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

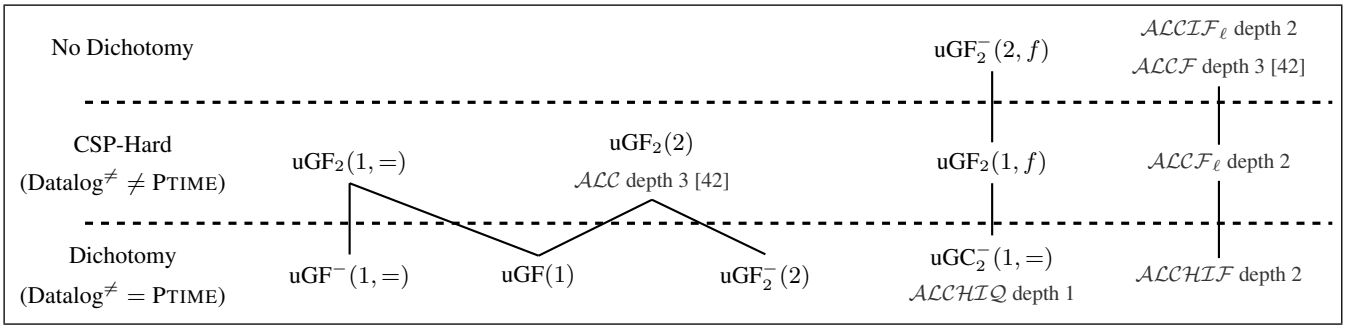


Figure 1: Summary of the results—Number in brackets indicates depth, f presence of partial functions, $\cdot_2$ restriction to two variables, $\cdot^-$ restricts outermost guards to be equality, $\mathcal{F}$ globally function roles, $\mathcal{F}_\ell$ concepts $(\leq 1\, R)$.

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 appendix.

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 $\mathrm{sig}(\mathfrak{A})$ and $\mathrm{dom}(\mathfrak{A})$ to denote the set of relation symbols and, respectively, constants and labelled nulls in $\mathfrak{A}$. We always assume that $\mathrm{sig}(\mathfrak{A})$ is finite while $\mathrm{dom}(\mathfrak{A})$ can be infinite. Whenever convenient, interpretations $\mathfrak{A}$ are presented in the form $(A, (R^{\mathfrak{A}})_{R \in \mathrm{sig}(\mathfrak{A})})$ where $A = \mathrm{dom}(\mathfrak{A})$ and $R^{\mathfrak{A}}$ is a k-ary relation on A for each $R \in \mathrm{sig}(\mathfrak{A})$ 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 $\mathrm{dom}(\mathfrak{A})$ to $\mathrm{dom}(\mathfrak{B})$ 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 $a_1, \ldots, a_k \in \mathrm{dom}(\mathfrak{A})$ 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 $a_1, \ldots, a_k \in \mathrm{dom}(\mathfrak{A})$ implies $R(a_1, \ldots, a_k) \in \mathfrak{A}$; if $\mathrm{dom}(\mathfrak{A}) = A$, 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 $\mathrm{dom}(\mathfrak{D})$ 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 $\vec{a} \in \mathrm{dom}(\mathfrak{D})$, $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ 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 appendix 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) $G_i \neq G_{i+1}$,

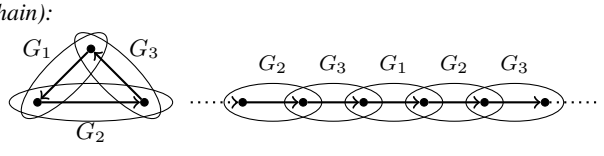
(b) $G_i \cap G_{i+1} \neq \emptyset$, and

(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.

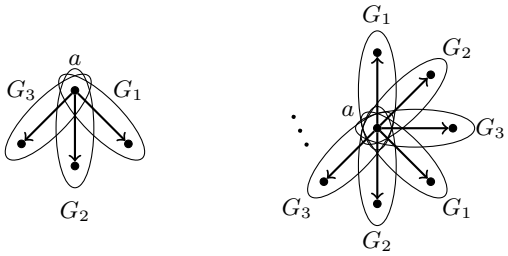
Set $\mathsf{tail}(G_0 \cdots G_n) = G_n$. Take an infinite supply of *copies* of any $d \in \mathrm{dom}(\mathfrak{D})$. We set $e^{\uparrow} = d$ if e is a copy of d. We define $\mathrm{bag}(t)$ (up to isomorphism) proceeding by induction on the length of the sequence t. For any $t = G$, $\mathrm{bag}(t)$ 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 bottommost 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 $\mathfrak{D}^u$.

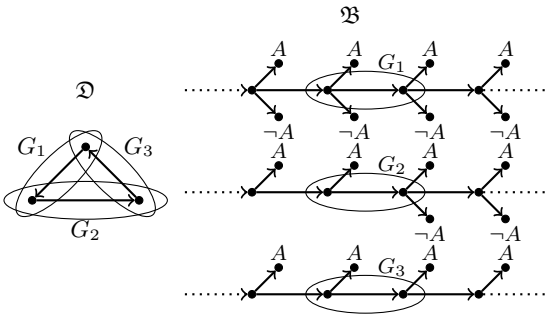
By Lemma 1 we may assume that $\mathfrak{B}$ is a forest model which is obtained from $\mathfrak{D}^u$ by hooking cg-tree decomposable $\mathfrak{B}_{\text{bag}(t)}$ to maximal guarded bag(t) in $\mathfrak{D}^u$. Now we would like to obtain a model of $\mathcal{O}$ and the original $\mathfrak{D}$ by 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 $a \in$ 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 $\{\text{rAQ}, \text{CQ}, \text{UCQ}\}$ 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 appendix, we show that $\mathcal{O}$ satisfies Conditions 1 and 2 from Definition 4 where the 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 appendix. 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 appendix 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 appendix.

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.

C. PROOFS FOR SECTION 2

Following the notation introduced after Theorem 1 we denote the guarded fragment with equality by GF(=) and uGF with equality in non-guard positions by uGF(=). In contrast, GF and uGF denote the corresponding languages without equality in non-guard positions. We use this notation in the formulation of Theorem 1 below.

Theorem 1 (restated) *(1) A sentence in GF(=) is invariant under disjoint unions iff it is equivalent to a sentence in uGF(=).*

(2) A sentence in GF is invariant under disjoint unions iff it is equivalent to a sentence in uGF.

PROOF. It remains to prove that every sentence in GF(=) is equivalent to a Boolean combination of sentences in uGF(=). Assume a sentence φ in GF(=) is given. First, we may assume that all subformulas $\exists y(x = y \wedge \psi(x,y))$ of φ are rewritten as $\neg\forall y(x = y \rightarrow \overline{\psi}(x,y))$. Then let $\psi = \forall y(x = y \rightarrow \psi'(x,y))$ be a subformula of φ with free variable x. It follows that $\psi \equiv \psi'(x,x)$. As $\psi'(x,x)$ is a GF(=) formula, any occurrence of ψ in φ can be replaced by $\psi'(x,x)$. Let φ' be the result of substituting all subformulas of the form of ψ in φ with the corresponding equivalent GF(=) formula. It follows that $\varphi \equiv \varphi'$ (note that $\forall x\forall y(x = y \rightarrow \psi'(x,y))$ can be rewritten as $\forall x(x = x \rightarrow \forall y(x = y \rightarrow \psi'(x,y)))$).

A sentence ψ in φ is a *simple sentence* if there is no strict subformula ψ' of ψ which is a sentence. For the second step, let $\varphi_0 = \varphi'$, and let

$$\varphi_i = (\varphi_{i-1}[\psi/\top] \wedge \psi) \vee (\varphi_{i-i}[\psi/\bot] \wedge \neg\psi)$$

for some simple sentence ψ of φ_{i-1} not already substituted in any step $j < i-1$. As at each step any simple sentence is a GF(=) sentence with guard $R(\vec{x})$ or $x = x$ and the guarded formula is an openGF formula, then any simple sentence is either the negation of a uGF(=) sentence or a uGF(=) sentence. Let φ_n be the resulting sentence. It follows that $\varphi \equiv \varphi_n$, and φ_n is a Boolean combination of uGF(=) sentences. □

For the proof of Lemma 1 we require the notion of guarded bisimulations that characterizes the expressive power of GF(=) [30]. To apply guarded bisimulations to characterize the fragment uGF(=) of GF(=) we modify the notion slightly by demanding that in the back-and-forth condition only overlapping guarded sets are used. To cover uGC$_2$(=) we also consider guarded bisimulations preserving the number of guarded sets of the same type that contain a fixed singleton set. Assume $\mathfrak{A}$ and $\mathfrak{B}$ are interpretations. A set I of partial isomorphisms $p : \vec{a} \mapsto \vec{b}$ between guarded tuples $\vec{a}$ and $\vec{b}$ in $\mathfrak{A}$ and $\mathfrak{B}$, respectively, is called a *connected guarded bisimulations* if the following holds for all $p : \vec{a} \mapsto \vec{b} \in I$:

- for every guarded tuple $\vec{a}'$ in $\mathfrak{A}$ which has a non-empty intersection with $\vec{a}$ there exists a guarded tuple $\vec{b}'$ in $\mathfrak{B}$ and $p' : \vec{a}' \mapsto \vec{b}' \in I$ that coincides with p on the intersection of $\vec{a}$ and $\vec{a}'$.
- for every guarded tuple $\vec{b}'$ in $\mathfrak{B}$ which has a non-empty intersection with $\vec{b}$ there exists a guarded tuple $\vec{a}'$ in $\mathfrak{A}$ and $p' : \vec{a}' \mapsto \vec{b}' \in I$ that coincides with p on the intersection of $\vec{b}$ and $\vec{b}'$.

We say that $(\mathfrak{A}, \vec{a})$ and $(\mathfrak{B}, \vec{b})$ are *connected guarded bisimilar* if there exists a connected guarded bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$ containing $\vec{a} \mapsto \vec{b}$.

Theorem 15 *If $(\mathfrak{A}, \vec{a})$ and $(\mathfrak{B}, \vec{b})$ are connected guarded bisimilar and $\varphi(\vec{x})$ is a formula in openGF, then $\mathfrak{A} \models \varphi(\vec{a})$ iff $\mathfrak{B} \models \varphi(\vec{b})$.*

For uGC$_2$(=) the guarded sets have cardinality at most two. To preserve counting we require the following modified version of the definition above. A set I of partial isomorphisms $p : \vec{a} \mapsto \vec{b}$ between guarded tuples $\vec{a} = (a_1, a_2)$ and $\vec{b} = (b_1, b_2)$ in $\mathfrak{A}$ and $\mathfrak{B}$, respectively, is called a *counting connected guarded bisimulations* if the following holds for all $p : (a_1, a_2) \mapsto (b_1, b_2) \in I$:

- for every finite number of distinct guarded tuples (a_1, a_2') in $\mathfrak{A}$ there exist (at least) the same number of guarded tuples (b_1, b_2') in $\mathfrak{B}$ and $p' : (a_1, a_2') \mapsto (b_1, b_2') \in I$.
- for every finite number of distinct guarded tuples (a_1', a_2) in $\mathfrak{A}$ there exist (at least) the same number of guarded tuples (b_1', b_2) in $\mathfrak{B}$ and $p' : (a_1', a_2) \mapsto (b_1', b_2) \in I$.
- for every finite number of distinct guarded tuples (b_1, b_2) in $\mathfrak{B}$ there exist (at least) the same number of guarded tuples (a_1, a_2') in $\mathfrak{A}$ and $p' : (a_1, a_2') \mapsto (b_1, b_2') \in I$.
- for every finite number of distinct guarded tuples (b_1', b_2) in $\mathfrak{B}$ there exist (at least) the same number of guarded tuples (a_1', a_2) in $\mathfrak{A}$ and $p' : (a_1', a_2) \mapsto (b_1', b_2) \in I$.

We say that $(\mathfrak{A}, \vec{a})$ and $(\mathfrak{B}, \vec{b})$ are *counting connected guarded bisimilar* if there exists a counting connected guarded bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$ containing $\vec{a} \mapsto \vec{b}$.

Theorem 16 *If $(\mathfrak{A}, \vec{a})$ and $(\mathfrak{B}, \vec{b})$ are counting connected guarded bisimilar and $\varphi(\vec{x})$ is a formula in openGC$_2$, then $\mathfrak{A} \models \varphi(\vec{a})$ iff $\mathfrak{B} \models \varphi(\vec{b})$.*

Lemma 1 (restated) *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 $\mathcal{O}$ such that there exists a homomorphism h from $\mathfrak{B}$ to $\mathfrak{A}$ that preserves dom$(\mathfrak{D})$.*

PROOF. Assume first a uGF(=) ontology $\mathcal{O}$, an instance $\mathfrak{D}$, and a model $\mathfrak{A}$ of $\mathcal{O}$ and $\mathfrak{D}$ are given. Let G be a maximal guarded set in $\mathfrak{D}$. We unfold $\mathfrak{A}$ at G into a cg-tree decomposable interpretation $\mathfrak{B}_G$ as follows: let $T(G)$ be the undirected tree with nodes $t = G_0G_1\cdots G_n$, where G_i, $0 \leq i \leq n$, are maximal guarded sets in $\mathfrak{A}$, $G_0 = G$, $G_1 \not\subseteq \text{dom}(\mathfrak{D})$, and (a) $G_i \neq G_{i+1}$, (b) $G_i \cap G_{i+1} \neq \emptyset$, and (c) $G_{i-1} \neq G_{i+1}$. Let $(t, t') \in E$ if $t' = tF$ for some maximal guarded F. Set $\text{tail}(G_0\cdots G_n) = G_n$. Take an infinite supply of copies of any $d \in \text{dom}(\mathfrak{A})$. We assume all copies are in Δ_D. We set $e^\uparrow = d$ if e is a copy of d. Define instances $\text{bag}(t)$ for $t \in T(G)$ inductively as follows. $\text{bag}(G)$ is an instance whose domain is a set of copies of elements $d \in G$ such that the mapping $e \mapsto e^\uparrow$ is an isomorphism from $\text{bag}(G)$ onto $\mathfrak{A}_{|G}$. Assume $\text{bag}(t)$ has been defined, $\text{tail}(t) = F$, and $\text{bag}(t')$ has not yet been defined for $t' = tF' \in T(G)$. Then take for any $d \in F' \setminus F$ a fresh copy d' of d and define $\text{bag}(t')$ with domain $\{d' \mid d \in F' \setminus F\} \cup \{e \in \text{bag}(t) \mid e^\uparrow \in F' \cap F)\}$ such that the mapping $e \mapsto e^\uparrow$ is an isomorphism from $\text{bag}(t')$ onto $\mathfrak{A}_{|F'}$.

Let $\mathfrak{B}_G = \bigcup_{t\in T(G)} \text{bag}(t)$. Now hook $\mathfrak{B}_G$ to $\mathfrak{D}$ at G by identifying $\text{dom}(\text{bag}(G))$ with G using the isomorphism $e \mapsto e^\uparrow$ and let $\mathfrak{B}$ be the union of all $\mathfrak{B}_G$ with G a maximal guarded set in $\mathfrak{D}$. It is not difficult to prove the following properties of $\mathfrak{B}$:

1. $(T(G), E, \text{bag})$ is a cg-tree decomposition of $\mathfrak{B}_G$, for every maximal guarded set G in $\mathfrak{D}$;
2. The mapping $h : e \mapsto e^\uparrow$ is a homomorphism from $\mathfrak{B}$ to $\mathfrak{A}$ which preserves $\text{dom}(\mathfrak{D})$ and is an isomorphism on each $\text{bag}(t)$ with $t \in T(G)$ for some maximal guarded set G in $\mathfrak{D}$.
3. For any maximal guarded set $G = \{a_1, \ldots, a_k\}$ in $\mathfrak{D}$, there is a connected guarded bisimulation between $(\mathfrak{B}, (a_1, \ldots, a_k))$ and $(\mathfrak{A}, (a_1, \ldots, a_k))$.

It follows from Theorem 15 that $\mathfrak{B}$ is a model of $\mathfrak{D}$ and $\mathcal{O}$ and thus as required.

