{O}$ be an ontology formulated in one of uGF(1), uGF$^-$(1,=), uGF$_2^-$(2), uGC$_2^-$(1,=), or an $\mathcal{ALCHIF}$ ontology of depth 2. Then the following conditions are equivalent (unless* PTIME = NP*):*

1. *$\mathcal{O}$ is materializable;*
2. *$\mathcal{O}$ is materializable for the class of cg-tree decomposable instances $\mathfrak{D}$ with sig($\mathfrak{D}$) $\subseteq$ sig($\mathcal{O}$);*
3. *$\mathcal{O}$ is unravelling tolerant;*
4. *query evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable (and Datalog-rewritable if $\mathcal{O}$ is formulated in uGF);*
5. *query evaluation w.r.t. $\mathcal{O}$ is in* PTIME.

Otherwise, query evaluation w.r.t. $\mathcal{O}$ is coNP*-hard.*

PROOF. (1) ⇒ (2) is not difficult to establish by a compactness argument. (2) ⇒ (3) is Theorem 6. (3) ⇒ (4) is Theorem 5. (4) ⇒ (5) is folklore. (5) ⇒ (1) is Theorem 3 (assuming PTIME ≠ NP). □

The qualification 'with sig($\mathfrak{D}$) $\subseteq$ sig($\mathcal{O}$)' in Point 2 of Theorem 7 can be dropped without compromising the correctness of the theorem, and the same is true for Theorem 6. It will be useful, though, in the decision procedures developed in Section 8.

6. CSP-HARDNESS

We establish the four CSP-hardness results displayed in the middle part of Figure 1, starting with a formal definition of CSP-hardness. In addition, we derive from the existence of CSPs in PTIME that are not Datalog definable the existence of ontologies in any of these languages with PTIME query evaluation that are not Datalog$^{\neq}$ rewritability.

Let $\mathfrak{A}$ be an instance. The *constraint satisfaction problem* CSP($\mathfrak{A}$) is to decide, given an instance $\mathfrak{D}$, whether there is a homomorphism from $\mathfrak{D}$ to $\mathfrak{A}$, which we denote with $\mathfrak{D} \to \mathfrak{A}$. In this context, $\mathfrak{A}$ is called the *template* of CSP($\mathfrak{A}$). We will generally and w.l.o.g. assume that relations in sig($\mathfrak{A}$) have arity at most two and that the template $\mathfrak{A}$ *admits precoloring*, that is, for each $a \in \mathsf{dom}(\mathfrak{A})$, there is a unary relation symbol P_a such that $P_a(b) \in \mathfrak{A}$ iff $b = a$ [19]. It is known that for every template $\mathfrak{A}$, there is a template $\mathfrak{A}'$ of this form such that CSP($\mathfrak{A}$) is polynomially equivalent to CSP($\mathfrak{A}'$) [39]. We use coCSP($\mathfrak{A}$) to denote the complement of CSP($\mathfrak{A}$).

Definition 4 *Let $\mathcal{L}$ be an ontology language and $\mathcal{Q}$ a class of queries. Then $\mathcal{Q}$-evaluation w.r.t. $\mathcal{L}$ is* CSP-hard *if for every template $\mathfrak{A}$, there exists an $\mathcal{L}$ ontology $\mathcal{O}$ such that*

1. *there is a $q \in \mathcal{Q}$ such that coCSP($\mathfrak{A}$) polynomially reduces to evaluating the OMQ $(\mathcal{O}, q)$ and*
2. *for every $q \in \mathcal{Q}$, evaluating the OMQ $(\mathcal{O}, q)$ is polynomially reducible to coCSP($\mathfrak{A}$).*

It can be verified that a dichotomy between PTIME and CONP for $\mathcal{Q}$-evaluation w.r.t. $\mathcal{L}$ ontologies implies a dichotomy between PTIME and NP for CSPs, a notorious open problem known as the Feder-Vardi conjecture [23, 7], when $\mathcal{Q}$-evaluation w.r.t. $\mathcal{L}$ is CSP-hard. As noted in the introduction, a tentative proof of the conjecture has recently been announced, but at the time this article is published, its status still remains unclear.

The following theorem summarizes our results on CSP-hardness. We formulate it for CQs, but remark that due to Theorem 4, a dichotomy between PTIME and CONP for any of the mentioned ontology languages and any query language from the set {rAQ, CQ, UCQ} implies the Feder-Vardi conjecture.

Theorem 8 *For any of the following ontology languages, CQ-evaluation w.r.t. $\mathcal{L}$ is CSP-hard: uGF$_2(1,=)$, uGF$_2(2)$, uGF$_2(1,f)$, and the class of $\mathcal{ALCF}_\ell$ ontologies of depth 2.*

PROOF. We sketch the proof for uGF$_2(1,=)$ and then indicate the modifications needed for uGF$_2(1,f)$ and $\mathcal{ALCF}_\ell$ ontologies of depth 2. For uGF$_2(2)$, the result follows from a corresponding result in [42] for $\mathcal{ALC}$ ontologies of depth 3.

Let $\mathfrak{A}$ be a template and assume w.l.o.g. that $\mathfrak{A}$ admits precoloring. Let R_a be a binary relation for each $a \in \mathsf{dom}(\mathfrak{A})$, and set

$$\begin{array}{rcl} \varphi_a^{\neq}(x) & = & \exists y(R_a(x,y) \wedge \neg(x=y)) \\ \varphi_a^{=}(x) & = & \exists y(R_a(x,y) \wedge (x=y)) \end{array}$$

Then $\mathcal{O}$ contains

$$\begin{array}{ll} \forall x(\bigwedge_{a \neq a'} \neg(\varphi_a^{\neq}(x) \wedge \varphi_{a'}^{\neq}(x)) \wedge \bigvee_a \varphi_a^{\neq}(x)) & \\ \forall x(A(x) \to \neg\varphi_a^{\neq}(x)) & \text{when } A(a) \notin \mathfrak{A} \\ \forall xy(R(x,y) \to \neg(\varphi_a^{\neq}(x) \wedge \varphi_{a'}^{\neq}(y))) & \text{when } R(a,a') \notin \mathfrak{A} \\ \forall x \varphi_a^{=}(x) & \text{for all } a \in \mathsf{dom}(\mathfrak{A}) \end{array}$$

where A and R range over symbols in sig($\mathfrak{A}$) of the respective arity. A formula $\varphi_a^{\neq}(x)$ being true at a constant c in an instance $\mathfrak{D}$ means that c is mapped to $a \in \mathsf{dom}(\mathfrak{A})$ by a homomorphism from $\mathfrak{D}$ to $\mathfrak{A}$. The first sentence in $\mathcal{O}$ thus ensures that every node in $\mathfrak{D}$ is mapped to exactly one node in $\mathfrak{A}$ and the second and third set of sentences ensure that we indeed obtain a homomorphism. The last set of sentences enforces that $\varphi_a^{=}(x)$ is true at every constant c. This makes the disjunction in the first sentence 'invisible' to the query (in which inequality is not available), thus avoiding that $\mathcal{O}$ is CONP-hard for trivial reasons. In the long version, we show that $\mathcal{O}$ satisfies Conditions 1 and 2 from Definition 4 where there query q used in Condition 1 is $q \leftarrow N(x)$ with N a fresh unary relation.

For uGF$_2(1,f)$, state that a binary relation F is a function and that $\forall x F(x,x)$. Now replace in $\mathcal{O}$ the formulas $\varphi_a^{\neq}(x)$ by $\exists y(R_a(x,y) \wedge \neg F(x,y))$ and $\varphi_a^{=}(x)$ by $\exists y(R_a(x,y) \wedge F(x,y))$.

For $\mathcal{ALCF}_\ell$ of depth 2, replace in $\mathcal{O}$ the formulas $\varphi_a^{\neq}(x)$ by $\exists^{\geq 2} y R_a(x,y)$ and $\varphi_a^{=}(x)$ by $\exists y R_a(x,y)$. The resulting ontology is equivalent to a $\mathcal{ALCF}_\ell$ ontology of depth 2. □

It is known that for some templates $\mathfrak{A}$, CSP($\mathfrak{A}$) is in PTIME while coCSP($\mathfrak{A}$) is not Datalog$^{\neq}$-definable [23]. Then CQ-evaluation w.r.t. the ontologies $\mathcal{O}$ constructed from $\mathfrak{A}$ in the proof of Theorem 4 is in PTIME, but not Datalog$^{\neq}$-rewritable.

Theorem 9 *In any of the following ontology languages $\mathcal{L}$ there exist ontologies with* PTIME *CQ-evaluation which are not Datalog$^{\neq}$-rewritable: uGF$_2(1,=)$, uGF$_2(2)$, uGF$_2(1,f)$, and the class of $\mathcal{ALCF}_\ell$ ontologies of depth 2.*

The ontology languages in Theorem 5 thus behave provably different from the languages for which we proved a dichotomy in Section 5, since there PTIME query evaluation and Datalog$^{\neq}$-rewritability coincide.

7. NON-DICHOTOMY AND UNDECIDABILITY

We show that ontology languages that admit sentences of depth 2 as well as functions symbols tend to be computationally problematic as they do neither enjoy a dichotomy between PTIME and CONP nor decidability of meta problems such as whether query evaluation w.r.t. a given ontology $\mathcal{O}$ is in PTIME, Datalog$^{\neq}$-rewritable, or CONP-hard, and whether $\mathcal{O}$ is materializable. We actually start with these undecidability results.

Theorem 10 *For the ontology languages uGF$_2^-(2,f)$ and $\mathcal{ALCIF}_\ell$ of depth 2, it is undecidable whether for a given ontology $\mathcal{O}$,*

1. *query evaluation w.r.t. $\mathcal{O}$ is in* PTIME*, Datalog$^{\neq}$-rewritable, or* CONP*-hard (unless* PTIME = NP*);*
2. *$\mathcal{O}$ is materializable.*

PROOF. The proof is by reduction of the undecidable finite rectangle tiling problem. To establish both Points 1 and 2, it suffices to exhibit, for any such tiling problem $\mathfrak{P}$, an ontology $\mathcal{O}_\mathfrak{P}$ such that if $\mathfrak{P}$ admits a tiling, then $\mathcal{O}_\mathfrak{P}$ is not materializable and thus query evaluation w.r.t. $\mathcal{O}_\mathfrak{P}$ is CONP-hard and if $\mathfrak{P}$ admits no tiling, then query evaluation w.r.t. $\mathcal{O}_\mathfrak{P}$ is Datalog$^{\neq}$-rewritable and thus materializable (unless PTIME = NP).

The rectangle to be tiled is represented in input instances using the binary relations X and Y, and $\mathcal{O}_\mathfrak{P}$ declares these relations and their inverses to be functional. The main idea in the construction of $\mathcal{O}_\mathfrak{P}$ is to verify the existence of a properly tiled grid in the input instance by propagating markers from the top right corner to the lower left corner. During the propagation, one makes sure that grid cells close (that is, the XY-successor coincides with the

YX-successor) and that there is a tiling that satisfies the constraints in $\mathfrak{P}$. Once the existence of a properly tiled grid is completed, a disjunction is derived by $\mathcal{O}_\mathfrak{P}$ to achieve non-materializability and coNP-hardness. The challenge is to implement this construction such that when $\mathfrak{P}$ has no solution (and thus the verification of a properly tiled grid can never complete), $\mathcal{O}_\mathfrak{P}$ is Datalog$^{\neq}$-rewritable. In fact, achieving this involves a lot of technical subtleties.

A central issue is how to implement the markers (as formulas with one free variable) that are propagated through the grid during the verification. The markers must be designed in a way so that they cannot be 'preset' in the input instance as this would make it possible to prevent the verification of a (possibly defective) part of the input. In $\mathcal{ALCIF}_\ell$, we use formulas of the form $\exists^{=1} y P(x,y)$ while additionally stating in $\mathcal{O}_\mathfrak{P}$ that $\forall x \exists y P(x,y)$. Thus, the choice is only between whether a constant has exactly one P-successor (which means that the marker is set) or more than one P-successor (which means that the marker is not set). Clearly, this difference is invisible to queries and we cannot preset a marker in an input instance in the sense that we make it true at some constant. We can, however, easily make the marker false at a constant c by adding two P-successors to c in the input instance. It seems that this effect, which gives rise to many technical complications, can only be avoided by using marker formulas with higher quantifier depth which would result in $\mathcal{O}_\mathfrak{P}$ not falling within $\mathcal{ALCIF}_\ell$ depth 2. For uGF$_2^-(2,f)$ we work with $\neg\exists y(P(x,y) \wedge \neg F(x,y))$, where F is a function for which we state $\forall x F(x,x)$ (as in the CSP encoding).

Full proof details can be found in the long version. We only mention that closing of a grid cell is verified by using marker formulas as second-order variables. □

Theorem 11 *For the ontology languages uGF$_2^-(2,f)$ and $\mathcal{ALCIF}_\ell$ of depth 2, there is no dichotomy between* PTIME *and* coNP *(unless* PTIME = coNP*).*

By Ladner's theorem [38], there is a non-deterministic polynomial time Turing machine (TM) whose word problem is neither in PTIME nor NP-hard (unless PTIME = coNP). Ideally, we would like to reduce the word problem of such TMs to prove Theorem 11. However, this does not appear to be easily possible, for the following reason. In the reduction, we use a grid construction and marker formulas as in the proof of Theorem 10, with the grid providing the space in which the run of the TM is simulated and markers representing TM states and tape symbols. We cannot avoid that the markers can be preset either positively or negatively in the input (depending on the marker formulas we choose), which means that some parts of the run are not 'free', but might be predetermined or at least constrained in some way. We solve this problem by first establishing an appropriate variation of Ladner's theorem.