Assume now that $\mathcal{O}$ is a uGC$_2(=)$ ontology, that $\mathfrak{D}$ is an instance, and that $\mathfrak{A}$ is a model of $\mathcal{O}$ and $\mathfrak{D}$. We can assume that $\mathfrak{D}$ and $\mathfrak{A}$ use unary and binary relation symbols only. We have to preserve the number of guarded sets of a given type intersecting with a singleton and therefore the unfolding is slightly different from the case without guarded counting quantifiers. Let $c \in \mathrm{dom}(\mathfrak{D})$. We unfold $\mathfrak{A}$ at c into a cg-tree decomposable $\mathfrak{B}_{\{c\}}$. Let $T(c)$ be the undirected tree with nodes $t = \{c\}G_1 \cdots G_n$, where G_i, $1 \leq i \leq n$, are maximal guarded sets in $\mathfrak{A}$, $c \in G_1$ and $G_1 \not\subseteq \mathrm{dom}(\mathfrak{D})$, and (a) $G_i \neq G_{i+1}$, (b) $G_i \cap G_{i+1} \neq \emptyset$, and (c) $G_{i-1} \cap G_i \neq G_{i+1} \cap G_i$. Let $(t, t') \in E$ if $t' = tF$ for some maximal guarded F in $\mathfrak{D}$. Observe that Condition (c) is stronger than the corresponding Condition (c) in the construction of forest models for uGF ontologies. The new condition ensures that we do not introduce more successors of the same type when we unfold an interpretation. Take an infinite supply of copies of any $d \in \mathrm{dom}(\mathfrak{A})$. We set, as before, $e^\uparrow = d$ if e is a copy of d. Define instances $\mathrm{bag}(t)$ for $t \in T(c)$ inductively as follows. Let c' be a copy of c and set $\mathrm{bag}(\{c\}) = \{P(c') \mid P(c) \in \mathfrak{A}\}$. Assume $\mathrm{bag}(t)$ has been defined, $\mathrm{tail}(t) = F$, and $\mathrm{bag}(t')$ has not yet been defined for $t' = tF' \in T(c)$. Take for the unique $d \in F' \setminus F$ a fresh copy d' of d and define $\mathrm{bag}(t')$ with domain $\{d'\} \cup \{e \in \mathrm{bag}(t) \mid e^\uparrow \in F' \cap F)\}$ such that the mapping $e \mapsto e^\uparrow$ is an isomorphism from $\mathrm{bag}(t')$ onto $\mathfrak{A}_{|F'}$. Let $\mathfrak{B}_{\{c\}} = \bigcup_{t \in T(c)} \mathrm{bag}(t)$. Now hook $\mathfrak{B}_{\{c\}}$ to $\mathfrak{D}$ at $\{c\}$ by identifying c' with c and let

$$\mathfrak{B} = \{R(\vec{a}) \in \mathfrak{A} \mid \vec{a} \subseteq \mathrm{dom}(\mathfrak{D})\} \cup \bigcup_{c \in \mathrm{dom}(\mathfrak{D})} \mathfrak{B}_{\{c\}}.$$

We can, of course, present the interpretation $\mathfrak{B}$ as an interpretation obtained from hooking interpretations $\mathfrak{B}_G$ with G maximal guarded sets to $\mathfrak{D}$ by choosing for each c a single maximal guarded set $f(c)$ with $c \in f(c)$ and setting for $G = \{c_1, c_2\}$,

$$\mathfrak{B}_G = \{R(\vec{a}) \in \mathfrak{A} \mid \vec{a} \subseteq G\} \cup \bigcup_{f(c)=G} \mathfrak{B}_{\{c\}}$$

It is not difficult to prove the following properties of $\mathfrak{B}$:

1. $(T(c), E, \mathrm{bag})$ is a guarded tree decomposition of $\mathfrak{B}_{\{c\}}$, for every $c \in \mathrm{dom}(\mathfrak{D})$;
2. The mapping $h : e \mapsto e^\uparrow$ is a homomorphism from $\mathfrak{B}$ to $\mathfrak{A}$ which preserves $\mathrm{dom}(\mathfrak{D})$ and is an isomorphism on each $\mathrm{bag}(t)$ with $t \in T(c)$ for some $c \in \mathrm{dom}(\mathfrak{D})$.
3. For any maximal guarded set $G = \{a_1, a_2\}$ in $\mathfrak{D}$, there is a counting connected guarded bisimulation between $(\mathfrak{B}, (a_1, a_2))$ and $(\mathfrak{A}, (a_1, a_2))$.

It follows from Theorem 16 that $\mathfrak{B}$ is a model of $\mathcal{O}$ and $\mathfrak{D}$ and thus as required. □

D. PROOFS FOR SECTION 3

Theorem 2 (restated) *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 equivalence (2) $\Leftrightarrow$ (3) is straightforward. We show (1) $\Rightarrow$ (2). Assume $\mathcal{O}$ is rAQ-materializable for $\mathcal{M}$ and assume the instance $\mathfrak{D} \in \mathcal{M}$ is consistent w.r.t. $\mathcal{O}$. Let $\mathfrak{A}$ be a rAQ-materialization of $\mathcal{O}$ and $\mathfrak{D}$. Consider a forest model $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}$ such that there is a homomorphism from $\mathfrak{B}$ to $\mathfrak{A}$ preserving $\mathrm{dom}(\mathfrak{D})$ (Lemma 1). Recall that $\mathfrak{B}$ is obtained from $\mathfrak{D}$ by hooking cg-tree decomposable $\mathfrak{B}_G$ to any maximal guarded set G in $\mathfrak{D}$. We show that $\mathfrak{B}$ is a CQ-materialization of $\mathcal{O}$ and $\mathfrak{D}$. To this end it is sufficient to prove that for any finite subinterpretation $\mathfrak{B}'$ of $\mathfrak{B}$ and any model $\mathfrak{A}'$ of $\mathcal{O}$ and $\mathfrak{D}$ there exists a homomorphism h from $\mathfrak{B}'$ to $\mathfrak{A}'$ that preserves $\mathrm{dom}(\mathfrak{D}) \cap \mathrm{dom}(\mathfrak{B}')$. We may assume that for any maximal guarded set G in $\mathfrak{D}$, if $\mathrm{dom}(\mathfrak{B}') \cap G \neq \emptyset$, then $G \subseteq \mathrm{dom}(\mathfrak{B}')$ and that $\mathfrak{B}' \cap \mathfrak{B}_G$ is connected if nonempty. But then we can regard every $\mathfrak{B}' \cap \mathfrak{B}_G$ as an rAQ q_G with answer variables G. From $\mathfrak{B} \models q_G(\vec{b})$ for $\vec{b}$ a suitable tuple containing all $b \in G$ we obtain $\mathfrak{A} \models q_G(\vec{b})$ since $\mathfrak{B}$ is a rAQ-materialization of $\mathfrak{D}$ and $\mathcal{O}$. Let h_G be the homomorphism that witnesses $\mathfrak{B} \models q_G(\vec{b})$. Then h_G is a homomorphism from $\mathfrak{B}' \cap \mathfrak{B}_G$ to $\mathfrak{A}$ that preserves G. The union h of all h_G, G a maximal guarded set in $\mathfrak{D}$, is the desired homomorphism from $\mathfrak{B}'$ to $\mathfrak{A}'$ preserving $\mathrm{dom}(\mathfrak{D}) \cap \mathrm{dom}(\mathfrak{B}')$. □

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

PROOF. The direction "$\Leftarrow$" is straightforward. Conversely, assume that $\mathcal{O}$ is materializable. Let $\mathfrak{D}$ be an instance that is consistent w.r.t. $\mathcal{O}$. By Lemma 1 there exists a forest model $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}$ that is a CQ-materialization of $\mathcal{O}$ and $\mathfrak{D}$. Using a straightforward selective filtration procedure one can show that there exists $\mathfrak{B}' \subseteq \mathfrak{B}$ such that $\mathfrak{B}'$ is a model of $\mathcal{O}$ and $\mathfrak{D}$ and for any guarded set G the number of guarded sets G' with $G \cap G' \neq \emptyset$ is finite. We show that for any model $\mathfrak{A}$ of $\mathcal{O}$ and $\mathfrak{D}$ there is a homomorphism from $\mathfrak{B}'$ into $\mathfrak{A}$ that preserves $\mathsf{dom}(\mathfrak{D})$. Let $\mathfrak{A}$ be a model of $\mathcal{O}$ and $\mathfrak{D}$. Again we may assume that $\mathfrak{A}$ is a forest model such that for any guarded set G the number of guarded sets G' with $G \cap G' \neq \emptyset$ is finite. Now, for any finite subset F of $\mathrm{dom}(\mathfrak{B}')$ there is a homomorphism h_F from $\mathfrak{B}'_{|F}$ to $\mathfrak{A}$ preserving $\mathrm{dom}(\mathfrak{D}) \cap F$. Let F_m be the set of all $d \in \mathrm{dom}(\mathfrak{B}')$ such that there is a sequence of at most m guarded sets $G_0, \ldots, G_m$ with $G_i \cap G_{i+1} \neq \emptyset$ for $i < m$, $G_0 \cap \mathrm{dom}(\mathfrak{D}) \neq \emptyset$, and $d \in G_m$. Then each F_m is finite and $\bigcup_{m \geq 0} F_m = \mathrm{dom}(\mathfrak{B}')$. Now we can construct the homomorphism h from $\mathfrak{B}'$ to $\mathfrak{A}$ as the limit of homomorphisms $h_{F_{n_m}}$ from $\mathfrak{B}'_{|F_{n_m}}$ to $\mathfrak{A}$ preserving $\mathrm{dom}(\mathfrak{D})$ for an infinite sequence $n_0 < n_1 \cdots$ using a standard pigeonhole argument.

We construct an ontology $\mathcal{O}$ that expresses that every element of a unary relation C is in the centre of a 'partial wheel' represented by a ternary relation W. The wheel can be generated by turning right or left. The two resulting models are homomorphically incomparable but cannot be distinguished by answers to CQs. Thus, neither of the two models is hom-universal but one can ensure that $\mathcal{O}$ is materializable. As a first attempt to construct $\mathcal{O}$ we take unary relations L (turn left) and R (turn right), the following sentence that says that one has to choose between L and R when generating the wheel with cente C:

$$\forall x \big(C(x) \to ((L(x) \vee R(x)) \wedge \exists y_1, y_2 W(x, y_1, y_2))\big)$$

and the following sentences that generates the wheel accordingly:

$$\forall x, y, z \big(W(x, y, z) \to (L(x) \to \exists z' W(x, z, z'))\big)$$

$$\forall x, y, z \big(W(x, y, z) \to (R(x) \to \exists y' W(x, y', y))\big)$$

Clearly this ontology does not admit hom-universal models. Note, however, that is also not materializable since for the instance $\mathfrak{D} = \{C(a)\}$ we have $\mathcal{O}, \mathfrak{D} \models L(a) \vee R(a)$ but we neither have $\mathcal{O}, \mathfrak{D} \models L(a)$ nor $\mathcal{O}, \mathfrak{D} \models R(a)$. The first step to ensure materializability is to replace $L(x)$ by $\exists y(\text{gen}(x, y) \wedge \neg L(x))$ and $R(x)$ by $\exists y(\text{gen}(x, y) \wedge \neg R(x))$ in the sentences above and also add $\forall x \exists y(\text{gen}(x, y) \wedge L(x))$ and $\forall x \exists y(\text{gen}(x, y) \wedge R(x))$ to $\mathcal{O}$. Then a CQ 'cannot detect' whether one satisfies the disjunct $\exists y(\text{gen}(x, y) \wedge \neg R(x))$ or the disjunct $\exists y(\text{gen}(x, y) \wedge \neg L(x))$ at a given node x in C. Note that the resulting ontology is still not materializable: if $W(a, b, c) \in \mathfrak{D}$ then CQs can detect whether one introduces a c' with $W(a, c, c') \in \mathfrak{A}$ or a b' with $W(a, b', b) \in \mathfrak{A}$. To deal with this problem we ensure that a wheel has to be generated from $W(a, b, c)$ only if b, c are not in the instance $\mathfrak{D}$. In detail, we construct $\mathcal{O}$ as follows. We use unary relations A, L, and R and binary relations aux and gen. For a binary relation S and unary relation B we abbreviate

$$\begin{aligned} \forall S.B &:= \forall y(S(x, y) \to B(y)) \\ \exists S.\neg B(x) &:= \exists y(S(x, y) \wedge \neg B(y)) \end{aligned}$$

Now let $\mathcal{O}$ state that every node has aux-successors in A, and gen-successors in L and R:

$$\forall x \exists y(\text{aux}(x, y) \wedge A(y))$$

and

$$\forall x \exists y(\text{gen}(x, y) \wedge L(y)) \quad \forall x \exists y(\text{gen}(x, y) \wedge R(y))$$

We abbreviate

$$W'(x, y, z) := W(x, y, z) \wedge \forall \text{aux}.A(y) \wedge \forall \text{aux}.A(z)$$

Next we introduce the disjunction that determines whether one generates the wheel by turning left or right, as indicated above:

$$\forall x \big(C(x) \to (\exists \text{gen}.\neg L(x) \vee \exists \text{gen}.\neg R(x))\big)$$

The following axiom starts the generation of the wheel. Observe that we use $W'(x, y_1, y_2)$ rather than $W(x, y_1, y_2)$ to ensure that new individuals are created for y_1 and y_2:

$$\forall x \big(C(x) \to \exists y_1, y_2 W'(x, y_1, y_2)\big)$$

Finally, we turn either left or right:

$$\forall x, y, z \big(W'(x, y, z) \to (\exists \text{gen}.\neg L(x) \to \exists z' W'(x, z, z'))\big)$$

$$\forall x, y, z \big(W'(x, y, z) \to (\exists \text{gen}.\neg R(x) \to \exists y' W'(x, y', y))\big)$$

This finishes the definition of $\mathcal{O}$. $\mathcal{O}$ is a uGF(2) ontology. To show that it is materializable observe that for any instance $\mathfrak{D}$ we can ensure that in a model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$ for all $a \in \text{dom}(\mathfrak{D})$ there exists b with $\text{aux}(a, b) \in \mathfrak{A}$ and $A(b) \notin \mathfrak{A}$. Thus, the wheel generation sentences apply only to $W(a, b, c)$ with $b, c \notin \text{dom}(\mathfrak{A})$. It is now easy to prove that CQ evaluation w.r.t. $\mathcal{O}$ is in PTIME. On the other hand, for $\mathfrak{D} = \{C(a)\}$ there does not exist a model $\mathfrak{B}$ of $\mathcal{O}$ and $\mathfrak{D}$ such that there is a homomorphism from $\mathfrak{B}$ into every model of $\mathcal{O}$ and $\mathfrak{D}$ that preserves $\mathsf{dom}(\mathfrak{D})$. □

In the proof of Lemma 3 (and related proofs below), it will be more convenient to work with a certain disjunction property instead of with materializability. We now introduce this property and show the equivalence of the two notions. Let $\mathcal{Q}$ be a class of non-Boolean connected CQs. An ontology $\mathcal{O}$ has the *$\mathcal{Q}$-disjunction property* if for all instances $\mathfrak{D}$, queries $q_1(\vec{x}_1), \ldots, q_n(\vec{x}_n) \in \mathcal{Q}$ and $\vec{d}_1, \ldots, \vec{d}_n$ in $\mathfrak{D}$ with $\vec{d}_i$ of the same length as $\vec{x}_i$: if $\mathcal{O}, \mathfrak{D} \models q_1(\vec{d}_1) \vee \ldots \vee q_n(\vec{d}_n)$, then there exists $i \leq n$ such that $\mathcal{O}, \mathfrak{D} \models q_i(\vec{d}_i)$.

Theorem 17 *Let $\mathcal{Q}$ be a class CQs and $\mathcal{O}$ an FO(=)-ontology. Then $\mathcal{O}$ is $\mathcal{Q}$-materializable iff $\mathcal{O}$ has the $\mathcal{Q}$-disjunction property.*

PROOF. For the nontrivial "⇐" direction, let $\mathfrak{D}$ be an intance that is consistent w.r.t. $\mathcal{O}$ and such that there is no $\mathcal{Q}$-materialization of $\mathcal{O}$ and $\mathfrak{D}$. Then the set of FO-sentences $\mathcal{O} \cup \mathfrak{D} \cup \Gamma$ is not satisfiable, where

$$\Gamma = \{\neg q(\vec{d}) \mid \mathcal{O}, \mathfrak{D} \not\models q(\vec{d}), \vec{d} \subseteq \text{dom}(\mathfrak{D}), q \in \mathcal{Q}\}$$

In fact, any satisfying interpretation would be a $\mathcal{Q}$-materialization of $\mathcal{O}$. By compactness, there is a finite subset Γ' of Γ such that $\mathcal{O} \cup \mathfrak{D} \cup \Gamma'$ is not satisfiable, i.e.

$$\mathcal{O}, \mathfrak{D} \models \bigvee_{\neg q(\vec{d}) \in \Gamma'} q(\vec{d})$$

By definition of Γ', $\mathcal{O}, \mathfrak{D} \not\models q(\vec{d})$ for all $q(\vec{d}) \in \Gamma'$. Thus, $\mathcal{O}$ lacks the $\mathcal{Q}$-disjunction property. □

Theorem 3 (restated) *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.*

PROOF SKETCH. It was proved in [42] that if an $\mathcal{ALC}$ ontology $\mathcal{O}$ is not ELIQ-materializable, then ELIQ-evaluation w.r.t. $\mathcal{O}$ is CONP-hard, where an ELIQ is a rAQ $q(\vec{x})$ such that the associated instance $\mathfrak{D}_{q(\vec{x})}$ viewed as an undirected graph is a tree (instead of cg-tree decomposable) with a single answer variable at the root.[2] The proof is by reduction from 2+2-SAT, the variant of propositional satisfiability where the input is a set of clauses of the form $(p_1 \vee p_2 \vee \neg n_1 \vee \neg n_2)$, each p_1, p_2, n_1, n_2 a propositional letter or a truth constant [50]. The proof of Theorem 3 can be obtained from the proof in [42] by minor modifications, which we sketch in the following.

The proof crucially exploits that if $\mathcal{O}$ is not rAQ-materializable, then by Theorem 17 it does not have the rAQ-disjunction property. In fact, we take an instance $\mathfrak{D}$, rAQs (resp. ELIQs) $q_1(\vec{x}_1), \ldots, q_n(\vec{x}_n) \in \mathcal{Q}$, and elements $\vec{d}_1, \ldots, \vec{d}_n$ of $\mathfrak{D}$ that witness failure of the disjunction property, copy them an appropriate number of times, and use the resulting set of gadgets to choose a truth value for the variables in the input 2+2-SAT formula. The fact that $\mathcal{O}$ is invariant under disjoint unions ensures that the choice of truth values for different variables is independent. A main difference between ELIQs and rAQs is that rAQs can have more than one answer variable. A straightforward way to handle this is to replace certain binary relations from the reduction in [42] with relations of higher arity (these are 'fresh' relations introduced in the reduction, that is, they do not occur in $\mathcal{O}$). To deal with a rAQ of arity k, one would use a $k + 1$-ary relation. However, with a tiny bit of extra effort (and if desired), one can replace these relations with k binary relations. □

We now turn to the proof of Theorem 4.