We consider non-deterministic TMs M with a single one-sided infinite tape. Configurations of M are represented by strings vqw, where q is the state, and v and w are the contents of the tape to the left and to the right of the tape head, respectively. A *partial configuration* of M is obtained from a configuration γ of M by replacing some or all symbols of γ by a wildcard symbol $\star$. A partial configuration $\tilde{\gamma}$ *matches* a configuration γ if it has the same length and agrees with γ on all non-wildcard symbols. A *partial run* of M is a finite sequence $\tilde{\gamma}_0, \ldots, \tilde{\gamma}_m$ of partial configurations of M of the same length. It is a *run* if each $\tilde{\gamma}_i$ is a configuration, and it *matches* a run $\gamma_0, \ldots, \gamma_n$ if $m = n$ and each $\tilde{\gamma}_i$ matches γ_i. A run is accepting if its last configuration has an accepting state. Note that runs need not start in any specific configuration (unless specified by a partial run that they extend). The *run fitting problem* for M is to decide whether a given partial run of M matches some accepting run of M. It is easy to see that for any TM M, the run fitting problem for M is in NP. We prove the following result in the long version by a careful adaptation of the proof of Ladner's theorem given in [2].

Theorem 12 *There is a non-deterministic Turing machine whose run fitting problem is neither in* PTIME *nor* NP*-hard (unless* PTIME = NP*).*

Now Theorem 11 is a consequence of the following lemma.

Lemma 4 *For every Turing machine M, there is a uGF$_2^-(2,f)$ ontology $\mathcal{O}$ and an $\mathcal{ALCIF}_\ell$ ontology $\mathcal{O}$ of depth 2 such that the following hold, where N is a distinguished unary relation:*

1. *there is a polynomial reduction of the run fitting problem for M to the complement of evaluating the OMQ $(\mathcal{O}, q \leftarrow N(x))$;*
2. *for every UCQ q, evaluating the OMQ $(\mathcal{O}, q)$ is polynomially reducible to the complement of the run fitting problem for M.*

To establish Lemma 4, we re-use the ontology $\mathcal{O}_\mathfrak{P}$ from the proof of Theorem 10, using a trivial rectangle tiling problem. When the existence of the grid has been verified, instead of triggering a disjunction as before, we now start a simulation of M on the grid. For both $\mathcal{ALCIF}_\ell$ and uGF$_2^-(2,f)$, we represent states q and tape symbols G using the same formulas as in the CSP encoding of homomorphisms. Thus, for $\mathcal{ALCIF}_\ell$ we use formulas $\exists^{\geq 2} y q(x,y)$ and $\exists^{\geq 2} y G(x,y)$, respectively, using q and G as binary relations. Note that here the encoding $\exists^{=1} y q(x,y)$ from the tiling problem does not work because states and tape symbols can be positively preset in the input instance rather than negatively, which is in correspondence with the run fitting problem.

8. DECISION PROBLEMS

We study the decidability and complexity of the problem to decide whether a given ontology admits PTIME query evaluation. Realistically, we can only hope for positive results in cases where there is a dichotomy between PTIME and coNP: first, we have shown in Section 7 that for cases with provably no such dichotomy, meta problems are typically undecidable; and second, it does not seem very likely that in the CSP-hard cases, one can decide whether an ontology admits PTIME query evaluation without resolving the dichotomy question and thus solving the Feder-Vardi conjecture. Our main results are EXPTIME-completeness of deciding PTIME-query evalutation of $\mathcal{ALCHIQ}$ ontologies of depth one (the same complexity as for satisfiability) and a NEXPTIME upper bound for uGC$_2^-(1,=)$ ontologies. Note that, in both of the considered languages, PTIME-query evalutation coincides with rewritability into Datalog$^{\neq}$. We remind the reader that according to our experiments, a large majority of real world ontologies are $\mathcal{ALCHIQ}$ ontologies of depth 1. We also show that for $\mathcal{ALC}$ ontologies of depth 2, the mentioned problem is NEXPTIME-hard.

Since the ontology languages relevant here admit at most binary relations, an interpretation $\mathfrak{B}$ is cg-tree decomposable if and only if the undirected graph $G_\mathfrak{B} = \{\{a,b\} \mid R(a,b) \in \mathfrak{B}, a \neq b\}$ is a tree. For simplicity, we speak of *tree interpretations* and of *tree instances*, defined likewise. The *outdegree of* $\mathfrak{B}$ is the outdegree of $G_\mathfrak{B}$.

Theorem 13 *For uGC$_2^-(1,=)$ ontologies, deciding whether query evaluation w.r.t. a given ontology is in* PTIME *(equivalently: rewritable into Datalog$^{\neq}$) is in* NEXPTIME*. For $\mathcal{ALCHIQ}$ ontologies of depth 1, this problem is in* EXPTIME*-complete.*

The main insight underlying the proof of Theorem 13 is that for ontologies formulated in the mentioned languages, materializability (which by Theorem 7 coincides with PTIME query evaluation) already follows from the existence of materializations for tree instances of depth 1. We make this precise in the following lemma. Given a tree interpretation $\mathfrak{B}$ and $a \in \text{dom}(\mathfrak{B})$, define the *1-neighbourhood* $\mathfrak{B}_a^{\leq 1}$ of a in $\mathfrak{B}$ as $\mathfrak{B}_{|X}$, where X is the union of all guarded sets in $\mathfrak{B}$ that contain a. $\mathfrak{B}$ is a *bouquet with root* a if $\mathfrak{B}_a^{\leq 1} = \mathfrak{B}$ and it is *irreflexive* if there exists no atom of the form $R(b, b)$ in $\mathfrak{B}$.

Lemma 5 *Let $\mathcal{O}$ be a uGC$_2^-(1, =)$ ontology (resp. an $\mathcal{ALCHIQ}$ ontology of depth 1). Then $\mathcal{O}$ is materializable iff $\mathcal{O}$ is materializable for the class of all (respectively, all irreflexive) bouquets $\mathfrak{D}$ of outdegree $\leq |\mathcal{O}|$ with sig$(\mathfrak{D}) \subseteq$ sig$(\mathcal{O})$.*

PROOF. We require some notation. An instance $\mathfrak{D}$ is called $\mathcal{O}$-*saturated* for an ontology $\mathcal{O}$ if for all facts $R(\vec{a})$ with $\vec{a} \subseteq \text{dom}(\mathfrak{D})$ such that $\mathcal{O}, \mathfrak{D} \models R(\vec{a})$ it follows that $R(\vec{a}) \in \mathfrak{D}$. For every $\mathcal{O}$ and instance $\mathfrak{D}$ there exists a unique minimal (w.r.t. set-inclusion) $\mathcal{O}$-saturated instance $\mathfrak{D}_\mathcal{O} \supseteq \mathfrak{D}$. We call $\mathfrak{D}_\mathcal{O}$ the $\mathcal{O}$-*saturation of* $\mathfrak{D}$. It is easy to see that there is a materialization of $\mathcal{O}$ and $\mathfrak{D}$ if and only if there is a materialization of $\mathcal{O}$ and the $\mathcal{O}$-saturation of $\mathfrak{D}$.

We first prove Lemma 5 for uGC$_2^-(1, =)$ ontologies $\mathcal{O}$ and without the condition on the outdegree. Let $\Sigma_0 =$ sig$(\mathcal{O})$ and assume that $\mathcal{O}$ is materializable for the class of all Σ_0-bouquets. By Theorem 7 it suffices to prove that $\mathcal{O}$ is materializable for the class of Σ_0-tree instances. Fix a Σ_0-tree instance $\mathfrak{D}$ that is consistent w.r.t. $\mathcal{O}$. We may assume that $\mathfrak{D}$ is $\mathcal{O}$-saturated. Note that a forest model materialization $\mathfrak{B}$ of an ontology $\mathcal{O}$ and an $\mathcal{O}$-saturated instance $\mathfrak{F}$ consists of $\mathfrak{F}$ and tree interpretations $\mathfrak{B}_a$, $a \in \text{dom}(\mathfrak{F})$, that are hooked to $\mathfrak{F}$ at a. Take for any $a \in \text{dom}(\mathfrak{D})$ the bouquet $\mathfrak{D}_a^{\leq 1}$ with root a and hook to $\mathfrak{D}$ at a the interpretation $\mathfrak{B}_a$ that is hooked to $\mathfrak{D}_a^{\leq 1}$ at a in a forest model materialization $\mathfrak{B}$ of $\mathfrak{D}_a^{\leq 1}$ and $\mathcal{O}$ (such a forest model materialization exists since $\mathfrak{D}_a^{\leq 1}$ is materializable). Denote by $\mathfrak{A}$ the resulting interpretation. Using the condition that $\mathcal{O}$ is a uGC$_2^-(1, =)$ ontology it is not difficult to prove that $\mathfrak{A}$ is a materialization of $\mathcal{O}$ and $\mathfrak{D}$.

We now prove the restriction on the outdegree. Assume $\mathcal{O}$ is given. Let $\mathfrak{D}$ be a bouquet with root a of minimal outdegree such that there is no materialization of $\mathcal{O}$ and $\mathfrak{D}$. We show that the outdegree of $\mathfrak{D}$ does not exceed $|\mathcal{O}|$. Assume the outdegree of $\mathfrak{D}$ is at least three (otherwise we are done). We may assume that $\mathfrak{D}$ is $\mathcal{O}$-saturated. Take for any formula $\chi = \exists^{\geq n} z_1 \alpha(z_1, z_2) \wedge \varphi(z_1, z_2)$ that occurs as a subformula in $\mathcal{O}$ the set Z_χ of all $b \neq a$ such that $\mathfrak{D} \models \alpha(b, a) \wedge \varphi(b, a)$. Let $Z'_\chi = Z_\chi$ if $|Z_\chi| \leq n + 1$; otherwise let Z'_χ be a subset of Z_χ of cardinality $n + 1$. Let $\mathfrak{D}'$ be the restriction $\mathfrak{D}_{|Z}$ of $\mathfrak{D}$ to the union Z of all Z'_χ and $\{a\}$. We show that there exists no materialization of $\mathfrak{D}'$ and $\mathcal{O}$. Assume for a proof by contradiction that there is a materialization $\mathfrak{B}$ of $\mathfrak{D}'$. Let $\mathfrak{B}'$ be the union of $\mathfrak{D} \cup \mathfrak{B}$ and the interpretations $\mathfrak{B}_b$, $b \in \text{dom}(\mathfrak{D}) \setminus (Z \cup \{a\})$, that are hooked to $\mathfrak{D}_{|\{a,b\}}$ at b in a forest model materialization of $\mathfrak{D}_{|\{a,b\}}$. We show that $\mathfrak{B}'$ is a materialization of $\mathfrak{D}$ and $\mathcal{O}$ (and thus derive a contradiction). Using the condition that $\mathfrak{D}$ is $\mathcal{O}$-saturated one can show that the restriction $\mathfrak{B}'_{|\text{dom}(\mathfrak{D})}$ of $\mathfrak{B}'$ to dom$(\mathfrak{D})$ coincides with $\mathfrak{D}$. Using the condition that $\mathcal{O}$ has depth 1 it is now easy to show that $\mathfrak{B}'$ is a model of $\mathcal{O}$. It is a materialization of $\mathfrak{D}$ and $\mathcal{O}$ since it is composed of materializations of subinstances of $\mathfrak{D}$ and $\mathcal{O}$.

The proof that irreflexive bouquets are sufficient for ontologies of depth 1 in $\mathcal{ALCHIQ}$ is similar to the proof above and uses the fact that one can always unravel models of $\mathcal{ALCHIQ}$ ontologies into irreflexive tree models. □

We now develop algorithms that decide PTIME query evaluation by checking the conditions given in Lemma 5, starting with the (easier) case of $\mathcal{ALCHIQ}$. Let $\mathfrak{D}$ be a bouquet with root a. Call a bouquet $\mathfrak{B} \supseteq \mathfrak{D}$ a *1-materialization of* $\mathcal{O}$ *and* $\mathfrak{D}$ if

- there exists a model $\mathfrak{A}$ of $\mathcal{O}$ and $\mathfrak{D}$ such that $\mathfrak{B} = \mathfrak{A}_a^{\leq 1}$;
- for any model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$ there exists a homomorphism from $\mathfrak{B}$ to $\mathfrak{A}$ that preserves dom$(\mathfrak{D})$.

It turns out that, when checking materializability, not only is it sufficient to consider bouquets instead of unrestricted instances, but additionally one can concentrate on 1-materializations of bouquets.

Lemma 6 *Let $\mathcal{O}$ be an $\mathcal{ALCHIQ}$ ontology of depth 1. If for all irreflexive bouquets $\mathfrak{D}$ that are consistent w.r.t. $\mathcal{O}$, of outdegree $\leq |\mathcal{O}|$, and satisfy sig$(\mathfrak{D}) \subseteq$ sig$(\mathcal{O})$ there is a 1-materialization of $\mathcal{O}$ and $\mathfrak{D}$, then $\mathcal{O}$ is materializable for the class of all such bouquets.*

PROOF. For brevity, we call an irreflexive bouquet $\mathfrak{F}$ *relevant* if it is consistent w.r.t. $\mathcal{O}$, of outdegree $\leq |\mathcal{O}|$ and satisfies sig$(\mathfrak{F}) \subseteq$ sig$(\mathcal{O})$. An *irreflexive 1-materializability witness* $(\mathfrak{F}, a, \mathfrak{B})$ consists of a relevant irreflexive bouquet $\mathfrak{F}$ with root a and a 1-materialization $\mathfrak{B}$ of $\mathfrak{F}$ w.r.t. $\mathcal{O}$. One can show that $\mathfrak{B}$ is an irreflexive tree interpretation.