Theorem 4 (restated) *Let $\mathcal{O}$ be a uGF(=) or uGC$_2$(=) ontology. The following are equivalent:*

1. *UCQ-evaluation w.r.t. $\mathcal{O}$ is in* PTIME *iff CQ-evaluation w.r.t. $\mathcal{O}$ is in* PTIME *iff rAQ-evaluation w.r.t. $\mathcal{O}$ is in* PTIME*.*
2. *UCQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable iff CQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable iff rAQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable. If $\mathcal{O}$ is a uGF ontology, then the same is true for Datalog-rewritability.*

[2] In the context of $\mathcal{ALC}$, relation symbols are at most binary and thus it should be clear what 'tree' means.

3. *UCQ-evaluation w.r.t. $\mathcal{O}$ is* coNP*-hard iff CQ-evaluation w.r.t. $\mathcal{O}$ is* coNP*-hard iff rAQ-evaluation w.r.t. $\mathcal{O}$ is* coNP*-hard.*

The "only if" directions of the first two parts of Theorem 4 and the "if" direction of the third part are trivial. For the non-trivial "if" directions of part 1 and 2 we decompose UCQs into finite disjunctions 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. The same decomposition of UCQs is also used in the "only if" direction of part 3. The following definition formally introduces decompositions of UCQs.

Definition 5 Let $\mathcal{O}$ be an ontology and let $q(\vec{x})$ be a UCQ. A *decomposition* of $q(\vec{x})$ w.r.t. $\mathcal{O}$ is a finite set $\mathcal{Q}$ of pairs $(\phi(\vec{y}), \mathcal{C})$, where

- $\phi(\vec{y})$ is a conjunction of atoms (possibly equality atoms) such that $\vec{y}$ contains all variables in $\vec{x}$;
- $\mathcal{C}$ is a finite set of CQs $q(\vec{z})$, where $\mathfrak{D}_q$ is cg-tree decomposable;

such that the following are equivalent for each instance $\mathfrak{D}$ and tuple $\vec{a} \in \mathsf{dom}(\mathfrak{D})^{|\vec{x}|}$:

1. $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$.
2. For each model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$ there exists a pair $(\phi(\vec{y}), \mathcal{C})$ in $\mathcal{Q}$ and an assignment π of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$ such that
   - $\pi(\vec{x}) = \vec{a}$,
   - $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and
   - $\mathfrak{A} \models q'(\pi(\vec{z}))$ for each $q'(\vec{z})$ in $\mathcal{C}$.

The decomposition is *strong* if for each pair $(\phi(\vec{y}), \mathcal{C})$, the set $\mathcal{C}$ consists of rAQs.

We aim to prove that for every uGF(=) or uGC$_2$(=) ontology $\mathcal{O}$ and each UCQ $q(\vec{x})$ there exists a strong decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$. As an intermediate step, we prove:

Lemma 8 *For each uGF(=) or uGC$_2$(=) ontology $\mathcal{O}$ and each UCQ $q(\vec{x})$ there is a decomposition $\mathcal{Q}$ of $q(\vec{x})$ w.r.t. $\mathcal{O}$. Moreover, if $\mathcal{O}$ is an uGC$_2$(=) ontology, then each rAQ that occurs in a pair of $\mathcal{Q}$ has a single free variable.*

PROOF. The construction is similar to that of squid decompositions in [12]. We first construct $\mathcal{Q}$ and then show that it is a decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$. To simplify the presentation, we will assume that each element that occurs in $\mathfrak{D}_{q'}$ for a CQ q' is identified with the variable it represents. Consider the set $\mathcal{I}$ of all tuples $(q', f, V, H, (T_i)_{i=1}^k)$, where

- q' is a disjunct of q;
- $f\colon \mathsf{dom}(\mathfrak{D}_{q'}) \to \mathsf{dom}(\mathfrak{D}_{q'})$ is an idempotent mapping such that for each answer variable x of q' we have that $f(x)$ is an answer variable;
- $V \subseteq f(\mathsf{dom}(\mathfrak{D}_{q'}))$ and V contains all variables in $f(\vec{x})$;
- $H, T_1, \ldots, T_n$ form a partition of $f(\mathfrak{D}_{q'})$, and each atom in H only contains variables in V;
- each T_i is cg-tree decomposable;
- if $\mathcal{O}$ is a uGC$_2$(=) ontology, then each $\mathsf{dom}(T_i)$ contains at most one variable from V;
- k is at most the number of atoms in q'.

Clearly, $\mathcal{I}$ is finite. Each tuple $I \in \mathcal{I}$ contributes exactly one pair p_I to $\mathcal{Q}$, which we describe next.

Let $I = (q', f, V, H, (T_i)_{i=1}^k) \in \mathcal{I}$. Define $\phi(\vec{y})$ to be the conjunction of all atoms of H and all equality atoms $x = f(x)$ for each variable x in $\vec{x}$ with $f(x) \neq x$. Furthermore, for each $i \in \{1, \ldots, k\}$, let $q_i(\vec{z}_i)$ be the CQ whose atoms are the atoms in T_i, and whose tuple $\vec{z}_i$ of answer variables consists of all variables in $\mathsf{dom}(T_i)$ that also occur in V. Let

$$p_I = (\phi(\vec{y}), \{q_i(\vec{z}_i) \mid 1 \leq i \leq k\}).$$

We show that $\mathcal{Q} = \{p_I \mid I \in \mathcal{I}\}$ is a decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$.

First note that each pair in $\mathcal{Q}$ has the form $(\phi(\vec{y}), \mathcal{C})$, where $\phi(\vec{y})$ is a conjunction of atomic formulas that contains all variables of $\vec{x}$, and $\mathcal{C}$ is a finite set of CQs $q'(\vec{z})$ whose instance $\mathfrak{D}_{q'}$ is cg-tree decomposable. Furthermore, if $\mathcal{O}$ is formulated in uGC$_2$(=), then each CQ in $\mathcal{C}$ has at most one answer variable.

Next we establish the equivalence between conditions 1 and 2 from Definition 5. For the direction from 2 to 1, suppose that for each model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$ there exists a pair $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$ and an assignment π of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$ such that $\pi(\vec{x}) = \vec{a}$, $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and $\mathfrak{A} \models q'(\pi(\vec{z}))$ for each $q'(\vec{z}) \in \mathcal{C}$. By construction of $(\phi(\vec{y}), \mathcal{C})$ this implies $\mathcal{O}, \mathfrak{D} \models q$.

For the direction from 2 to 1, suppose that $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$, and consider a model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$. By Lemma 1, there is a forest-model $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}$ and a homomorphism h from $\mathfrak{B}$ to $\mathfrak{A}$ that preserves $\mathsf{dom}(\mathfrak{D})$. Let

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

where $\mathcal{G}$ is the set of all maximal guarded subsets of $\mathfrak{D}$, and for each $\mathfrak{B}_G$ there is a cg-tree decomposition of $\mathfrak{B}_G$ whose root is labeled with a bag with domain G. Now, $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ implies $\mathfrak{B} \models q(\vec{a})$, so there is a disjunct $q'(\vec{x})$ of $q(\vec{x})$ and a homomorphism π from $\mathfrak{D}_{q'}$ to $\mathfrak{B}$ with $\pi(\vec{x}) = \vec{a}$. Note that π is the composition of

- an idempotent mapping $f\colon \mathsf{dom}(\mathfrak{D}_{q'}) \to \mathsf{dom}(\mathfrak{D}_{q'})$ such that $f(\vec{x})$ contains only variables from $\vec{x}$, and
- an injective mapping g from $f(\mathsf{dom}(\mathfrak{D}_{q'}))$ to $\mathfrak{B}$.

Let

- $V := \{f(x) \in \mathsf{dom}(\mathfrak{D}_{q'}) \mid g(f(x)) \in \mathsf{dom}(\mathfrak{D})\}$;
- $H := \{f(\alpha) \mid \alpha \in \mathfrak{D}_{q'},\ g(f(\alpha)) \in \mathfrak{D} \setminus \bigcup_{G \in \mathcal{G}} \mathfrak{B}_G\}$;
- $T_G := \{f(\alpha) \mid \alpha \in \mathfrak{D}_{q'},\ g(f(\alpha)) \in \mathfrak{B}_G\}$ for $G \in \mathcal{G}$.

Here, for each atom $\alpha = R(x_1, \ldots, x_k)$ and mapping h with domain $\{x_1, \ldots, x_k\}$ we use the notation $h(\alpha)$ to denote the atom $R(h(x_1), \ldots, h(x_k))$. Let $T_1, \ldots, T_k$ be an enumeration of all connected components of the sets T_G, $G \in \mathcal{G}$. Then each T_i is cg-tree decomposable, and k is bounded by the number of atoms in q'. Now,

$$I = (q'(\vec{x}), f, V, H, (T_i)_{i=1}^k) \in \mathcal{I},$$

and $p_I = (\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$. It is straightforward to verify that $\mathfrak{D} \models \phi(\pi(\vec{y}))$ and $\mathfrak{B} \models q'(\pi(\vec{z}))$ for each $q'(\vec{z}) \in \mathcal{C}$. Since h is a homomorphism from $\mathfrak{B}$ to $\mathfrak{A}$ that preserves $\mathsf{dom}(\mathfrak{D})$, we obtain that $\pi' := h \circ \pi$ is an assignment of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$ with $\pi'(\vec{x}) = \vec{a}$, $\mathfrak{D} \models \phi(\pi'(\vec{y}))$, and $\mathfrak{A} \models q'(\pi'(\vec{z}))$ for each $q'(\vec{z}) \in \mathcal{C}$.

Altogether, this completes the proof that $\mathcal{Q}$ is a decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$. □

Next we show that for uGF(=) or uGC$_2$(=) ontologies $\mathcal{O}$ we can transform any decomposition of a UCQ $q(\vec{x})$ w.r.t. $\mathcal{O}$ into a

strong decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$. This transformation is based on Lemma 9 below. Given a uGF(=) or uGC$_2$(=) ontology $\mathcal{O}$ and a decomposition $\mathcal{Q}$ of a UCQ w.r.t. $\mathcal{O}$, the lemma tells us that for each model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$, each pair $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$, each assignment π of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$, and each $q'(\vec{z}) \in \mathcal{C}$, it suffices to consider homomorphisms from $\mathfrak{D}_{q'}$ to $\mathfrak{A}$ whose image contains an element at a bounded distance from an element in $\mathsf{dom}(\mathfrak{D})$. Let us first make more precise what we mean by the distance between elements in an instance.

Definition 6 Let $\mathfrak{A}$ be an instance.

- The *Gaifman graph* of an instance $\mathfrak{A}$ is the undirected graph whose vertex set is $\mathsf{dom}(\mathfrak{A})$ and which has an edge between any two distinct $a, b \in \mathsf{dom}(\mathfrak{A})$ if a and b occur together in an atom of $\mathfrak{A}$.
- Given $a, b \in \mathsf{dom}(\mathfrak{A})$, the *distance* from a to b in $\mathfrak{A}$, denoted by $\mathsf{dist}_{\mathfrak{A}}(a, b)$, is the length of a shortest path from a to b in the Gaifman graph of $\mathfrak{A}$, or ∞ if there exists no such path.
- Given $A, B \subseteq \mathsf{dom}(\mathfrak{A})$, the *distance* from A to B in $\mathfrak{A}$ is defined as $\mathsf{dist}_{\mathfrak{A}}(A, B) := \min_{a \in A, b \in B} \mathsf{dist}_{\mathfrak{A}}(a, b)$.

Lemma 9 *Let $\mathcal{O}$ be a uGF(=) or uGC$_2$(=) ontology, and let $\mathcal{Q}$ be a decomposition of a UCQ $q_0(\vec{x})$ w.r.t. $\mathcal{O}$. Then there exists an integer $d \geq 0$ such that the following is true for all instances $\mathfrak{D}$ and all tuples $\vec{a} \in \mathsf{dom}(\mathfrak{D})^{|\vec{x}|}$.*

If $\mathcal{O}, \mathfrak{D} \models q_0(\vec{a})$, then for each model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$ there exists a pair $(\phi(\vec{y}), \mathcal{C})$ in $\mathcal{Q}$ and an assignment π of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$ such that

- *$\pi(\vec{x}) = \vec{a}$,*
- *$\mathfrak{D} \models \phi(\pi(\vec{y}))$, and*
- *for each $q(\vec{z}) \in \mathcal{C}$ there is a homomorphism h from $\mathfrak{D}_q$ to $\mathfrak{A}$ such that $h(\vec{z}) = \pi(\vec{z})$ and the distance from $\mathsf{dom}(h(\mathfrak{D}_q))$ to $\mathsf{dom}(\mathfrak{D})$ in $\mathfrak{A}$ is at most d.*

PROOF. For each model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$ and each positive integer d, let $\mathcal{Q}_d(\mathfrak{A})$ be the set of all triples $(\phi(\vec{y}), \mathcal{C}, \pi)$ such that $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$ and π is an assignment of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$ such that

- $\pi(\vec{x}) = \vec{a}$,
- $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and
- for each $q(\vec{z}) \in \mathcal{C}$ there is a homomorphism h from $\mathfrak{D}_q$ to $\mathfrak{A}$ such that $h(\vec{z}) = \pi(\vec{z})$ and the distance from $\mathsf{dom}(h(\mathfrak{D}_q))$ to $\mathsf{dom}(\mathfrak{D})$ in $\mathfrak{A}$ is at most d.

We show that there exists a constant $d \geq 0$ with the following property: if $\mathcal{O}, \mathfrak{D} \models q_0(\vec{a})$, then for each model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$ we have $\mathcal{Q}_d(\mathcal{A}) \neq \emptyset$.

The constant d depends on the number of C-types, which we define next. Let $\mathcal{Q}'$ be the set of all *Boolean* CQs that occur in the set $\mathcal{C}$ of some pair $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$. For the definition of C-types we assume that each $q \in \mathcal{Q}'$ is a GF sentence (if $\mathcal{O}$ is formulated in uGF(=)), or a GC$_2$ sentence (if $\mathcal{O}$ is formulated in uGC$_2$(=)). We can assume this w.l.o.g. because for each $q \in \mathcal{Q}'$ the instance $\mathfrak{D}_q$ has a cg-tree decomposition, which can be used to construct a GF or GC$_2$ sentence that is equivalent to q. Now, let $\mathsf{cl}(\mathcal{O}, q)$ be the smallest set satisfying:

- $\mathcal{O} \cup \mathcal{Q}' \subseteq \mathsf{cl}(\mathcal{O}, q)$;
- $\mathsf{cl}(\mathcal{O}, q)$ contains one atomic formula $R(\vec{x})$ for each relation symbol R that occurs in $\mathcal{O}$ or q, where $\vec{x}$ is a tuple of distinct variables;
- $\mathsf{cl}(\mathcal{O}, q)$ is closed under subformulas and single negation, where the subformulas of a formula $\phi = Q\vec{x}\,\psi$ with a quantifier Q are ϕ itself and ψ.

Given a set $C \subseteq \Delta_D$, a *C-type* is a set of formulas $\phi(\vec{c})$, where $\phi(\vec{x}) \in \mathsf{cl}(\mathcal{O}, q)$ and $\vec{c}$ is a tuple of constants in C of the appropriate length. Let w be the maximum arity of a relation symbol in $\mathcal{O}$ or q, and define τ to be the number of C-types for a set C of size $2w$. Note that τ depends only on $\mathcal{O}$ and q, and is bounded by $2^{|\mathsf{cl}(\mathcal{O},q)| \cdot (2w)^w}$. Fix

$$d := s + d^* \quad \text{with} \quad d^* := \tau^2 + 1,$$

where s is the maximum number of atoms in a non-Boolean CQ that occurs in $\mathcal{C}$ for some pair $(\phi(\vec{y}, \mathcal{C}) \in \mathcal{Q}$. This finishes the definition of the constant d.

Suppose now that $\mathcal{O}, \mathfrak{D} \models q_0(\vec{a})$. We aim to show that each model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$ satisfies $\mathcal{Q}_d(\mathcal{A}) \neq \emptyset$. Let $\mathfrak{A}$ be a model of $\mathfrak{D}$ and $\mathcal{O}$. By Lemma 1, there is a forest-model $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}$ and a homomorphism g from $\mathfrak{B}$ to $\mathfrak{A}$ that preserves $\mathsf{dom}(\mathfrak{D})$. We show that $\mathcal{Q}_d(\mathcal{B}) \neq \emptyset$. Since g preserves CQs and does not increase distances (i.e., for all $a, b \in \mathsf{dom}(\mathfrak{B})$ we have $\mathsf{dist}_{\mathfrak{A}}(g(a), g(b)) \leq \mathsf{dist}_{\mathfrak{B}}(a, b)$), this implies $\mathcal{Q}_d(\mathcal{A}) \neq \emptyset$, as desired.

For a contradiction, suppose that $\mathcal{Q}_d(\mathfrak{B}) = \emptyset$. We use $\mathfrak{B}$ to construct a model of $\mathfrak{D}$ and $\mathcal{O}$ that does not satisfy $q_0(\vec{a})$. This contradicts $\mathcal{O}, \mathfrak{D} \models q_0(\vec{a})$, and finishes the proof. The following is the core of the construction.