Now let $\mathfrak{D}$ be a relevant irreflexive bouquet with root a and assume that $\mathfrak{D}$ is 1-materializable w.r.t. $\mathcal{O}$. We have to show that there exists a materialization of $\mathcal{O}$ and $\mathfrak{D}$. Note that for every relevant irreflexive bouquet $\mathfrak{F}$, there is a 1-materializability witness $(\mathfrak{F}, a, \mathfrak{B})$. We construct the desired materialization step-by-step using these pairs also memorizing sets of frontier elements that have to be expanded in the next step. We start with the irreflexive 1-materializability witness $(\mathfrak{D}, a, \mathfrak{B})$ and set $\mathfrak{B}^0 = \mathfrak{B}$ and $F_0 = \text{dom}(\mathfrak{B}) \setminus \{a\}$. Then we construct a sequence of irreflexive tree interpretations $\mathfrak{B}^0 \subseteq \mathfrak{B}^1 \subseteq \ldots$ and frontier sets $F_{i+1} \subseteq \text{dom}(\mathfrak{B}^{i+1}) \setminus \text{dom}(\mathfrak{B}^i)$ inductively as follows: given $\mathfrak{B}^i$ and F_i, take for any $b \in F_i$ its predecessor a in $\mathfrak{B}^i$ and an irreflexive 1-materializability witness $(\mathfrak{B}^i_{|\{a,b\}}, b, \mathfrak{B}_b)$ and set

$$\mathfrak{B}^{i+1} := \mathfrak{B}^i \cup \bigcup_{b \in F_i} \mathfrak{B}_b \qquad F_{i+1} := \bigcup_{b \in F_i} \text{dom}(\mathfrak{B}_b) \setminus \{b\}$$

Let $\mathfrak{B}^*$ be the union of all $\mathfrak{B}_i$. We show that $\mathfrak{B}$ is a materialization of $\mathcal{O}$ and $\mathfrak{D}$. $\mathfrak{B}$ is a model of $\mathcal{O}$ by construction since $\mathcal{O}$ is an $\mathcal{ALCHIQ}$ ontology of depth 1. Consider a model $\mathfrak{A}$ of $\mathcal{O}$ and $\mathfrak{D}$. It suffices to construct a homomorphism h from $\mathfrak{B}^*$ to $\mathfrak{A}$ that preserves dom$(\mathfrak{D})$. We may assume that $\mathfrak{A}$ is an irreflexive tree interpretation. We construct h as the limit of a sequence $h_0, \ldots$ of homomorphisms from $\mathfrak{B}^i$ to $\mathfrak{A}$. By definition, there exists a homomorphism h_0 from $\mathfrak{B}^0$ to $\mathfrak{A}_a^{\leq 1}$ preserving dom$(\mathfrak{D})$. Now, inductively, assume that h_i is a homomorphism from $\mathfrak{B}^i$ to $\mathfrak{A}$. Assume c has been added to $\mathfrak{B}^i$ in the construction of $\mathfrak{B}^{i+1}$. Then there exists $b \in F_i$ and its predecessor a in $\mathfrak{B}^i$ such that $c \in \text{dom}(\mathfrak{B}_b) \setminus \{b\}$, where $\mathfrak{B}_b$ is the irreflexive tree interpretation that has been added to $\mathfrak{B}^i$ as the last component of the irreflexive model pair $(\mathfrak{B}^i_{|\{a,b\}}, b, \mathfrak{B}_b)$. But then, as $\mathfrak{B}_b$ is a 1-materialization of $\mathfrak{B}^i_{|\{a,b\}}$ and h_i is injective on $\mathfrak{B}^i_{|\{a,b\}}$ (since $\mathfrak{A}$ is irreflexive), we can expand the homomorphism h_i to a homomorphism to $\mathfrak{A}$ with domain dom$(\mathfrak{B}^i) \cup \{c\}$. Thus, we can expand h_i to a homomorphism from $\mathfrak{B}^{i+1}$ to $\mathfrak{A}$. □

Lemma 5 and Lemma 6 imply that an $\mathcal{ALCHIQ}$ ontology $\mathcal{O}$ of depth 1 enjoys PTIME query evaluation if and only if all irreflexive bouquets $\mathfrak{D}$ that are consistent w.r.t. $\mathcal{O}$, of outdegree $\leq |\mathcal{O}|$, and satisfy sig$(\mathfrak{D}) \subseteq$ sig$(\mathcal{O})$ have a 1-materialization w.r.t. $\mathcal{O}$.

The latter condition can be checked in deterministic exponential time since the satisfiability problem for $\mathcal{ALCHIQ}$ ontologies is in EXPTIME. Moreover, there are only exponentially many relevant bouquets. We have thus proved the EXPTIME upper bound in Theorem 13. A matching lower bound can be proved by a straightforward reduction from satisfiability.

The following example shows that, in contrast to $\mathcal{ALCHIQ}$ depth 1, for $\text{uGC}_2^-(1,=)$ the existence of 1-materializations does not guarantee materializability of bouquets.

Example 7 *We use $\exists^{\neq} y W(x,y)$ to abbreviate $\exists y(W(x,y) \wedge (x \neq y))$ and likewise for $\exists^{\neq} y W(y,x)$. Let S, S', R, R' be binary relation symbols and $\mathcal{O}$ the $\text{uGF}_2^-(1,=)$ ontology $\mathcal{O}$ that contains*

$$\forall x\big(S(x,x) \rightarrow (R(x,x) \rightarrow (\exists^{\neq} y R(x,y) \vee \exists^{\neq} y S(x,y)))\big)$$
$$\forall x(\exists^{\neq} y W(y,x) \rightarrow \exists y W'(x,y))$$

where (W, W') range over $\{(R,R'),(S,S')\}$. Observe that for the instance $\mathfrak{D} = \{S(a,a), R(a,a)\}$ and the Boolean UCQ

$$q \leftarrow R'(x,y) \vee S'(x,y),$$

we have $\mathcal{O}, \mathfrak{D} \models q$. Also, for $q_R \leftarrow R'(x,y)$ and $q_S \leftarrow S'(x,y)$ we have $\mathcal{O}, \mathfrak{D} \not\models q_R$ and $\mathcal{O}, \mathfrak{D} \not\models q_S$. Thus, $\mathcal{O}$ is not materializable. It is, however, easy to show that for every bouquet $\mathfrak{D}$ there exists a 1-materialization of $\mathfrak{D}$ w.r.t. $\mathcal{O}$.