Claim. Let $\mathfrak{C}$ be a forest-model of $\mathfrak{D}$ and $\mathcal{O}$ and let $e \geq d$ be such that $\mathcal{Q}_e(\mathfrak{C}) = \emptyset$. Then there is a forest-model $\mathfrak{C}'$ of $\mathfrak{D}$ and $\mathcal{O}$ with $\mathcal{Q}_{e+1}(\mathfrak{C}') = \emptyset$, and $\mathfrak{C}'$ agrees with $\mathfrak{C}$ on all elements at distance at most $e - d^* + 1$ from $\mathsf{dom}(\mathfrak{D})$.

Proof. Recall the definition of the set $\mathcal{Q}'$ given at the beginning of the proof of Lemma 9. Let X be the set of all $q \in Q'$ such that for each homomorphism h from $\mathfrak{D}_q$ to $\mathfrak{C}$,

$$\mathsf{dist}_{\mathfrak{C}}(\mathsf{dom}(h(\mathfrak{D}_q)), \mathsf{dom}(\mathfrak{D})) \geq e + 1.$$

For each instance $\mathfrak{C}'$ and each integer e', let $H_{e'}(\mathfrak{C}')$ be the set of all pairs (q, h) such that $q \in X$ and h is a homomorphism from $\mathfrak{D}_q$ to $\mathfrak{C}'$ with

$$\mathsf{dist}_{\mathfrak{C}'}(\mathsf{dom}(h(\mathfrak{D}_q)), \mathsf{dom}(\mathfrak{D})) \leq e'.$$

Pick a forest-model $\mathfrak{C}'$ of $\mathfrak{D}$ and $\mathcal{O}$ such that

1. $H_e(\mathfrak{C}')$ is empty;
2. $H_{e+1}(\mathfrak{C}')$ is inclusion-minimal;
3. for each $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$, each assignment π with $\pi(\vec{x}) = \vec{a}$ and $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and each non-Boolean $q(\vec{z}) \in \mathcal{C}$ we have $\mathfrak{C} \models q(\pi(\vec{z}))$ iff $\mathfrak{C}' \models q(\pi(\vec{z}))$;
4. $\mathfrak{C}'$ agrees with $\mathfrak{C}$ on all elements at distance at most $e - d + 1$ from $\mathsf{dom}(\mathfrak{D})$.

Such a model exists since $\mathfrak{C}$ is a forest-model of $\mathfrak{D}$ and $\mathcal{O}$ that satisfies conditions 1 and 3. We show that $H_{e+1}(\mathfrak{C}') = \emptyset$. This implies $\mathcal{Q}_{e+1}(\mathfrak{C}') = \emptyset$ as follows. Suppose, for a contradiction, that $H_{e+1}(\mathfrak{C}') = \emptyset$ and $\mathcal{Q}_{e+1}(\mathfrak{C}') \neq \emptyset$. Then any $(\phi(\vec{y}), \mathcal{C}, \pi) \in \mathcal{Q}_{e+1}(\mathfrak{C}')$ satisfies $\pi(\vec{x}) = \vec{a}$, $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and $\mathfrak{C} \models q(\pi(\vec{z}))$ for all non-Boolean $q(\vec{z}) \in \mathcal{C}$ (by point 3). But then, there is a Boolean $q \in \mathcal{C}$ with $q \in X$ (as otherwise $(\phi(\vec{y}), \mathcal{C}, \pi) \in \mathcal{Q}_e(\mathfrak{C})$). Since by $(\phi(\vec{y}), \mathcal{C}, \pi) \in \mathcal{Q}_{e+1}(\mathfrak{C}')$ there exists a homomorphism h from $\mathfrak{D}_q$ to $\mathfrak{C}'$ such that the distance from $\mathsf{dom}(h(\mathfrak{D}_q))$ to $\mathsf{dom}(\mathfrak{D})$ is at most $e+1$, we obtain $(q, h) \in H_{e+1}(\mathfrak{C}') = \emptyset$, a contradiction. Hence, it suffices to show that $H_{e+1}(\mathfrak{C}') = \emptyset$.

For a contradiction, suppose that $H_{e+1}(\mathfrak{C}') \neq \emptyset$. We construct a new forest-model $\mathfrak{C}''$ of $\mathfrak{D}$ and $\mathcal{O}$ that satisfies conditions 1, 3,

and 4, but with $H_{e+1}(\mathfrak{C}'') \subsetneq H_{e+1}(\mathfrak{C}')$. This will contradict that $H_{e+1}(\mathfrak{C}')$ is inclusion-minimal. The construction is based on a pumping argument.

By definition we have $\mathfrak{C}' = \mathfrak{D} \cup \bigcup_{G \in \mathcal{G}} \mathfrak{C}'_G$, where $\mathcal{G}$ is the set of all maximal guarded subsets of $\mathfrak{D}$, and for each $G \in \mathcal{G}$ there is a cg-tree decomposition $(T_G, E_G, \mathsf{bag}_G)$ of $\mathfrak{C}'_G$ whose root r_G satisfies $\mathsf{dom}(\mathsf{bag}_G(r_G)) = G$. For each node $t \in T_G$, let $\mathsf{bag}^*_G(t)$ be the union of $\mathsf{bag}_G(t')$, where t' ranges over the descendants of t in (T_G, E_G), including t.

Observe that for each $(q, h) \in H_{e+1}(\mathfrak{C}')$ there is a $G \in \mathcal{G}$ and a node $t \in T_G$ at depth at least e in the tree (T_G, E_G) such that $h(\mathfrak{D}_q) \subseteq \mathsf{bag}^*_G(t)$. Indeed, since

$$\mathsf{dist}_{\mathfrak{C}'}(\mathsf{dom}(h(\mathfrak{D}_q)), \mathsf{dom}(\mathfrak{D})) \geq e + 1, \tag{1}$$

the set $h(\mathfrak{D}_q)$ does not contain any constant from $\mathsf{dom}(\mathfrak{D})$. Moreover, since $\mathfrak{D}_q$ is connected, there must be a $G \in \mathcal{G}$ and a node $t \in T_G$ such that $h(\mathfrak{D}_q) \subseteq \mathsf{bag}^*_G(t)$. This node t can be chosen to have depth at least e in (T_G, E_G), because otherwise we could construct a path of length at most e from an element in $\mathsf{dom}(h(\mathfrak{D}_q))$ to an element in $\mathsf{dom}(\mathfrak{D})$ in the Gaifman graph of $\mathfrak{C}'$, contradicting (1).

Pick any $G \in \mathcal{G}$ and $t \in T_G$ at depth e in (T_G, E_G) such that for some $(q, h) \in H_{e+1}(\mathfrak{C}')$ we have $h(\mathfrak{D}_q) \subseteq \mathsf{bag}^*_G(t)$. Let $t_0, t_1, \ldots, t_{d^*}$ be the last $d^* + 1$ nodes on the path from r_G to t in (T_G, E_G), and define $C_i := \mathsf{dom}(\mathsf{bag}_G(t_i))$ as well as $C_i^+ := C_{i-1} \cup C_i$. For each $i \in \{1, \ldots, d^*\}$, define the type and extended type of t_i as follows:

$$\theta_i := \{\phi(\vec{a}) \mid \phi(\vec{x}) \in \mathsf{cl}(\mathcal{O}, q),\ \vec{a} \in C_i^{|\vec{x}|},\ \mathfrak{C}' \models \phi(\vec{a})\},$$
$$\theta_i^+ := \{\phi(\vec{a}) \mid \phi(\vec{x}) \in \mathsf{cl}(\mathcal{O}, q),\ \vec{a} \in (C_i^+)^{|\vec{x}|},\ \mathfrak{C}' \models \phi(\vec{a})\}.$$

Note that θ_i is a C_i-type and θ_i^+ is a C_i^+-type. By our choice of d^*, there are $1 \leq i < j \leq d^*$ and a bijective mapping $f\colon C_i^+ \to C_j^+$ with $f(\theta_i^+) = \theta_j^+$ and $f(\theta_i) = \theta_j$, where $f(\theta_i^+)$ is obtained from θ_i^+ by replacing each constant a that occurs in it by $f(a)$, and likewise for $f(\theta_i)$. In particular, f is an isomorphism from $\mathsf{bag}_G(t_i)$ to $\mathsf{bag}_G(t_j)$. We extend f to an injective mapping with domain $\mathsf{dom}(\mathsf{bag}^*_G(t_i))$ such that each element that does not occur in C_i is mapped to an element in $\Delta_D \setminus \mathsf{dom}(\mathfrak{C}')$.

We are now ready to construct $\mathfrak{C}''$. First, define

$$\mathfrak{C}''_G := (\mathfrak{C}'_G \setminus \mathsf{bag}^*_G(t_j)) \cup f(\mathsf{bag}^*_G(t_i)).$$

Note that a cg-tree decomposition of $\mathfrak{C}''_G$ can be obtained from $(T_G, E_G, \mathsf{bag}_G)$ by removing the subtree of (T_G, E_G) rooted at t_j, adding a fresh copy of the subtree rooted at t_i, making its root a child of t_{j-1}, and renaming each element a that occurs in the bag of a node in the new subtree to $f(a)$. The root of this cg-tree decomposition is the root of $(T_G, E_G, \mathsf{bag}_G)$, and is therefore labeled with a bag with domain G. We let

$$\mathfrak{C}'' := \mathfrak{D} \cup \mathfrak{C}''_G \cup \bigcup_{G' \in \mathcal{G} \setminus \{G\}} \mathfrak{C}'_{G'}.$$

It is straightforward to verify that $\mathfrak{C}''$ is a forest-model of $\mathfrak{D}$ and $\mathcal{O}$ that has the desired properties. For point 4, note that $\mathfrak{C}''$ agrees with $\mathfrak{C}'$ on all elements whose distance from $\mathsf{dom}(\mathfrak{D})$ in $\mathfrak{C}''$ is at most the depth of t_j in (T_G, E_G), which is at least $e - d^* + 1$. Note that point 3 follows from point 4, because $e - d^* + 1 \geq s + 1$ and s was chosen in such a way that any non-Boolean CQ that occurs in $\mathcal{C}$ for some $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$ and that has a match into $\mathfrak{C}''$ has a match into the subinstance of $\mathfrak{C}''$ induced by all elements at distance at most s from $\mathsf{dom}(\mathfrak{D})$. ⌟

To complete the proof of Lemma 9, we apply the claim repeatedly to $\mathfrak{B}$ to obtain a sequence $\mathfrak{B}_0, \mathfrak{B}_1, \mathfrak{B}_2, \ldots$ of forest models $\mathfrak{B}_i$ of $\mathfrak{D}$ and $\mathcal{O}$ such that for each $i \geq 0$,

- $\mathcal{Q}_{d+i}(\mathfrak{B}_i) = \emptyset$;
- $\mathfrak{B}_i$ and $\mathfrak{B}_{i+1}$ agree on all elements at distance at most $s + i + 1$ from elements in $\mathsf{dom}(\mathfrak{D})$.

By compactness, we obtain a forest-model $\mathfrak{B}'$ of $\mathfrak{D}$ and $\mathcal{O}$ with $\mathcal{Q}_e(\mathfrak{B}') = \emptyset$ for all integers $e \geq 0$. Since $\mathcal{Q}$ is a decomposition of $q_0(\vec{x})$ w.r.t. $\mathcal{O}$, this implies $\mathfrak{B}' \not\models q_0(\vec{a})$, as desired. □

We now use Lemma 9 to show that any decomposition of a UCQ $q(\vec{x})$ w.r.t. a uGF(=) or uGC$_2$(=) ontology $\mathcal{O}$ can be turned into a strong decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$.

Lemma 10 *Let $\mathcal{O}$ be a uGF(=) or uGC$_2$(=) ontology, and let $\mathcal{Q}$ be a decomposition of a UCQ $q(\vec{x})$ w.r.t. $\mathcal{O}$. Then there is a strong decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$.*

PROOF. Fix the constant d from Lemma 9. Consider a pair $p = (\phi(\vec{x}), \mathcal{C}) \in \mathcal{Q}$, and let $q_1(\vec{z}_1), \ldots, q_n(\vec{z}_n)$ be an enumeration of all CQs in $\mathcal{C}$. For each $i \in \{1, \ldots, n\}$, we define a set $\mathcal{C}_i$ of rAQs as follows. If $q_i(\vec{z}_i)$ is non-Boolean, then $q_i(\vec{x})$ is an rAQ and we define $\mathcal{C}_i := \{q_i(\vec{x})\}$. Otherwise, if q_i is Boolean, let $q_i \leftarrow \phi$. Then, $\mathcal{C}_i$ consists of all rAQs of the form

$$q'(x) \leftarrow R_1(\vec{y}_1) \wedge \cdots \wedge R_e(\vec{y}_e) \wedge \phi,$$

for $e \leq d$, where

- x occurs in $\vec{y}_1$;
- each R_i is an at least binary relation symbol in $\mathcal{O}$;
- $\vec{y}_e$ contains a variable y in one of the atoms of ϕ, but no other variable from ϕ occurs in any of the $\vec{y}_i$.

Let $\mathcal{Q}'_p$ be the set of all pairs $(\phi(\vec{x}), \{q'_i(\vec{z}_i) \mid 1 \leq i \leq n\})$, where $q'_i(\vec{z}_i) \in \mathcal{C}_i$ for each $i \in \{1, \ldots, n\}$. By the construction of $\mathcal{Q}'_p$ and our choice of d, it follows that the following are equivalent for each model $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$:

1. there exists an assignment π such that $\pi(\vec{x}) = \vec{a}$, $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and $\mathfrak{A} \models q_i(\pi(\vec{z}_i))$ for each $1 \leq i \leq n$;
2. there exists a pair $(\phi(\vec{x}), \mathcal{C}') \in \mathcal{Q}'_p$ and an assignment π such that $\pi(\vec{x}) = \vec{a}$, $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and $\mathfrak{A} \models q'(\pi(\vec{z}))$ for each $q'(\vec{z}) \in \mathcal{C}'$.

Altogether, this implies that $\mathcal{Q}' := \bigcup_{p \in \mathcal{Q}} \mathcal{Q}'_p$ is a strong decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$. □

Corollary 1 *For each uGF(=) or uGC$_2$(=) ontology $\mathcal{O}$ and each UCQ $q(\vec{x})$ there is a strong decomposition $\mathcal{Q}$ of $q(\vec{x})$ w.r.t. $\mathcal{O}$. Moreover, if $\mathcal{O}$ is an uGC$_2$(=) ontology, then each rAQ that occurs in a pair of $\mathcal{Q}$ has a single free variable.*

We are now ready to give the proof of Theorem 4.

PROOF OF THEOREM 4. PART 1 AND 2: Since the "only if" directions are trivial, it suffices to focus on the direction from rAQ-evaluation to UCQ-evaluation. Suppose that rAQ-evaluation w.r.t. $\mathcal{O}$ is in PTIME (resp., Datalog$^{\neq}$-rewritable), and let $q(\vec{x})$ be a UCQ. We show that evaluating $q(\vec{x})$ w.r.t. $\mathcal{O}$ is in PTIME (resp., Datalog$^{\neq}$-rewritable).

Let $\mathcal{Q}$ be a strong decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$, which exists by Corollary 1. Then for all instances $\mathfrak{D}$ and all tuples $\vec{a}$ over

$\mathsf{dom}(\mathfrak{D})$ of length $|\vec{x}|$ we have $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ iff the following is true:

There exists $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$ and an assignment π of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$ with

1. $\pi(\vec{x}) = \vec{a}$,
2. $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and
3. $\mathcal{O}, \mathfrak{D} \models q'(\pi(\vec{z}))$ for each rAQ $q'(\vec{z})$ in $\mathcal{C}$.

(2)

Indeed, since rAQ-evaluation w.r.t. $\mathcal{O}$ is in PTIME (this also holds if rAQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable), we may assume that $\mathcal{O}$ is rAQ-materializable (by Theorem 3). Pick a rAQ-materialization $\mathfrak{A}$ of $\mathcal{O}$ and $\mathfrak{D}$. Then condition 2 of Definition 5 means that there exists a pair $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$ and an assignment π of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$ such that $\pi(\vec{x}) = \vec{a}$, $\mathfrak{D} \models \phi(\pi(\vec{y}))$, and $\mathfrak{A} \models q'(\pi(\vec{z}))$ for each rAQ $q'(\vec{z})$ in $\mathcal{C}$. Since $\mathfrak{A}$ is a rAQ-materialization of $\mathcal{O}$ and $\mathfrak{D}$, we can replace $\mathfrak{A} \models q'(\pi(\vec{z}))$ in the third condition by $\mathcal{O}, \mathfrak{D} \models q'(\pi(\vec{z}))$, which yields (2).

If rAQ-evaluation w.r.t. $\mathcal{O}$ is in PTIME, then (2) yields a polynomial time procedure for evaluating q w.r.t. $\mathcal{O}$.

Moreover, if rAQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable, then we can construct a Datalog$^{\neq}$ program for evaluating q w.r.t. $\mathcal{O}$ as follows. Let $\mathcal{Q}'$ be the set of rAQs that occur in some pair of $\mathcal{Q}$. For each $q' \in \mathcal{Q}'$, let $\Pi_{q'}$ be a Datalog$^{\neq}$ program that evaluates q' w.r.t. $\mathcal{O}$. Without loss of generality we assume that the intensional predicates used in different programmes $\Pi_{q'}$ and $\Pi_{q''}$ are disjoint, and that the goal predicate of $\Pi_{q'}$ is $\mathsf{goal}_{q'}$. Now let Π be the Datalog$^{\neq}$ program containing the rules of all programs $\Pi_{q'}$, for $q' \in \mathcal{Q}'$, and the following rule for each $(\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$:

$$\mathsf{goal}(\vec{x}) \leftarrow \phi(\vec{y}) \wedge \bigwedge_{q'(\vec{z}) \in \mathcal{C}} \mathsf{goal}_{q'}(\vec{z}).$$

Note that if each $\Pi_{q'}$ is a Datalog program, then Π is a Datalog program as well. Using (2) it is straightforward to verify that for all instances $\mathfrak{D}$ we have $\mathfrak{D} \models \Pi(\vec{a})$ iff $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$.

PART 3: The "if" directions are trivial, so we focus on the direction from UCQ-evaluation to rAQ-evaluation. Suppose that UCQ-evaluation w.r.t. $\mathcal{O}$ is CONP-hard, and let $q(\vec{x})$ be a UCQ that witnesses this. We show that rAQ-evaluation w.r.t. $\mathcal{O}$ is CONP-hard via a polynomial-time reduction from evaluating q w.r.t. $\mathcal{O}$.

We start by describing a translation of instances $\mathfrak{D}$ and tuples $\vec{a}$, and will show afterwards that this translation is a polynomial-time reduction from evaluating q w.r.t. $\mathcal{O}$ to evaluating a suitably chosen rAQ w.r.t. $\mathcal{O}$.

Let $\mathcal{Q}$ be a strong decomposition of $q(\vec{x})$ w.r.t. $\mathcal{O}$. Fix an enumeration $q'_1(\vec{z}_1), \ldots, q'_m(\vec{z}_m)$ of all rAQs that occur in some pair of $\mathcal{Q}$, and let k_i be the length of $\vec{z}_i$. Without loss of generality, we can assume that each $\mathfrak{D}_{q'_i}$ is consistent w.r.t. $\mathcal{O}$. We use fresh relation symbols R, S, and T_i ($1 \leq i \leq m$), where R and S are binary, and T_i is $(k_i + 1)$-ary. Note that each of these relation symbols is at most binary in the case that $\mathcal{O}$ is a uGC$_2(=)$ ontology.

Given an instance $\mathfrak{D}$ and a tuple $\vec{a} \in \mathsf{dom}(\mathfrak{D})^{|\vec{x}|}$, we compute a new instance $\tilde{\mathfrak{D}}$ by adding the following atoms to $\mathfrak{D}$. First, we add all atoms of $\mathfrak{D}_{q'_i}$, for each $i \in \{1, \ldots, m\}$, where we assume w.l.o.g. that the domain of $\mathfrak{D}_{q'_i}$ is disjoint from that of $\mathfrak{D}$ and $\mathfrak{D}_{q'_j}$ for $j \neq i$. Let $\vec{c}_i$ be the tuple of elements in $\mathfrak{D}_{q'_i}$ that represents the tuple $\vec{z}_i$ of the answer variables of q'_i. Next, we add the following atoms for each pair $p = (\phi(\vec{y}), \mathcal{C}) \in \mathcal{Q}$, and for each assignment π of elements in $\mathsf{dom}(\mathfrak{D})$ to the variables in $\vec{y}$ that satisfies $\pi(\vec{x}) = \vec{a}$ and $\mathfrak{D} \models \phi(\pi(\vec{y}))$:

- $R(a_0, a_p)$;
- $S(a_p, a_{p,\pi})$;
- $T_i(a_{p,\pi}, \pi(\vec{z}_i))$ for each $i \in \{1, \ldots, m\}$ with $q'_i \in \mathcal{C}$;
- $T_i(a_{p,\pi}, \vec{c}_i)$ for each $i \in \{1, \ldots, m\}$ with $q'_i \notin \mathcal{C}$.

Since $\mathcal{Q}$ is of constant size, the instance $\tilde{\mathfrak{D}}$ can be computed in time polynomial in the size of $\mathfrak{D}$.

It is now straightforward to verify that $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ holds iff $\tilde{\mathfrak{D}}, \mathcal{O} \models \tilde{q}(a_0)$, where $\tilde{q}(x)$ is the rAQ

$$\tilde{q}(x) \leftarrow R(x, y) \wedge S(y, z) \wedge \bigwedge_{i=1}^{m} \big(T_i(z, \vec{u}_i) \wedge q'_i(\vec{u}_i)\big).$$

Since evaluating $q(\vec{x})$ w.r.t. $\mathcal{O}$ is CONP-hard, we conclude that evaluating $\tilde{q}(x)$ w.r.t. $\mathcal{O}$ is CONP-hard. □

E. PROOFS FOR SECTION 4

Theorem 5 (restated) *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. Let q be an rAQ. We show that answering q w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable. We provide separate proofs for the case that $\mathcal{O}$ is a uGF(=) ontology (Part 1) and for the case that $\mathcal{O}$ is a uGC$_2$(=) ontology (Part 2).

Part 1: $\mathcal{O}$ is a uGF(=) ontology. To simplify the presentation, we will regard q as an openGF formula, which we can do w.l.o.g. because q has a connected guarded tree decomposition whose root is labeled by the answer variables of q. Let $\mathsf{cl}(\mathcal{O}, q)$ be the smallest set satisfying:

- $\mathcal{O} \cup \{q\} \subseteq \mathsf{cl}(\mathcal{O}, q)$;
- $\mathsf{cl}(\mathcal{O}, q)$ contains one atomic formula $R(\vec{x})$ for each relation symbol R that occurs in $\mathcal{O}$ or q, where $\vec{x}$ is a tuple of distinct variables;
- $\mathsf{cl}(\mathcal{O}, q)$ contains a single equality atom $x = y$, where x and y are distinct variables;
- $\mathsf{cl}(\mathcal{O}, q)$ is closed under subformulas and single negation, where the subformulas of a formula $\phi = Q\vec{x}\,\psi$ with a quantifier Q are ϕ itself and ψ.

Let w be the maximum arity of a relation symbol in $\mathcal{O}$ or q, and let $x_1, \ldots, x_{2w-1}$ be distinct variables that do not occur in $\mathcal{O}$ or q. Given a tuple $\vec{x}$ over $X = \{x_1, \ldots, x_{2w-1}\}$, an *$\vec{x}$-type* θ is a maximal consistent set of formulas, where each formula in θ is obtained from a formula $\phi(\vec{y}) \in \mathsf{cl}(\mathcal{O}, q)$ by substituting a variable from $\vec{x}$ for each variable in $\vec{y}$, and one formula in θ is a relational atomic formula containing all the variables in θ. If θ_i is an $\vec{x}_i$-type for each $i \in \{1, 2\}$, then θ_1 and θ_2 are *compatible* if they agree on all formulas containing only the variables that occur both in $\vec{x}_1$ and $\vec{x}_2$. An $\vec{x}$-type θ is *realizable* in an interpretation $\mathfrak{A}$ if there is an assignment π of elements in $\mathsf{dom}(\mathfrak{A})$ to the variables in $\vec{x}$ such that $\mathfrak{A} \models \phi(\pi(\vec{y}))$ for each $\phi(\vec{y}) \in \theta$. In this case we also say that $\vec{a} = \pi(\vec{x})$ *realizes* θ in $\mathfrak{A}$. Let $\mathsf{tp}(\vec{x})$ be the set of all $\vec{x}$-types that are realizable in some model of $\mathcal{O}$. Since $\mathcal{O}$ and q are fixed, each $\mathsf{tp}(\vec{x})$ is of constant size and can be computed in constant time using a standard satisfiability procedure for the guarded fragment [29].

We now describe a Datalog$^{\neq}$ program Π for evaluating q w.r.t. $\mathcal{O}$. For each $l \in \{1, \ldots, w\}$, each l-tuple $\vec{x}$ over X, and each $\Theta \subseteq \mathsf{tp}(\vec{x})$, the program uses an intensional l-ary relation symbol P^l_Θ. Intuitively, $P^l_\Theta(\vec{a})$ encodes an assignment of possible types (namely those in Θ) for $\vec{a}$ in a model of $\mathcal{O}$. In the description

below, l_1 and l_2 range over integers in $\{1, \dots, w\}$, and $\vec{x}_1$ and $\vec{x}_2$ range over tuples over X of length l_1 and l_2, respectively, such that $\vec{x}_2$ contains at least one variable from $\vec{x}_1$. Let k be the arity of q. The program contains the following rules:

1. $P^{l_1}_{\Theta}(\vec{x}_1) \leftarrow R(\vec{y}) \wedge \alpha(\vec{z})$, where Θ is the set of all $\theta \in \mathsf{tp}(\vec{x}_1)$ that contain $R(\vec{y})$ and $\alpha(\vec{z})$, α is an atomic formula (possibly an equality) or an inequality, and $\vec{y}$ contains exactly the variables from $\vec{x}_1$;
2. $P^{l_1}_{\Theta}(\vec{x}_1) \leftarrow P^{l_1}_{\Theta_1}(\vec{x}_1) \wedge P^{l_2}_{\Theta_2}(\vec{x}_2)$, where $\Theta_i \subseteq \mathsf{tp}(\vec{x}_i)$, and Θ is the set of all $\theta_1 \in \Theta_1$ such that there exists a $\theta_2 \in \Theta_2$ that is compatible with θ_1;
3. $P^{l_1}_{\Theta_1 \cap \Theta_2}(\vec{x}_1) \leftarrow P^{l_1}_{\Theta_1}(\vec{x}_1) \wedge P^{l_1}_{\Theta_2}(\vec{x}_1)$, where $\Theta_i \subseteq \mathsf{tp}(\vec{x}_1)$;
4. $\mathsf{goal}(x_{i_1}, \dots, x_{i_k}) \leftarrow P^{l_1}_{\Theta}(\vec{x}_1)$, where $\Theta \subseteq \mathsf{tp}(\vec{x}_1)$, and $q(x_{i_1}, \dots, x_{i_k}) \in \theta$ for each $\theta \in \Theta$;
5. $\mathsf{goal}(x_1, \dots, x_k) \leftarrow P^{l}_{\emptyset}(\vec{y})$, where $l \leq w$ and $\vec{y}$ is a tuple of l variables from X distinct from $x_1, \dots, x_k$.

It remains to show that Π is a Datalog$^{\neq}$-rewriting for q w.r.t. $\mathcal{O}$. To this end, let $\mathfrak{D}$ be an instance. We show that $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$ iff $\mathfrak{D} \not\models \Pi(\vec{a})$.

For the "only if" direction, assume $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$, and let $\mathfrak{B}$ be a model of $\mathfrak{D}$ and $\mathcal{O}$ with $\mathfrak{B} \not\models q(\vec{a})$. Consider a tuple $\vec{b} = (b_1, \dots, b_l) \in \mathsf{dom}(\mathfrak{D})^l$ for some $l \in \{1, \dots, w\}$. Note that if $\vec{b}$ is not guarded in $\mathfrak{B}$, then for all tuples $\vec{x} \in X^l$ and all $\Theta \subseteq \mathsf{tp}(\vec{x})$ the program Π does not derive $P^l_{\Theta}(\vec{b})$ on input $\mathfrak{D}$. This is because of the rules in line 1, which require $\vec{b}$ to be guarded in order to derive $P^l_{\Theta}(\vec{b})$. For each tuple $\vec{y} = (y_1, \dots, y_l) \in X^l$, define $\theta_{\vec{y}}(\vec{b})$ to be the set of all formulas obtained from a formula $\phi(z_1, \dots, z_n) \in \mathsf{cl}(\mathcal{O}, q)$ and $1 \leq i_1, \dots, i_n \leq l$ with $\mathfrak{B} \models \phi(b_{i_1}, \dots, b_{i_n})$ by substituting y_{i_j} for each z_j. Since $\vec{b}$ is guarded in $\mathfrak{D}$, $\theta_{\vec{y}}(\vec{b})$ contains a relational atomic formula that contains all variables from $\vec{y}$, hence $\theta_{\vec{y}}(\vec{b})$ is a $\vec{y}$-type. Furthermore, $\theta_{\vec{y}}(\vec{b})$ is realizable in $\mathfrak{B}$, which implies $\theta_{\vec{y}}(\vec{b}) \in \mathsf{tp}(\vec{y})$. By induction on rule applications of Π one can show that for each guarded tuple $\vec{b}$ in $\mathsf{dom}(\mathfrak{B})$ of length l, each tuple $\vec{y} \in X^l$, and each $\Theta \subseteq \mathsf{tp}(\vec{y})$ such that $P^l_{\Theta}(\vec{b})$ is derivable by Π on $\mathfrak{D}$ we have $\theta_{\vec{y}}(\vec{a}) \in \Theta$. Since $\mathfrak{B} \not\models q(\vec{a})$, for each guarded tuple $\vec{b} = (b_1, \dots, b_l)$ in $\mathfrak{D}$ with $\vec{a} = (b_{i_1}, \dots, b_{i_k})$ and each $\vec{y} = (y_1, \dots, y_l) \in X^l$ we have $q(y_{i_1}, \dots, y_{i_k}) \notin \theta_{\vec{y}}(\vec{b})$. Altogether, this implies that $\mathfrak{D} \not\models \Pi(\vec{a})$.

For the "if" direction, assume $\mathfrak{D} \not\models \Pi(\vec{a})$. Let $G_{\vec{a}}$ be a maximal guarded set of $\mathfrak{D}$ containing the elements of $\vec{a}$. Recall the definition of the uGF(=)-unravelling of $\mathfrak{D}$ from Section 4. We show that $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$, where $\vec{b}$ is the copy of $\vec{a}$ in $\mathsf{bag}(G_{\vec{a}})$, which implies $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$ by unravelling tolerance of $\mathcal{O}$. More precisely, we construct a model $\mathfrak{B}$ of $\mathfrak{D}^u$ and $\mathcal{O}$ with $\mathfrak{B} \not\models q(\vec{b})$.

Observe that for each $l \in \{1, \dots, w\}$, each $\vec{c} \in \mathsf{dom}(\mathfrak{D})^l$, each $\Theta \subseteq \mathsf{tp}(\vec{x})$, and each bijective mapping $f\colon X \to X$, the program Π derives $P^l_{\Theta}(\vec{c})$ iff it derives $P^l_{f(\Theta)}(\vec{c})$, where $f(\Theta)$ is the $f(\vec{x})$-type obtained from Θ by substituting $f(x)$ for each variable x in $\vec{x}$. In other words, the sets Θ of types that Π derives for each tuple $\vec{c}$ do not depend on the choice of the variables $\vec{x}$. Observe also that for each guarded tuple $\vec{c}$ in $\mathfrak{D}$ there is a unique minimal $\Theta(\vec{c}) \subseteq \mathsf{tp}(x_1, \dots, x_l)$ such that Π derives $P^l_{\Theta(\vec{c})}(\vec{c})$. Since Π does not derive $\mathsf{goal}(\vec{a})$, we must have $\Theta(\vec{c}) \neq \emptyset$ for all guarded tuples $\vec{c}$ in $\mathfrak{D}$. Furthermore, if $\vec{c} = (c_1, \dots, c_l)$ and $\vec{a} = (c_{i_1}, \dots, c_{i_k})$, then $q(x_{i_1}, \dots, x_{i_k}) \notin \theta$ for some $\theta \in \Theta(\vec{c})$. Let us denote the set of such types in $\Theta(\vec{c})$ by $\Theta^{\neg q}(\vec{c})$.

To construct the desired model $\mathfrak{B}$, we first assign to each maximally guarded tuple $\vec{c}$ of $\mathfrak{D}^u$ a type in $\Theta(\vec{c})$ as follows. Recall the definition of the tree $T(\mathfrak{D})$ and the bags $\mathsf{bag}(t)$, $t \in T(\mathfrak{D})$, used in the definition of the uGF-unravelling in Section 4. For each node t of $T(\mathfrak{D})$, let $G_t := \mathsf{dom}(\mathsf{bag}(t))$ and $\vec{c}_t$ a $|G_t|$-tuple of all elements in G_t. We inductively assign to each node t of $T(\mathfrak{D})$ a $(x_1, \dots, x_{|G_t|})$-type θ_t as follows. If $t = G$ for a maximal guarded subset G of $\mathfrak{D}$, pick a type $\theta_t \in \Theta^{\neg q}(\vec{c}_t)$. For the induction step, suppose that $t = t'G$ and $\mathsf{tail}(t') = G'$. Let $\vec{c}_{t'} = (c'_1, \dots, c'_m)$, $\vec{c}_t = (c_1, \dots, c_n)$, and define a bijective mapping $f\colon X \to X$ such that for all $i \in \{1, \dots, n\}$,

- if $c_i = c'_j$, then $f(x_i) = x_j$;
- if $c_i \notin \{c'_1, \dots, c'_m\}$, then $f(x_i) \notin \{x_1, \dots, x_m\}$.

Since $\theta_{t'} \in \Theta(\vec{c}_{t'})$, there exists a type $\theta_t \in \Theta(\vec{c}_t)$ such that $\theta_{t'}$ and $f(\theta_t)$ are compatible. This follows from the rules in line 2 of the definition of Π.