In $\text{uGC}_2^-(1,=)$, we thus have to check unrestricted materializability of bouquets, instead of 1-materializability. In fact, it suffices to consider materializations that are tree interpretations. To decide the existence of such materialization, we use a mosaic approach. In each mosaic piece, we essentially record a 1-neighborhood of the materialization, a 1-neighborhood of a model of the bouquet and ontology, and a homomorphism from the former to the latter. We then identify certain conditions that characterize when a set of mosaics can be assembled into a materialization in a way that is similar to the model construction in the proof of Lemma 6. There are actually two different kinds of mosaic pieces that we use, with one kind of piece explicitly addressing reflexive loops which, as illustrated by Example 7, are the reason why we cannot work with 1-materializations. The decision procedure then consists of guessing a set of mosaics and verifying that the required conditions are satisfied. Details are in the long version.

Theorem 13 only covers ontology languages of depth 1. It would be desirable to establish decidability also for ontology languages of depth 2 that enjoy a dichotomy between PTIME and CONP, such as $\text{uGF}_2^-(2)$. The following example shows that this requires more sophisticated techniques than those used above. In particular, materializability of bouquets does not imply materializability.

Example 8 *We give a family of $\mathcal{ALC}$-ontologies $(\mathcal{O}_n)_{n\geq 0}$ of depth 2 such that each $\mathcal{O}_n$ is materializable for the class of tree interpretations of depth at most $2^n - 1$ while it is not materializable. The idea is that any instance $\mathfrak{D}$ that witnesses non-materializability of $\mathcal{O}_n$ must contain an R-chain of length 2^n, R a binary relation. The presence of this chain is verified by propagating a marker upwards along the chain. To avoid that $\mathcal{O}$ is 1-materializable, we represent this marker by a universally quantified formula and also hide some other unary predicates in the same way. For each unary predicate P, let $H_P(x)$ denote the formula $\forall y(S(x,y) \rightarrow P(y))$ and include in $\mathcal{O}_n$ the sentence $\forall x \exists y(S(x,y) \wedge P(y))$. The remaining sentences in $\mathcal{O}_n$ are:*

$$\overline{X}_1(x) \wedge \cdots \wedge \overline{X}_n(x) \rightarrow H_V(x)$$
$$X_i(x) \wedge \exists R.(X_i(y) \wedge X_j(y)) \rightarrow H_{\mathsf{ok}_i}(x)$$
$$\overline{X}_i(x) \wedge \exists R.(\overline{X}_i(y) \wedge X_j(y)) \rightarrow H_{\mathsf{ok}_i}(x)$$
$$X_i(x) \wedge \exists R.(\overline{X}_i(y) \wedge X_1(y) \wedge \cdots \wedge X_{i-1}(y)) \rightarrow H_{\mathsf{ok}_i}(x)$$
$$\overline{X}_i(x) \wedge \exists R.(X_i(y) \wedge X_1(y) \wedge \cdots \wedge \overline{X}_{i-1}(y)) \rightarrow H_{\mathsf{ok}_i}(x)$$
$$H_{\mathsf{ok}_1}(x) \wedge \cdots \wedge H_{\mathsf{ok}_n}(x) \wedge \exists R.H_V(y) \rightarrow H_V(x)$$
$$\exists R.X_i(y) \wedge \exists R.\overline{X}_i(y) \rightarrow \bot$$
$$X_1(x) \wedge \cdots \wedge X_n(x) \wedge H_V(x) \rightarrow B_1(x) \vee B_2(x)$$

where x is universally quantified, $\exists R.\varphi(y)$ is an abbreviation for $\exists y(R(x,y) \wedge \varphi(y))$, and i ranges over 1..n. Note that $X_1, \ldots, X_n$ and $\overline{X}_1, \ldots, \overline{X}_n$ represent a binary counter and that lines two to five implement incrementation of this counter. The second last formula is necessary to avoid that multiple successors of a node interact in undesired ways. On instances that contain no R-chain of length 2^n, a materialization can be constructed by a straightforward chase procedure.

We also observe that the ideas from Example 8 gives rise to a NEXPTIME lower bound.

Theorem 14 *For $\mathcal{ALC}$ ontologies of depth 2, deciding whether query-evaluation is in* PTIME *is* NEXPTIME*-hard (unless* PTIME = CONP*).*

We remark that the decidability of PTIME query evaluation of $\mathcal{ALC}$ ontologies of depth 2 remains open.

9. CONCLUSION

Perhaps the most surprizing result of our analysis is that it is possible to escape Ladner's Theorem and prove a PTIME/CONP dichotomy for query evaluation for rather large subsets of the guarded fragment that cover almost all practically relevant DL ontologies. This result comes with a characterization of PTIME query evaluation in terms of materializability and unravelling tolerance, with the guarantee that PTIME query evaluation coincides with Datalog$^{\neq}$-rewritability, and with decidability of (and complexity results for) meta problems such as deciding whether a given ontology enjoys PTIME query evaluation. Our study also shows that when we increase the expressive power in seemingly harmless ways, then often there is provably no PTIME/CONP dichotomy or one obtains CSP-hardness. The proof of the non-dichotomy results comes with a variation of Ladner's Theorem that could prove useful in other contexts where some form of precoloring of the input is unavoidable, such as in consistent query answering [43].

There are a number of interesting future research questions. The main open questions regarding dichotomies are whether the PTIME/CONP dichotomy can be generalized from $\text{uGF}_2^-(2)$ to $\text{uGF}^-(2)$ and whether the CSP-hardness results can be sharpened to either CSP-equivalence results (this is known for $\mathcal{ALC}$ ontologies of depth 3 [42]) or to non-dichotomy results. Also of interest is the complexity of deciding PTIME query evaluation for $\text{uGF}(1)$, where the characterization of PTIME query evaluation via hom-universal models fails. Improving our current complexity results to tight complexity bounds for PTIME query evaluation for $\mathcal{ALCHIF}$ ontologies of depth 2 and $\text{uGF}_2^-(2)$ ontologies appears to be challenging as well. It would also be interesting to study the case where invariance under disjoint union is not guaranteed (as we have observed, the complexities of CQ and UCQ evaluation might then diverge), and to add the ability to declare in an ontology that a binary relation is transitive.

10. ACKNOWLEDGMENTS

André Hernich, Fabio Papacchini, and Frank Wolter were supported by EPSRC UK grant EP/M012646/1. Carsten Lutz was supported by ERC CoG 647289 CODA.

11. REFERENCES

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

[2] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[3] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *J. of Artifical Intelligence Research*, 36:1–69, 2009.

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

[5] V. Bárány, G. Gottlob, and M. Otto. Querying the guarded fragment. *Logical Methods in Computer Science*, 10(2), 2014.

[6] V. Bárány, B. ten Cate, and L. Segoufin. Guarded negation. *J. ACM*, 62(3):22:1–22:26, 2015.

[7] L. Barto. Constraint satisfaction problem and universal algebra. *SIGLOG News*, 1(2):14–24, 2014.

[8] C. Beeri and P. A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Trans. Database Syst.*, 4(1):30–59, 1979.

[9] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[10] M. Bienvenu and M. Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Proc. of Reasoning Web*, pages 218–307, 2015.

[11] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter. Ontology-based data access: A study through disjunctive datalog, csp, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

[12] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res. (JAIR)*, 48:115–174, 2013.

[13] A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

[14] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. *Artificial Intelligence*, 195:335–360, 2013.

[15] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS*, pages 149–158, 1998.

[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. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[17] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing and constraint satisfaction. In *LICS*, pages 361–371, 2000.

[18] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query containment. In *PODS*, pages 56–67, 2003.

[19] D. Cohen and P. Jeavons. *The complexity of constraint languages*, chapter 8. Elsevier, 2006.

[20] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In M. Lenzerini and D. Lembo, editors, *Proc. of PODS*, pages 149–158. ACM, 2008.

[21] R. Fagin, B. Kimelfeld, and P. G. Kolaitis. Dichotomies in the complexity of preferred repairs. In *Proc. of PODS*, pages 3–15, 2015.

[22] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

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

[24] G. Fontaine. Why is it hard to obtain a dichotomy for consistent query answering? *ACM Trans. Comput. Log.*, 16(1):7:1–7:24, 2015.

[25] C. Freire, W. Gatterbauer, N. Immerman, and A. Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries. *PVLDB*, 9(3):180–191, 2015.

[26] G. Gottlob, S. Kikot, R. Kontchakov, V. V. Podolskii, T. Schwentick, and M. Zakharyaschev. The price of query rewriting in ontology-based data access. *Artif. Intell.*, 213:42–59, 2014.

[27] G. Gottlob, M. Manna, and A. Pieris. Polynomial rewritings for linear existential rules. In *Proc. of IJCAI*, pages 2992–2998, 2015.

[28] G. Gottlob, G. Orsi, and A. Pieris. Query rewriting and optimization for ontological databases. *ACM Trans. Database Syst.*, 39(3):25:1–25:46, 2014.

[29] E. Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999.

[30] E. Grädel and M. Otto. The freedoms of (guarded) bisimulation. In *Johan van Benthem on Logic and Information Dynamics*, pages 3–31. 2014.

[31] M. Kaminski, Y. Nenov, and B. C. Grau. Datalog rewritability of disjunctive datalog programs and non-horn ontologies. *Artif. Intell.*, 236:90–118, 2016.

[32] Y. Kazakov. A polynomial translation from the two-variable guarded fragment with number restrictions to the guarded fragment. In *Proc. of JELIA*, pages 372–384, 2004.

[33] B. Kimelfeld. A dichotomy in the complexity of deletion propagation with functional dependencies. In M. Benedikt, M. Krötzsch, and M. Lenzerini, editors, *Proc. of PODS*, pages 191–202. ACM, 2012.

[34] R. Kontchakov and M. Zakharyaschev. An introduction to description logics and query rewriting. In *Proc. of Reasoning Web*, pages 195–244, 2014.

[35] P. Koutris and D. Suciu. A dichotomy on the complexity of consistent query answering for atoms with simple keys. In *Proc. of ICDT*, pages 165–176. OpenProceedings.org, 2014.

[36] P. Koutris and J. Wijsen. The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In *Proc. of PODS*, pages 17–29, 2015.

[37] M. Krötzsch. OWL 2 profiles: An introduction to lightweight ontology languages. In *Proc. of Reasoning Web*, pages 112–183, 2012.

[38] R. E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

[39] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410(18):1629–1647, 2009.

[40] A. Y. Levy and M. Rousset. Combining horn rules and description logics in CARIN. *Artif. Intell.*, 104(1-2):165–209, 1998.

[41] C. Lutz, I. Seylan, and F. Wolter. Ontology-based data access with closed predicates is inherently intractable(sometimes). In *Proc. of IJCAI*, pages 1024–1030, 2013.

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

[43] C. Lutz and F. Wolter. On the relationship between consistent query answering and constraint satisfaction problems. In *Proc. of ICDT*, pages 363–379, 2015.

[44] J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Elsevier, 1988.

[45] M. Mugnier and M. Thomazo. An introduction to ontology-based query answering with existential rules. In *Proc. of Reasoning Web*, pages 245–278, 2014.

[46] M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008.

[47] A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[48] I. Pratt-Hartmann. Complexity of the guarded two-variable fragment with counting quantifiers. *J. Log. Comput.*, 17(1):133–155, 2007.

[49] A. Rafiey, J. Kinne, and T. Feder. Dichotomy for digraph homomorphism problems. *CoRR*, abs/1701.02409, 2017.

[50] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intel. Inf. Systems*, 2:265–278, 1993.

[51] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[52] P. L. Whetzel, N. F. Noy, N. H. Shah, P. R. Alexander, C. Nyulas, T. Tudorache, and M. A. Musen. BioPortal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic acids research*, 39(suppl 2):W541–W545, 2011.

APPENDIX

A. INTRODUCTION TO DESCRIPTION LOGIC

We give a brief introduction to the syntax and semantics of DLs and establish their relationship to the guarded fragment of FO. We consider the DL $\mathcal{ALC}$ and its extensions by inverse roles, role inclusions, qualified number restrictions, functional roles, and local functionality. Recall that $\mathcal{ALC}$*-concepts* are constructed according to the rule

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

where A ranges over unary relations and R ranges over binary relations. DLs extended by *inverse roles* (denoted in the name of a DL by the letter $\mathcal{I}$) admit, in addition, *inverse relations* denoted by R^-, with R a relation. Thus, in $\mathcal{ALCI}$ inverse relations can be used in place of relations in any $\mathcal{ALC}$ concept. DLs extended by *qualified number restrictions* (denoted by $\mathcal{Q}$) admit concepts of the form $(\geq\ n\ R\ C)$, $(=\ n\ R\ C)$, and $(\leq\ n\ R\ C)$, where $n \geq 1$ is a natural number, R is a relation or an inverse relation (if inverse relations are in the original DL), and C is a concept. When extending a DL with *local functionality* (denoted by $\mathcal{F}_\ell$) one can use only number restrictions of the form $(\leq 1\ R\ \top)$ in which R is a relation or an inverse relation (if inverse relations are in the original DL). We abbreviate $(\leq\ 1\ R\ \top)$ with $(\leq 1R)$ and use $(=\ 1R)$ as an abbreviation for $(\exists R.\top) \sqcap (\leq 1R)$ and $(\geq 2R)$ as an abbreviation for $(\exists R.\top) \sqcap \neg(\leq 1R)$.