Finally, for each node t of $T(\mathfrak{D})$, pick a model $\mathfrak{B}_t$ of $\mathcal{O}$ such that $\vec{c}_t$ realizes θ_t in $\mathfrak{B}_t$. Without loss of generality we assume that $\mathsf{dom}(\mathfrak{B}_t) \cap \mathsf{dom}(\mathfrak{D}^u) = G_t$ for each node t of $T(\mathfrak{D})$, and $\mathsf{dom}(\mathfrak{B}_t) \cap \mathsf{dom}(\mathfrak{B}_{t'}) = G_t \cap G_{t'}$ for every two distinct $t, t' \in T(\mathfrak{D})$ Let $\mathfrak{B}$ be the interpretation obtained from $\mathfrak{D}^u$ by hooking $\mathfrak{B}_t$ to $\mathfrak{D}^u$ for each node t of $T(\mathfrak{D})$:

$$\mathfrak{B} \;:=\; \mathfrak{D}^u \cup \bigcup_{t \in T(\mathfrak{D})} \mathfrak{B}_t.$$

Clearly, $\mathfrak{B}$ is a model of $\mathfrak{D}^u$. It remains to show that $\mathfrak{B}$ is a model of $\mathcal{O}$ with $\mathfrak{B} \not\models q(\vec{b})$.

Claim. For all openGF formulas $\phi(\vec{x})$, all guarded tuples $\vec{c}$ of $\mathfrak{B}$, and all nodes $t \in T(\mathfrak{D})$ with $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_t)$,

$$\mathfrak{B}_t \models \phi(\vec{c}) \iff \mathfrak{B} \models \phi(\vec{c}). \tag{3}$$

Proof. By induction on the structure of ϕ. For the base case assume that ϕ is an atomic formula. Let $\vec{c}$ be a guarded tuple of $\mathfrak{B}$ and let $t \in T(\mathfrak{D})$ be such that $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_t)$. Then, $\mathfrak{B}_t \models \phi(\vec{c})$ implies $\mathfrak{B} \models \phi(\vec{c})$ because $\mathfrak{B}_t \subseteq \mathfrak{B}$. For the converse assume that $\mathfrak{B} \models \phi(\vec{c})$. Since ϕ is atomic, this implies $\mathfrak{B}_{t'} \models \phi(\vec{c})$ for some $t' \in T(\mathfrak{D})$ with $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_{t'})$. By the choice of the types θ_t and $\theta_{t'}$, we obtain $\mathfrak{B}_t \models \phi(\vec{c})$.

For the inductive step, we distinguish the following cases:

Case 1: $\phi(\vec{x}) = \neg\phi'(\vec{x})$. For each guarded tuple $\vec{c}$ of $\mathfrak{B}$ and each node $t \in T(\mathfrak{D})$ with $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_t)$, we have

$$\begin{aligned} \mathfrak{B}_t \models \phi(\vec{c}) &\iff \mathfrak{B}_t \not\models \phi'(\vec{c}) \\ &\iff \mathfrak{B} \not\models \phi'(\vec{c}) \iff \mathfrak{B} \models \phi(\vec{c}), \end{aligned}$$

where the second equivalence follows from the induction hypothesis.

Case 2: $\phi(\vec{x}) = \phi_1(\vec{x}_1) \wedge \phi_2(\vec{x}_2)$. Consider a guarded tuple $\vec{c}$ of $\mathfrak{B}$ and a node $t \in T(\mathfrak{D})$ with $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_t)$. Let $\vec{c}_i$ be the projection of $\vec{c}$ onto the positions of $\vec{x}$ that contain a variable from $\vec{x}_i$. Then,

$$\begin{aligned} \mathfrak{B}_t \models \phi(\vec{c}) &\iff \mathfrak{B}_t \models \phi_i(\vec{c}_i) \text{ for each } i \in \{1, 2\} \\ &\iff \mathfrak{B} \models \phi_i(\vec{c}_i) \text{ for each } i \in \{1, 2\} \\ &\iff \mathfrak{B} \models \phi(\vec{c}), \end{aligned}$$

where the second equivalence follows from the induction hypothesis.

Case 3: $\phi(\vec{x}) = \forall\vec{y}(\alpha(\vec{x}, \vec{y}) \to \psi(\vec{x}, \vec{y}))$. Consider a guarded tuple $\vec{c}$ of $\mathfrak{B}$ and a $t \in T(\mathfrak{D})$ with $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_t)$.

We first prove that $\mathfrak{B}_t \models \phi(\vec{c})$ implies $\mathfrak{B} \models \phi(\vec{c})$. Suppose that $\mathfrak{B}_t \models \phi(\vec{c})$, and let $\mathfrak{B} \models \alpha(\vec{c}, \vec{d})$ for some tuple $\vec{d}$. Since α is atomic, we have $\mathfrak{B}_{t'} \models \alpha(\vec{c}, \vec{d})$ for some node $t' \in T(\mathfrak{D})$ with

$\vec{c}, \vec{d} \subseteq \mathsf{dom}(\mathfrak{B}_{t'})$. Note that $\vec{c}$ contains only values in $\mathsf{dom}(\mathfrak{B}_t) \cap \mathsf{dom}(\mathfrak{B}_{t'})$ and that $\vec{c}$ is non-empty (because ϕ is open). By the choice of the types θ_t and $\theta_{t'}$ and since $\mathfrak{B}_t \models \phi(\vec{c})$, we obtain $\mathfrak{B}_{t'} \models \phi(\vec{c})$. Hence, $\mathfrak{B}_{t'} \models \psi(\vec{c}, \vec{d})$. The induction hypothesis now yields $\mathfrak{B} \models \psi(\vec{c}, \vec{d})$. We have thus shown that $\mathfrak{B} \models \phi(\vec{c})$.

For the converse, assume $\mathfrak{B} \models \phi(\vec{c})$, and let $\mathfrak{B}_t \models \alpha(\vec{c}, \vec{d})$ for some tuple $\vec{d}$. Since α is atomic and $\mathfrak{B}_t \subseteq \mathfrak{B}$, we have $\mathfrak{B} \models \alpha(\vec{c}, \vec{d})$, and therefore $\mathfrak{B} \models \psi(\vec{c}, \vec{d})$. Since $\vec{c}, \vec{d}$ is guarded and contained in $\mathsf{dom}(\mathfrak{B}_t)$, the induction hypothesis implies $\mathfrak{B}_t \models \psi(\vec{c}, \vec{d})$. This shows that $\mathfrak{B}_t \models \phi(\vec{c})$.

Case 4: $\phi(\vec{x}) = \exists \vec{y}(\alpha(\vec{x}, \vec{y}) \wedge \psi(\vec{x}, \vec{y}))$. Consider a guarded tuple $\vec{c}$ of $\mathfrak{B}$ and a node $t \in T(\mathfrak{D})$ with $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_t)$.

If $\mathfrak{B}_t \models \phi(\vec{c})$, then there exists a tuple $\vec{d} \subseteq \mathsf{dom}(\mathfrak{B}_t)$ such that $\mathfrak{B}_t \models \alpha(\vec{c}, \vec{d})$ and $\mathfrak{B}_t \models \psi(\vec{c}, \vec{d})$. By the induction hypothesis, this implies $\mathfrak{B} \models \alpha(\vec{c}, \vec{d})$ and $\mathfrak{B} \models \psi(\vec{c}, \vec{d})$, and hence $\mathfrak{B} \models \phi(\vec{c})$.

Conversely, if $\mathfrak{B} \models \phi(\vec{c})$, then there exists a tuple $\vec{d}$ such that $\mathfrak{B} \models \alpha(\vec{c}, \vec{d})$ and $\mathfrak{B} \models \psi(\vec{c}, \vec{d})$. Since α is atomic, there exists a node $t' \in T(\mathfrak{D})$ with $\vec{c}, \vec{d} \subseteq \mathsf{dom}(\mathfrak{B}_{t'})$, and hence by the induction hypothesis we obtain $\mathfrak{B}_{t'} \models \alpha(\vec{c}, \vec{d})$ and $\mathfrak{B}_{t'} \models \psi(\vec{c}, \vec{d})$, and therefore $\mathfrak{B}_{t'} \models \phi(\vec{c})$. By the choice of θ_t and $\theta_{t'}$, we obtain $\mathfrak{B}_t \models \phi(\vec{c})$. ⌟

Using the claim we now show that $\mathfrak{B}$ is a model of $\mathcal{O}$ with $\mathfrak{B} \not\models q(\vec{b})$. By construction we have $\theta_t \in \Theta^{\neg q}(\vec{c}_t)$, where $t = G_{\vec{a}}$. This implies $\mathfrak{B}_t \not\models q(\vec{b})$, and using the claim we obtain $\mathfrak{B} \not\models q(\vec{b})$.

To prove that $\mathfrak{B}$ is a model of $\mathcal{O}$, let $\psi = \forall \vec{x}(\alpha(\vec{x}) \to \phi(\vec{x}))$ be a sentence in $\mathcal{O}$, and let $\vec{c}$ be a tuple with $\mathfrak{B} \models \alpha(\vec{c})$. Since α is atomic, $\vec{c}$ is a guarded tuple in $\mathfrak{B}$. In particular, since $\mathfrak{B} \models \alpha(\vec{c})$, there must be a node $t_0 \in T(\mathfrak{D})$ with $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_{t_0})$ and $\mathfrak{B}_{t_0} \models \alpha(\vec{c})$. In fact, by the choice of the types assigned to the maximally guarded tuples of $\mathfrak{D}^u$, every node $t \in T(\mathfrak{D})$ with $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_t)$ must satisfy $\mathfrak{B}_t \models \alpha(\vec{c})$. Now let t be a node in $T(\mathfrak{D})$ such that $\vec{c} \subseteq \mathsf{dom}(\mathfrak{B}_t)$. Since $\mathfrak{B}_t \models \alpha(\vec{c})$ and $\mathfrak{B}_t$ is a model of $\mathcal{O}$, we get $\mathfrak{B}_t \models \phi(\vec{c})$, so by (3) we have $\mathfrak{B} \models \phi(\vec{c})$. Therefore, $\mathfrak{B} \models \psi$. This holds for all sentences in $\mathcal{O}$, hence $\mathfrak{B}$ is a model of $\mathcal{O}$.

If $\mathcal{O}$ is a uGF-ontology, we obtain a Datalog-rewriting from Π by removing inequalities from types and the rules in line 1.

Part 2: $\mathcal{O}$ is a uGC$_2(=)$ ontology. Analogous to part 1 we regard q as an openGC$_2$ formula, which we can do w.l.o.g. because q has a cg-tree decomposition where the domain of each bag consists of at most two elements and whose root bag has the answer variables of q as its domain. We define $\mathsf{cl}(\mathcal{O}, q)$, types, and the corresponding notion of realization as in part 1. Since in the context of uGC$_2(=)$ we work over an at most binary signature, we will only use types over one or two variables. The sets $\mathsf{tp}(x)$ and $\mathsf{tp}(x, y)$ of these types can be computed in constant time using a satisfiability procedure for the two-variable guarded counting fragment [48]. Given a $\vec{x}_0$-type θ_0 and a set $\Theta_i \subseteq \mathsf{tp}(\vec{x}_i)$ for each $i \in \{1, \dots, \ell\}$ we write $\theta_0 \rightsquigarrow (\Theta_i)_{i=1}^{\ell}$ iff there exists a $\theta_i \in \Theta_i$ for each $i \in \{1, \dots, \ell\}$ such that $\bigcup_{i=0}^{\ell} \theta_i$ is realizable in a model of $\mathcal{O}$. Furthermore, if (x, y) and (u, v) are pairs of distinct variables and $\Theta \subseteq \mathsf{tp}(u, v)$, then we write $\Theta_{x,y}$ for the set of all types in $\mathsf{tp}(x, y)$ that can be obtained from a type in Θ by renaming u to x and v to y.

We now turn to the description of the Datalog$^{\neq}$ program Π for evaluating q w.r.t. $\mathcal{O}$. The program uses distinct variables $x, y, z_0, z_1, z_2, \dots, z_{N\tau 2^\tau}$. Here, τ is one more than the number of types of two distinct variables, and N is the largest integer n such that a formula of the form $\exists^{\geq n} x\, \phi$ occurs in $\mathsf{cl}(\mathcal{O}, q)$, or 1 if there is no such formula in $\mathsf{cl}(\mathcal{O}, q)$. For each $i \in \{1, 2\}$ and $\Theta \subseteq \mathsf{tp}(u, v)$, Π uses an intensional binary relation symbol P_Θ with the same intended interpretation as the symbols P^l_Θ in part 1. Let $\mathsf{neq}_\ell := \bigwedge_{0 \leq i < j \leq \ell} z_i \neq z_j$. Then Π consists of the following rules:

1. $P_\Theta(x, y) \leftarrow R(\vec{v}) \wedge \alpha(\vec{w})$, where Θ is the set of all types in $\mathsf{tp}(x, y)$ containing $R(\vec{v})$ and $\alpha(\vec{w})$, α is an atomic formula (including an equality) or an inequality, and $\vec{v}$ contains both x and y;
2. $P_\Theta(x, z_0) \leftarrow \bigwedge_{i=0}^{l} P_{\Theta_i}(x, z_i) \wedge \mathsf{neq}_\ell$, where $\ell \leq N\tau 2^\tau$, $\Theta_i \subseteq \mathsf{tp}(x, z_i)$ for $i = 0, 1, \dots, \ell$, and Θ consists of all $\theta_0 \in \Theta_0$ with $\theta_0 \rightsquigarrow (\Theta_i)_{i=1}^{\ell}$;
3. $P_{\Theta_1 \cap \Theta_2}(x, y) \leftarrow \bigwedge_{i=1}^{2} P_{\Theta_i}(x, y)$ with $\Theta_i \subseteq \mathsf{tp}(x, y)$;
4. $\mathsf{goal}(\vec{v}) \leftarrow P_\Theta(x, y)$, where $\Theta \subseteq \mathsf{tp}(x, y)$, and $q(\vec{v}) \in \theta$ for each $\theta \in \Theta$;
5. $\mathsf{goal}(\vec{v}) \leftarrow P_\emptyset(x)$.

In addition, for each set $\Theta \subseteq \mathsf{tp}(x, y)$, the program contains $P_{\Theta_{x,z_i}}(x, y) \leftarrow P_\Theta(x, y)$ and $P_\Theta(x, z_0) \leftarrow P_{\Theta_{x,z_0}}(x, z_0)$ to rename variables in types, and $P_{\Theta_{y,x}}(y, x) \leftarrow P_\Theta(x, y)$ to swap positions of variables.

Intuitively, Π computes for each guarded tuple (a, b) of an instance $\mathfrak{D}$ a set $\Theta(a, b)$ of (x, y)-types θ_0 that contain all information about the atomic formulas that hold in $\mathfrak{D}_{|\{a,b\}}$, and for each collection $(a, c_1), \dots, (a, c_\ell)$ of $\ell \leq N\tau 2^\tau$ guarded tuples of $\mathfrak{D}$ that have a non-empty intersection with (a, b) there are types $\theta_i \in \Theta(a, c_i)$ such that $\bigcup_{i=0}^{\ell} (\theta_i)_{x,z_i}$ is realizable in a model of $\mathcal{O}$. The renaming of the variables in each θ_i is necessary to account for the overlap of the tuples (a, b) and $\vec{c}_i$. Π derives $\mathsf{goal}(\vec{c})$ if some $\Theta(a, b)$ is empty (which will happen if $\mathfrak{D}$ is inconsistent w.r.t. $\mathcal{O}$) or if $\Theta(a, b)$ contains $q(\vec{v})$ for some (a, b), where the ith variable in $\vec{v}$ is x if the ith position of $\vec{a}$ is a, and it is y otherwise.

It remains to show that Π is a Datalog$^{\neq}$-rewriting for q w.r.t. $\mathcal{O}$. To this end, let $\mathfrak{D}$ be an instance. We show that $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$ iff $\mathfrak{D} \not\models \Pi(\vec{a})$.

The "only if" direction is similar to part 1. For the "if" direction, assume $\mathfrak{D} \not\models \Pi(\vec{a})$. Recall the definition of the uGF$_2$-unravelling of $\mathfrak{D}$ from Section 4, and let $G_{\vec{a}}$ be a maximal guarded set of $\mathfrak{D}$ containing the elements of $\vec{a}$. As in part 1, it suffices to construct a model $\mathfrak{B}$ of $\mathfrak{D}^u$ and $\mathcal{O}$ with $\mathfrak{B} \not\models q(\vec{b})$, where $\vec{b}$ is the copy of $\vec{a}$ in $\mathrm{bag}(G_{\vec{a}})$.

Observe that for each guarded tuple $\vec{c}$ in $\mathfrak{D}$ there is a unique minimal set $\Theta(\vec{c}) \subseteq \mathsf{tp}(x, y)$ such that Π derives $P_{\Theta(\vec{c})}(\vec{c})$ on input $\mathfrak{D}$. Since Π does not derive $\mathsf{goal}(\vec{a})$, we must have $\Theta(\vec{c}) \neq \emptyset$ for all guarded tuples $\vec{c}$ in $\mathfrak{D}$. Furthermore, if $\vec{a} = f(\vec{v})$ for a mapping from the variables in $\vec{v}$ to the constants in $\vec{c}$, then $q(\vec{v}) \notin \theta$ for some $\theta \in \Theta(\vec{c})$. Let us denote the set of such types in $\Theta(\vec{c})$ by $\Theta^{\neg q}(\vec{c})$.