In DLs, ontologies are formalized as finite sets of *concept inclusions* $C \sqsubseteq D$, where C, D are concepts in the respective language. We use $C \equiv D$ as an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. In the DLs extended with *functionality* (denoted by $\mathcal{F}$) one can use *functionality assertions* $\mathsf{func}(R)$, where R is a relation or an inverse relation (if present in the original DL). Such an R is interpreted as a partial function. Extending a DL with *role inclusions* (denoted by $\mathcal{H}$) allows one to use expressions of the form $R \sqsubseteq S$, where R and S are relations or inverse relations (if present in the original DL), and which state that R is a subset of S.

The semantics of DLs is given by interpretations $\mathfrak{A}$. The interpretation $C^{\mathfrak{A}}$ of a concept C in an interpretation $\mathfrak{A}$ is defined inductively as follows:

$$\begin{aligned}
\top^{\mathfrak{A}} &= \mathrm{dom}(\mathfrak{A}) & \bot^{\mathfrak{A}} &= \emptyset \\
A^{\mathfrak{A}} &= \{a \in \mathrm{dom}(\mathfrak{A}) \mid A(a) \in \mathfrak{A}\} & (\neg C)^{\mathfrak{A}} &= \mathrm{dom}(\mathfrak{A}) \setminus C^{\mathfrak{A}} \\
(C \sqcap D)^{\mathfrak{A}} &= C^{\mathfrak{A}} \cap D^{\mathfrak{A}} & (C \sqcup D)^{\mathfrak{A}} &= C^{\mathfrak{A}} \cup D^{\mathfrak{A}} \\
(\exists R.C)^{\mathfrak{A}} &= \{a \in \mathrm{dom}(\mathfrak{A}) \mid \exists a' :\ R(a,a') \in \mathfrak{A} \text{ and } a' \in C^{\mathfrak{A}}\} \\
(\forall R.C)^{\mathfrak{A}} &= \{a \in \mathrm{dom}(\mathfrak{A}) \mid \forall a' :\ R(a,a') \in \mathfrak{A} \text{ implies } a' \in A^{\mathfrak{A}}\} \\
(\geq n\ R\ C)^{\mathfrak{A}} &= \{a \in \mathrm{dom}(\mathfrak{A}) \mid |\{b \mid R(a,b) \in \mathfrak{A} \text{ and } b \in C^{\mathfrak{A}}\}| \geq n\} \\
(\leq n\ R\ C)^{\mathfrak{A}} &= \{a \in \mathrm{dom}(\mathfrak{A}) \mid |\{b \mid R(a,b) \in \mathfrak{A} \text{ and } b \in C^{\mathfrak{A}}\}| \leq n\} \\
(= n\ R\ C)^{\mathfrak{A}} &= \{a \in \mathrm{dom}(\mathfrak{A}) \mid |\{b \mid R(a,b) \in \mathfrak{A} \text{ and } b \in C^{\mathfrak{A}}\}| = n\}
\end{aligned}$$

Then $\mathfrak{A}$ *satisfies* a concept inclusion $C \sqsubseteq D$ if $C^{\mathfrak{A}} \subseteq D^{\mathfrak{A}}$. Alternatively, one can define the semantics of DLs by translating them into FO; the following table gives such a translation:

$$\begin{aligned}
\top^*(x) &= \top & \bot^*(x) &= \bot \\
A^*(x) &= A(x) & (\neg C)^*(x) &= \neg(C^*(x)) \\
(C \sqcap D)^*(x) &= C^*(x) \wedge D^*(x) & (C \sqcup D)^*(x) &= C^*(x) \vee D^*(x) \\
(\exists R.C)^*(x) &= \exists y\,(R(x,y) \wedge C^*(y)) \\
(\forall R.C)^*(x) &= \forall y\,(R(x,y) \rightarrow C^*(y)) \\
(\geq n\ R\ C)^*(x) &= \exists^{\geq n} y (R(x,y) \wedge C^*(y))
\end{aligned}$$

We observe the following relationships between DLs and fragments of the guarded fragment. For a DL $\mathcal{L}$ and fragment $\mathcal{L}'$ of the guarded fragment we say that an $\mathcal{L}$ *ontology* $\mathcal{O}$ *can be written as an* $\mathcal{L}'$ *ontology* if the translation given above translates $\mathcal{O}$ into an $\mathcal{L}'$ ontology.

Lemma 7 *The following inclusions hold:*

1. *Every $\mathcal{ALCHI}$ ontology can be written as a uGF$_2$ ontology. If the ontology has depth 2, then it can be written as a uGF$_2^-(2)$ ontology.*
2. *Every $\mathcal{ALCHIF}$ ontology can be written as a uGF$_2^-(f)$ ontology.*
3. *Every $\mathcal{ALCHIQ}$ ontology can be written as a uGC$_2$ ontology. If the ontology has depth 1, then it can be written as a uGC$_2^-(1)$ ontology.*

B. INTRODUCTION TO DATALOG

We give a brief introduction to the notation used for Datalog. A *datalog$^{\neq}$ rule* ρ takes the form

$$S(\vec{x}) \leftarrow R_1(\vec{x}_1) \wedge \cdots \wedge R_m(\vec{x}_m)$$

where S is a relation symbol, $m \geq 1$, and $R_1, \ldots, R_m$ are either relation symbols or the symbol $\neq$ for inequality. We call $S(\vec{x})$ the *head* of ρ and $R_1(\vec{x}_1) \wedge \cdots \wedge R_m(\vec{x}_m)$ its *body*. Every variable in the head of ρ is required to occur in its body. We call a datalog$^{\neq}$ rule that does not use inequality a *datalog rule*. A *Datalog$^{\neq}$ program* is a finite set Π of datalog$^{\neq}$ rules with a selected *goal relation symbol* goal that does not occur in rule bodies in Π and only in *goal rules* of the form $\mathrm{goal}(\vec{x}) \leftarrow R_1(\vec{x}_1) \wedge \cdots \wedge R_m(\vec{x}_m)$. The *arity* of Π is the arity of its goal relation. A *Datalog program* is a Datalog$^{\neq}$ program not using inequality.

For every instance $\mathfrak{D}$ and Datalog$^{\neq}$ program Π, we call a model $\mathfrak{A}$ of $\mathfrak{D}$ a *model of* Π if $\mathfrak{A}$ is a model of all FO sentences $\forall\vec{x}\forall\vec{x}_1 \cdots \forall\vec{x}_m(R_1(\vec{x}_1) \wedge \cdots \wedge R_m(\vec{x}_m) \rightarrow S(\vec{x}))$ with $S(\vec{x}) \leftarrow R_1(\vec{x}_1) \wedge \cdots \wedge R_m(\vec{x}_m) \in \Pi$. We set $\mathfrak{D} \models \Pi(\vec{a})$ if $\mathrm{goal}(\vec{a}) \in \mathfrak{A}$ for all models $\mathfrak{A}$ of $\mathfrak{D}$ and Π.