To construct the desired model $\mathfrak{B}$, we first assign to each maximally guarded tuple $\vec{c}$ of $\mathfrak{D}^u$ a type in $\Theta(\vec{c}^{\uparrow})$ and a model $\mathfrak{A}_t$ of $\mathcal{O}$ as follows. Recall the definition of the tree $T(\mathfrak{D})$ and the bags $\mathrm{bag}(t)$, for $t \in T(\mathfrak{D})$, used in the definition of the uGF$_2$-unravelling in Section 4. We inductively assign to each node t of $T(\mathfrak{D})$ a (x, y)-type θ_t as follows. If $t = \{a, b\}$ for a maximal guarded set $\{a, b\}$ in $\mathfrak{D}$, let θ_t be any type in $\Theta^{\neg q}(a, b)$. For the induction step, assume that we have assigned a type θ_t to $t \in T(\mathfrak{D})$, but that $\theta_{t'}$ is undefined for each node $t' = tG$ in $T(\mathfrak{D})$. Let $\mathrm{tail}(t) = \{a, b\}$. We distinguish the following two cases.

Case 1: There exists a $t_0 \in T(\mathfrak{D})$ with $t = t_0\{a, b\}$. In this case, there is only one kind of successor node of t in $T(\mathfrak{D})$: nodes of the form $t\{a, c\}$, where a does not occur in $\mathrm{tail}(t_0)$. Let $c_1, \dots, c_n$ be an enumeration of all elements of $\mathsf{dom}(\mathfrak{D})$ such that $t_i := t\{a, c_i\}$ is a node in $T(\mathfrak{D})$. Denote by $\sim$ the equivalence relation on $\{c_1, \dots, c_n\}$ with $c_i \sim c_j$ iff $\Theta(a, c_i) = \Theta(a, c_j)$. There are

$s \leq 2^\tau$ equivalence classes w.r.t. $\sim$. Let $E_1, \dots, E_s$ be an enumeration of these classes. For each $i \in \{1, \dots, s\}$, pick an arbitrary subset E_i' of E_i of size $N\tau$, or the entire set if E_i has fewer than $N\tau$ elements. Let $c_1', \dots, c_\ell'$ be an enumeration of the elements in $E_1' \cup \dots \cup E_s'$. Then, $\ell \leq N\tau 2^\tau$.

Since $\theta_t \in \Theta(a, b)$, the rules in line 2 of the definition of Π ensure $\theta_t \rightsquigarrow (\Theta_i)_{i=1}^{\ell}$, where $\Theta_i := \{\theta_{x,z_i} \mid \theta \in \Theta(a, c_i')\}$. Hence, for each $i \in \{1, \dots, \ell\}$ there is a type $\theta_i \in \Theta(a, c_i')$ such that $\theta_t \cup \bigcup_{i=1}^{\ell} (\theta_i)_{x,z_i}$ is realizable in a model of $\mathcal{O}$. Let $\theta_{t\{a,c_i'\}} := \theta_i$ for each $i \in \{1, \dots, \ell\}$.

It remains to assign types to the elements in $E_i \setminus E_i'$, for each $i \in \{1, \dots, s\}$. By construction, we have $E_i \setminus E_i' \neq \emptyset$ only if $|E_i'| = N\tau$. Thus there must be at least one type θ_i^* that is assigned to at least N of the nodes $t\{a, c_i\}$. We define $\theta_{t\{a,c\}} := \theta_i^*$ for each $c \in E_i \setminus E_i'$.

This finishes the assignment of types $\theta_{t'}$ to all successor nodes t' of t. Note that by our choice of N and the type θ_i^*, the set $\theta := \theta_t \cup \bigcup_{i=1}^{n} (\theta_{t_i})_{x,z_i}$ is realizable in a model $\mathfrak{A}_t$ of $\mathcal{O}$. In fact, $\mathfrak{A}_t$ can be chosen such that the assignment π_t with $\pi_t(x)^\uparrow = a$ and $\pi_t(z_i)^\uparrow = c_i$ realizes θ in $\mathfrak{A}_t$.

Case 2: $t = \{a, b\}$. In this case, there are two kinds of successor nodes of t: nodes of the form $t\{a, c\}$ and nodes of the form $t\{b, c\}$. Let $t_1, \dots, t_n$ be an enumeration of all nodes of the form $t\{a, c\}$, and let $u_1, \dots, u_m$ be an enumeration of all nodes of the form $t\{b, c\}$. We assign types θ_{t_i} to each node t_i as in Case 1. We do the same for all successors nodes u_j, but here we have to use the rule $P_{\Theta_{y,x}}(y, x) \leftarrow P_\Theta(x, y)$ which implies that $\Theta(b, a) = \{\theta_{y,x} \mid \theta \in \Theta(a, b)\}$. One can now show that $\theta := \theta_t \cup \bigcup_{i=1}^{n} (\theta_{t_i})_{x,z_i} \cup \bigcup_{i=1}^{m} (\theta_{u_i})_{y,z_i'}$ is realizable in a model $\mathfrak{A}_t$ of $\mathcal{O}$, and that this model can be chosen in such a way that the assignment π_t with $\pi_t(x)^\uparrow = a$, $\pi_t(z_i)^\uparrow \in \text{tail}(t_i) \setminus \{a\}$ and $\pi_t(u_j)^\uparrow \in \text{tail}(u_j) \setminus \{b\}$ realizes θ in $\mathfrak{A}_t$.

This concludes the induction step.

We are now ready to construct $\mathfrak{B}$. Note that any two $\mathfrak{A}_t$ and $\mathfrak{A}_{t'}$ with t' a neighbor of t in $T(\mathfrak{D})$ coincide on all atoms that only involve elements of $\mathsf{dom}(\mathfrak{D}^u)$. Let $\mathfrak{B}_0$ be the collection of all atoms that occur in some $\mathfrak{A}_t$ and only involve elements of $\mathsf{dom}(\mathfrak{D}_u)$. For each $a \in \mathsf{dom}(\mathfrak{D}^u)$ we now attach the following structure $\mathfrak{B}_{\{a\}}$ to $\mathfrak{B}_0$. Pick any $t \in T(\mathfrak{D})$ such that $a \in \mathsf{dom}(\text{bag}(t))$. Using the construction in the second part of the proof of Lemma 1, we unfold $\mathfrak{A}_t$ at a into a cg-decomposable model $\mathfrak{B}_{\{a\}}$, where we identify a and its neighbors b with the copies of a and b in the bag of the root and its neighbors. We assume that all other elements have been renamed so that they do not occur in $\mathsf{dom}(\mathfrak{D}^u)$. Let

$$\mathfrak{B} := \mathfrak{B}_0 \cup \bigcup_{a \in \mathsf{dom}(\mathfrak{D}^u)} \mathfrak{B}_{\{a\}}.$$

It is not difficult to prove that $\mathfrak{B}$ is a model of $\mathfrak{D}^u$ and $\mathcal{O}$ with $\mathfrak{B} \not\models q(\vec{b})$. $\square$

F. PROOFS FOR SECTION 5

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

PROOF. We first give the proof for the ontology languages uGF(1), uGF$_2^-$(2) and uGF$^-$(1, =). Assume such an ontology $\mathcal{O}$ is given. Let $\mathfrak{D}$ be an instance and $\mathfrak{D}^u$ its uGF-unravelling. Let G_0 be a maximal guarded set in $\mathfrak{D}$, $\vec{a}$ in G_0, $\vec{b}$ the copy of $\vec{a}$ in bag(G_0), and q an rAQ. We have to show that $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ implies $\mathcal{O}, \mathfrak{D}^u \models q(\vec{b})$.

Define an equivalence relation $\sim$ on $T(\mathfrak{D})$ by setting $t \sim t'$ if tail(t) = tail(t'). Recall that for any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ the mapping $h_{t,t'}$ that sends every $e \in \text{dom}(\text{bag}(t))$ to the unique $f \in \text{dom}(\text{bag}(t'))$ with $e^\uparrow = f^\uparrow$ is an isomorphism from bag(t) to bag(t'). Call $h_{t,t'}$ the *canonical isomorphism* from bag(t) onto bag(t'). By construction of the undirected graph $(T(\mathfrak{D}), E)$, for any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ there is an automorphism $i_{t,t'}$ of $(T(\mathfrak{D}), E)$ such that $i_{t,t'}(t) = t'$ and $i_{t,t'}(s) \sim s$ for every $s \in T(\mathfrak{D})$. $i_{t,t'}$ is uniquely determined on the connected component of t in $(T(\mathfrak{D}), E)$ and induces the *extended canonical automorphism* $\hat{h}_{t,t'}$ of $\mathfrak{D}^u$ by setting $\hat{h}_{t,t'} = \bigcup_{s \in T(\mathfrak{D})} h_{s, i_{t,t'}(s)}$.

Fact 1. If $t, t' \in T(\mathfrak{D})$ such that $t \sim t'$ then $\hat{h}_{t,t'}$ is an automorphism of $\mathfrak{D}^u$.

Our aim now is to construct a materialization $\mathfrak{B}$ of $\mathcal{O}$ and $\mathfrak{D}^u$ that is a forest model such that the automorphism $\hat{h}_{t,t'}$ can be lifted to an automorphism of $\mathfrak{B}$. Once this is done it is straightforward to construct a materialization $\mathfrak{B}'$ of $\mathcal{O}$ and $\mathfrak{D}$ by hooking to any maximal guarded set G of $\mathfrak{D}$ a copy of $\mathfrak{B}_{\text{bag}(G)}$, the guarded tree decomposable subinstances of $\mathfrak{B}$ hooked to bag(G) in $\mathfrak{B}$. Then $(\mathfrak{B}', \vec{a})$ and $(\mathfrak{B}, \vec{b})$ are guarded bisimilar and if $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$ then $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$ follows.

Let $\mathfrak{B}$ be a materialization of $\mathcal{O}$ and $\mathfrak{D}^u$. (To show that $\mathfrak{B}$ exists let red($\mathfrak{D}^u$) be the sig($\mathcal{O}$)-reduct of $\mathfrak{D}$. As $\mathcal{O}$ is invariant under disjoint unions and materializable for the class of (possibly infinite) cg-tree decomposable instances $\mathfrak{D}$ with sig($\mathfrak{D}$) $\subseteq$ sig($\mathcal{O}$) there exists a materialization $\mathfrak{B}_{\text{red}}$ of red($\mathfrak{D}^u$). Clearly $\{R \mid R(\vec{a}) \in \mathfrak{B}_{\text{red}}\} \subseteq$ sig($\mathcal{O}$). Now let

$$\mathfrak{B} = \mathfrak{B}_{\text{red}} \cup \{R(\vec{a}) \in \mathfrak{D}^u \mid R \notin \text{sig}(\mathcal{O})\}$$

One can show that $\mathfrak{B}$ is a materialization of $\mathfrak{D}^u$ and $\mathcal{O}$.)

Fact 1 entails the following.

Fact 2. For all t, t' with $t \sim t'$ and $\vec{e}$ in dom(bag(t)) and any rAQ $q(\vec{x})$ such that $\vec{x}$ has the same length as $\vec{e}$:

$$\mathfrak{B} \models q(\vec{e}) \quad \Leftrightarrow \quad \mathfrak{B} \models q(h_{t,t'}(\vec{e}))$$

We now distinguish two cases.

Case 1. $\mathcal{O}$ is a uGF(1) or a uGF$_2^-$(2) ontology. The following observation is crucial for the proof (and does not hold for languages with equality such as uGF$^-$(1, =)).

Observation 1. If there is a homomorphism h from an instance $\mathfrak{D}$ to an instance $\mathfrak{D}'$ then $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ implies $\mathcal{O}, \mathfrak{D}' \models q(h(\vec{a}))$ for any CQ q and $\vec{a}$ in dom($\mathfrak{D}$).

We hook in $\mathfrak{D}^u$ to any bag(t) with $t \in T(\mathfrak{D})$ a copy of any rAQ q (regarded as an instance) from which there is a homomorphism into $\mathfrak{B}$ that is injective on the root bag of q and maps it into dom(bag(t)). In more detail, let X_t be the set of all pairs $(\pi, \mathfrak{D}_q)$ such that $\mathfrak{D}_q$ is an instance corresponding to an rAQ q and π is a homomorphism from $\mathfrak{D}_q$ to $\mathfrak{B}$ mapping the constants d_x corresponding to answer variables x of q to distinct $\pi(d_x) \in \text{dom}(\text{bag}(t))$. By renaming constants in $\mathfrak{D}_q$ we obtain an instance $\mathfrak{D}_{q,\pi}$ isomorphic to $\mathfrak{D}_q$ such that $\pi(d_x) = d_x$ and such that $\text{dom}(\mathfrak{D}_{q,\pi}) \cap \text{dom}(\mathfrak{D}^u)$ is the set of all d_x with x an answer variable of q. Now let

$$\mathfrak{D}_t = \bigcup_{(q,\pi) \in X_t} \mathfrak{D}_{q,\pi}, \quad \mathfrak{D}^{u+} = \mathfrak{D}^u \cup \bigcup_{t \in T(\mathfrak{D})} \mathfrak{D}_t$$

The following properties of $\mathfrak{D}^{u+}$ follow directly from the definition:

1. For any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ there is an isomorphism from $\mathfrak{D}_t$ onto $\mathfrak{D}_{t'}$ that extends the canonical isomorphism $h_{t,t'}$;
2. there is a homomorphism from $\mathfrak{D}^{u+}$ into $\mathfrak{B}$ preserving $\mathrm{dom}(\mathfrak{D}^u)$. Thus, by Observation 1, any materialization of $\mathfrak{D}^{u+}$ is a materialization of $\mathfrak{D}^u$ and for every CQ $q(\vec{x})$ and $\vec{d}$ in $\mathfrak{D}^u$ of the same length as $\vec{x}$:

$$\mathcal{O}, \mathfrak{D}^{u+} \models q(\vec{d}) \quad \Leftrightarrow \quad \mathcal{O}, \mathfrak{D}^{u} \models q(\vec{d})$$

Now take a materialization $\mathfrak{B}^{u+}$ of $\mathfrak{D}^{u+}$ and $\mathcal{O}$ that is a forest model of $\mathfrak{D}^{u+}$ and $\mathcal{O}$. Thus $\mathfrak{B}^{u+}$ is obtained from $\mathfrak{D}^{u+}$ by hooking cg-tree decomposable models $\mathfrak{B}_t^{u+}$ of $\mathfrak{D}_t$ to every $\mathrm{bag}(t)$ with $t \in T(\mathfrak{D})$. By Point 1 and Fact 1, we have the following:

Fact 3. For any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ the canonical automorphism $\hat{h}_{t,t'}$ extends to an isomorphism from $\mathfrak{B}^{u+}_{|\mathrm{dom}(\mathfrak{D}_t)}$ onto $\mathfrak{B}^{u+}_{|\mathrm{dom}(\mathfrak{D}_{t'})}$.

Fact 4. For any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ and any finite subinstance $\mathfrak{A}$ of $\mathfrak{B}_t^{u+}$ there exists an isomorphic embedding of $\mathfrak{A}$ into $\mathfrak{B}^{u+}_{|\mathrm{dom}(\mathfrak{D}_{t'})}$ extending the canonical automorphism $\hat{h}_{t,t'}$.

We prove Fact 4. By Fact 3 it suffices to prove that for any $t \in T(\mathfrak{D})$ and any finite subinstance $\mathfrak{A}$ of $\mathfrak{B}_t^{u+}$ there exists an isomorphic embedding of $\mathfrak{A}$ into $\mathfrak{B}^{u+}_{|\mathrm{dom}(\mathfrak{D}_t)}$ preserving $\mathrm{dom}(\mathrm{bag}(t))$. But this follows from the fact that $\mathfrak{B}^{u+}$ is a materialization of $\mathfrak{D}^u$ (Point (2)): then there is an isomorphism from $\mathfrak{A}$ to some $\mathfrak{D}_{\pi,q}$ used in the construction of $\mathfrak{D}^{u+}$ which preserves $\mathrm{dom}(\mathrm{bag}(t))$. Fix such a $\mathfrak{D}_{\pi,q}$. It remains to be proved that there does not exist any $R(\vec{a})$ with $\vec{a}$ in $\mathrm{dom}(\mathfrak{D}_{\pi,q})$ such that $R(\vec{a}) \in \mathfrak{B}^{u+} \setminus \mathfrak{D}_{\pi,q}$. But using the fact that $\mathfrak{A}$ is a subinstance of the model $\mathfrak{B}^{u+}$ of $\mathcal{O}$ and $\mathfrak{D}^{u+}$ isomorphic to $\mathfrak{D}_{\pi,q}$ one can easily construct a model of $\mathfrak{D}^{u+}$ and $\mathcal{O}$ that contains no $R(\vec{a}) \notin \mathfrak{D}_{\pi,q}$ with $\vec{a}$ in $\mathrm{dom}(\mathfrak{D}_{\pi,q})$. Thus $\mathfrak{B}^{u+}$ contains no such $R(\vec{a})$ since $\mathfrak{B}^{u+}$ is a materialization of $\mathcal{O}$ and $\mathfrak{D}^{u+}$. This finishes the proof of Fact 4.

Fact 4 easily generalizes to $\mathfrak{B}^{u+}$ (by Fact 3): for any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ and any finite subinstance $\mathfrak{A}$ of $\mathfrak{B}^{u+}$ there exists an isomorphic embedding of $\mathfrak{A}$ into $\mathfrak{B}^{u+}$ extending the canonical automorphism $\hat{h}_{t,t'}$. We now uniformize $\mathfrak{B}^{u+}$. For each $t \in T(\mathfrak{D})$ with t not a guarded set in $\mathfrak{D}$ we hook at $\mathrm{bag}(t)$ to $\mathfrak{D}^u$ an isomorphic copy $\mathfrak{B}_t^{u*}$ of the interpretation $\mathfrak{B}^{u+}_{\mathrm{bag}(G)}$ with $t \sim G$ and remove $\mathfrak{B}_t^{u+}$. Denote the resulting model by $\mathfrak{B}^{u*}$. $\mathfrak{B}^{u*}$ is uniform in the sense that for any two $t, t' \in T(\mathfrak{D})$, the cg-tree decomposable models hooked to $\mathrm{bag}(t)$ and $\mathrm{bag}(t')$ in $\mathfrak{B}^{u*}$ are isomorphic. We now show that $\mathfrak{B}^{u*}$ is a materialization of $\mathfrak{D}$ and $\mathcal{O}$. For $a \in \mathrm{dom}(\mathfrak{B}_t^{u*})$ and $t \sim G$ denote by $a^\sim$ the corresponding element of $\mathfrak{B}^{u+}_{\mathrm{bag}(G)}$ such that for $a \in \mathrm{bag}(t)$ we have $a^\uparrow = (a^\sim)^\uparrow$.

Fact 5. $\mathfrak{B}^{u*}$ is a materialization of $\mathcal{O}$ and $\mathfrak{D}^u$.

We show that $\mathfrak{B}^{u*}$ is a model of $\mathcal{O}$. Then $\mathfrak{B}^{u*}$ is a materialization of $\mathcal{O}$ and $\mathfrak{D}^u$ since it is a model of $\mathfrak{D}^u$ and since $\mathfrak{B}^{u*}$ is obtained from the materialization $\mathfrak{B}^{u+}$ of $\mathcal{O}$ and $\mathfrak{D}^u$ by replacing certain interpretations that are hooked to $\mathrm{bag}(t)$ by interpretations that are hooked to $\mathrm{bag}(t')$ for $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ which preserves the answers to rAQs (use Fact 1).

Consider first a sentence φ of the form $\forall \vec{y}(R(\vec{y}) \rightarrow \psi(\vec{y}))$ in $\mathcal{O}$, where ψ is a formula in openGF of depth one. We show that $\mathfrak{B}^{u*} \models \varphi$. Let $\mathfrak{B}^{u*} \models R(\vec{a})$ for some $\vec{a} = (a_1, \ldots, a_k)$ in $\mathrm{dom}(\mathfrak{B}^{u*})$. Then $a_1, \ldots, a_k$ are contained in $\mathfrak{B}_t^{u*}$ for some $t \in T(\mathfrak{D})$. We show that $\mathfrak{B}^{u*} \models \psi(\vec{a})$ iff $\mathfrak{B}^{u+} \models \psi(\vec{a}^\sim)$ where $\vec{a}^\sim = (a_1^\sim, \ldots, a_k^\sim)$. Assume $t \sim G$. If $a_1, \ldots, a_k \notin \mathrm{dom}(\mathfrak{D}^u)$, then this is clear by construction since the truth of $\psi(\vec{a})$ then only depends on the subinterpretation $\mathfrak{B}_t^{u*}$ of $\mathfrak{B}^{u*}$ and this is isomorphic to the subinterpretation $\mathfrak{B}^{u+}_{\mathrm{bag}(G)}$ of the model $\mathfrak{B}^{u+}$ of $\mathcal{O}$. Now assume that $\{a_1, \ldots, a_k\} \cap \mathrm{dom}(\mathfrak{D}^u) = Z \neq \emptyset$. By Fact 4, for any guarded set F in $\mathfrak{B}^{u*}$ with $G' \cap Z \neq \emptyset$ there exists a guarded set F' in $\mathfrak{B}^{u*}$ such that there exists an isomorphism h from $\mathfrak{B}^{u*}_{|F}$ onto $\mathfrak{B}^{u*}_{|F'}$ that extends the canonical automorphism $\hat{h}_{t,G}$. The converse direction holds as well: let $Z' = \{a_1^\sim, \ldots, a_k^\sim\} \cap \mathrm{dom}(\mathfrak{D}^u)$. By Fact 4, for any guarded set F' in $\mathfrak{B}^{u*}$ with $F' \cap Z' \neq \emptyset$ there exists a guarded set F in $\mathfrak{B}^{u*}$ such that there exists an isomorphism h from $\mathfrak{B}^{u*}_{|F'}$ onto $\mathfrak{B}^{u*}_{|F}$ that extends the canonical automorphism $\hat{h}_{G,t}$.Thus $\mathfrak{B}^{u*} \models \psi(\vec{a})$ iff $\mathfrak{B}^{u+} \models \psi(\vec{a}^\sim)$ follows.

For sentences φ of the form $\forall x \psi(x)$ in $\mathcal{O}$, where $\psi(x)$ is an openGF formula of depth two using at most binary relations the argument is similar using the fact that guarded sets $\{a, b\}$ are either completely contained in $\mathrm{dom}(\mathfrak{D}^u)$ or contain at most one element from $\mathrm{dom}(\mathfrak{D}^u)$. This finishes the proof of Fact 5.

Finally we hook for any maximal guarded G in $\mathfrak{D}$ the interpretation $\mathfrak{B}^{u+}_{\mathrm{bag}(G)}$ to G in the original instance $\mathfrak{D}$ and obtain a forest model $\mathfrak{B}^+$. It is straightforward to prove that for any maximal guarded set G in $\mathfrak{D}$, any tuple $\vec{e}$ containing all elements of G, and the copy $\vec{f}$ of $\vec{e}$ in $\mathrm{bag}(G)$ there is a connected guarded bisimulation between $(\mathfrak{B}^{u*}, \vec{f})$ and $(\mathfrak{B}^+, \vec{e})$. It follows that $\mathfrak{B}^+$ is a model of $\mathcal{O}$ and $\mathfrak{D}$ and that $\mathfrak{B}^+ \not\models q(\vec{a})$ if $\mathfrak{B}^{u*} \not\models q(\vec{b})$.

Case 2. $\mathcal{O}$ is a $\mathrm{uGF}^-(1,=)$ ontology. In this case the construction is simpler as we do not modify $\mathfrak{B}$ further. There is no need to manipulate $\mathfrak{B}$ as we are in a fragment of depth one in which the outermost universal quantifier is guarded by an equality. Define $\mathfrak{B}^+$ by adding to $\mathfrak{D}$

- all atoms $R(a_1^\uparrow, \ldots, a_k^\uparrow)$ such that $R(a_1, \ldots, a_k) \in \mathfrak{B}$ and $a_1, \ldots, a_k \in \mathrm{dom}(\mathfrak{D}^u)$;
- for any $a \in \mathrm{dom}(\mathfrak{D})$ for a fixed copy a' of a in $\mathfrak{D}^u$ a copy $\mathfrak{B}_{a'}$ of $\mathfrak{B}$ that is hooked to $\mathfrak{D}$ at a by identifying a' and a.

Using Fact 1 and the condition that $\mathcal{O}$ is a $\mathrm{uGF}^-(1,=)$ ontology it is straightforward to prove that $\mathfrak{B}^+$ is a model of $\mathcal{O}$ and $\mathfrak{D}$. Also by Fact 1 and construction $\mathfrak{B}^+ \not\models q(\vec{a})$ if $\mathfrak{B} \not\models q(\vec{b})$.

We now assume that $\mathcal{O}$ is a $\mathrm{uGC}_2^-(1,=)$ or a $\mathcal{ALCHIF}$ ontology of depth 2. Let $\mathfrak{D}$ be an instance, G_0 be a maximal guarded set in $\mathfrak{D}$, $\vec{a}$ in G_0, $\vec{b}$ the copy of $\vec{a}$ in $\mathrm{bag}(G_0)$, and q an rAQ. We have to show that $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ implies $\mathcal{O}, \mathfrak{D}^u \models q(\vec{b})$. Observe that now we have to consider the uGC_2-unravelling rather than the uGF-unravelling of $\mathfrak{D}$. First we establish again the existence of certain automorphisms of $\mathfrak{D}^u$. In this case, however, they are *not* induced by automorphisms of the tree $(T(\mathfrak{D}), E)$ but are determined directly on the interpretation. Recall that for any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ the mapping $h_{t,t'}$ that sends every $e \in \mathrm{dom}(\mathrm{bag}(t))$ to the unique $f \in \mathrm{dom}(\mathrm{bag}(t'))$ with $e^\uparrow = f^\uparrow$ is an isomorphism from $\mathrm{bag}(t)$ to $\mathrm{bag}(t')$. Call $h_{t,t'}$ the *canonical isomorphism* from $\mathrm{bag}(t)$ onto $\mathrm{bag}(t')$. One can easily prove the existence of an *extended canonical automorphism* $\hat{h}_{t,t'}$ of $\mathfrak{D}^u$ that extends $h_{t,t'}$.

Fact 1. If $t, t' \in T(\mathfrak{D})$ such that $t \sim t'$ there is an automorphism $\hat{h}_{t,t'}$ of $\mathfrak{D}^u$ that extends $h_{t,t'}$.

Let $\mathfrak{B}$ be a materialization of $\mathcal{O}$ and $\mathfrak{D}^u$. Fact 1 entails the following.

Fact 2. For all t, t' with $t \sim t'$ and $\vec{e}$ in $\mathrm{dom}(\mathrm{bag}(t))$ and any rAQ $q(\vec{x})$ such that $\vec{x}$ has the same length as $\vec{e}$:

$$\mathfrak{B} \models q(\vec{e}) \quad \Leftrightarrow \quad \mathfrak{B} \models q(h_{t,t'}(\vec{e}))$$

We now distinguish two cases.

Case 3. $\mathcal{O}$ is a $\text{uGC}_2^-(1,=)$ ontology. This case is similar to Case 2. No further modification of $\mathfrak{B}$ is needed. We may assume that $\mathfrak{B}$ is obtained from $\mathfrak{D}^u$ by hooking cg-tree decomposable interpretations $\mathfrak{B}_c$ to c for every $c \in \text{dom}(\mathfrak{D}^u)$ and by adding atoms $R(c,d)$ to $\mathfrak{D}^u$ for distinct $c, d \in \text{dom}(\mathfrak{D}^u)$. Define $\mathfrak{B}^+$ by adding to $\mathfrak{D}$

- all atoms $R(a_1^\uparrow, \dots, a_k^\uparrow)$ such that $R(a_1, \dots, a_k) \in \mathfrak{B}$ and $a_1, \dots, a_k \in \text{dom}(\mathfrak{D}^u)$;
- for any $a \in \text{dom}(\mathfrak{D})$ for a fixed copy a' of a in $\mathfrak{D}^u$ a copy of $\mathfrak{B}_{a'}$ that is hooked to $\mathfrak{D}$ at a by identifying a' and a.

Using Fact 1 and the condition that $\mathcal{O}$ is a $\text{uGC}_2^-(1,=)$ ontology it is straightforward to prove that $\mathfrak{B}^+$ is a model of $\mathcal{O}$ and $\mathfrak{D}$. Also by Fact 1 and construction $\mathfrak{B}^+ \not\models q(\vec{a})$ if $\mathfrak{B} \not\models q(\vec{b})$.

Case 4. $\mathcal{O}$ is a $\mathcal{ALCHIF}$ ontology of depth 2. The proof that follows is similar to Case 1, but one cannot hook to any $\text{bag}(t)$ a copy of any rAQ from which there is a homomorphism into $\mathfrak{B}$ that is injective on the root bag of q and maps it into $\text{dom}(\text{bag}(t))$ as this can lead to violations of functionality. Two modifications are needed: firstly, we do not independently hook interpretations to bags $\text{bag}(t)$ with two elements in $\mathfrak{D}^u$. This is to avoid violations of functionality due to guarded sets G_1 and G_2 with $G_1 \cap G_2 = \{d\}$ when the interpretations we hook to G_1 and G_2 independently add an R-successor to d for a function R (this has already been done in Case 3). Secondly, we cannot hook arbitrarily many rAQs to bags as this will lead to violations of functionality as well.

We may again assume that $\mathfrak{B}$ is obtained from $\mathfrak{D}^u$ by hooking cg-tree decomposable interpretations $\mathfrak{B}_c$ to c for every $c \in \text{dom}(\mathfrak{D}^u)$ and by adding atoms $R(c,d)$ to $\mathfrak{D}^u$ for distinct $c, d \in \text{dom}(\mathfrak{D}^u)$. Observe that $G_{\mathfrak{B}_c} = \{\{a,b\} \mid R(a,b) \in \mathfrak{B}_c, a \neq b\}$ is an undirected tree. We call c its root and in this proof call such an interpretation a *tree interpretation with root c*. We have to modify $\mathfrak{D}^u$ to be able to uniformize. For any $c \in \text{dom}(\mathfrak{D}^u)$ we define the tree instance $\mathfrak{D}_c$ with root c as follows. Let $\mathfrak{D}_q$ be the instance corresponding to an rAQ $q = q(x) \leftarrow \phi$ with a single answer variable x and a single additional variable y such that there is an injective homomorphism h from $\mathfrak{D}_q$ to $\mathfrak{B}$ mapping x to some c in $\text{dom}(\mathfrak{D}^u)$ and such that neither $R(h(x), h(y)) \in \mathfrak{B}$ nor $R(h(y), h(x)) \in \mathfrak{B}$ for any R that is functional in $\mathcal{O}$. Then $\mathfrak{D}_c$ contains a copy of $\mathfrak{D}_q$ obtained by identifying the variable x with c. Set

$$\mathfrak{D}^{u+} = \{R(\vec{a}) \in \mathfrak{B} \mid \vec{a} \subseteq \text{dom}(\mathfrak{D}^u)\} \cup \bigcup_{c \in \text{dom}(\mathfrak{D}^u)} \mathfrak{D}_c.$$

The following properties of $\mathfrak{D}^{u+}$ follow directly from the definition and standard properties of $\mathcal{ALCHIF}$:

1. For any $c, d \in \text{dom}(\mathfrak{D}^u)$ with $c^\uparrow = d^\uparrow$ there is an isomorphism from $\mathfrak{D}_c$ onto $\mathfrak{D}_d$ mapping c to d;
2. there is a homomorphism from $\mathfrak{D}^{u+}$ into $\mathfrak{B}$ preserving $\text{dom}(\mathfrak{D}^u)$ and functionality. Thus, in particular, any materialization of $\mathfrak{D}^{u+}$ is a materialization of $\mathfrak{D}^u$ and for every CQ $q(\vec{x})$ and $\vec{d}$ in $\mathfrak{D}^u$ of the same length as $\vec{x}$:

$$\mathcal{O}, \mathfrak{D}^{u+} \models q(\vec{d}) \quad \Leftrightarrow \quad \mathcal{O}, \mathfrak{D}^u \models q(\vec{d})$$

Now take a materialization $\mathfrak{B}^{u+}$ of $\mathfrak{D}^{u+}$ and $\mathcal{O}$. Thus $\mathfrak{B}^{u+}$ is obtained from $\mathfrak{D}^{u+}$ by hooking tree-interpretations $\mathfrak{B}_c^{u+}$ that are models of $\mathfrak{D}_c$ to every $c \in \text{dom}(\mathfrak{D}^u)$. By Point 1 and Fact 1 and the properties of $\mathcal{ALCHIF}$ we have the following:

Fact 3. For any $c, d \in \text{dom}(\mathfrak{D}^u)$ with $c^\uparrow = d^\uparrow$, $c \in \text{dom}(\text{bag}(t))$, and $d \in \text{dom}(\text{bag}(t'))$ such that $t \sim t'$ the canonical automorphism $\hat{h}_{t,t'}$ extends to an isomorphism from $\mathfrak{B}^{u+}_{|\text{dom}(\mathfrak{D}_c)}$ onto $\mathfrak{B}^{u+}_{|\text{dom}(\mathfrak{D}_d)}$.

The following fact can now be proved by modifying in a straightforward way the proof of Fact 4 above.

Fact 4. For any $c, d \in \text{dom}(\mathfrak{D}^u)$ with $c^\uparrow = d^\uparrow$ and any $e \in \text{dom}(\mathfrak{B}_c^{u+})$ there exists an isomorphic embedding of $\mathfrak{B}^{u+}_{|\{c,e\}}$ into $\mathfrak{B}^{u+}_{|\text{dom}(\mathfrak{D}_d)}$.

Let for $c, d \in \text{dom}(\mathfrak{D}^u)$, $c \sim d$ if $c^\uparrow = d^\uparrow$. Fix for any equivalence class $[c] = \{d \mid d \sim c\}$ a unique $c_\sim \in [c]$. We now uniformize $\mathfrak{B}^{u+}$. For each $d \in \text{dom}(\mathfrak{D}^u)$ we hook at d to $\mathfrak{D}^u$ an isomorphic copy $\mathfrak{B}_d^{u*}$ of the interpretation $\mathfrak{B}_{c_\sim}^{u+}$ with $c_\sim^\uparrow = d^\uparrow$ and remove $\mathfrak{B}_d^{u+}$. Denote the resulting model by $\mathfrak{B}^{u*}$. $\mathfrak{B}^{u*}$ is uniform in the sense that for any $c \sim d$ the interpretations $\mathfrak{B}_c^{u*}$ hooked to c in $\mathfrak{D}^u$ and $\mathfrak{B}_d^{u*}$ hooked to d in $\mathfrak{D}^u$ are isomorphic. One can now prove similary to the proof of Fact 5 above the following:

Fact 5. $\mathfrak{B}^{u*}$ is a materialization of $\mathcal{O}$ and $\mathfrak{D}^u$.

It remains construct the materialization $\mathfrak{B}^+$ of $\mathfrak{D}$. To this end we hook to any $c \in \text{dom}(\mathfrak{D})$ the tree interpretation $\mathfrak{B}_d^{u*}$ with $d^\uparrow = c$. In addition we include in $\mathfrak{B}^+$ all $R(a_1^\uparrow, \dots, a_k^\uparrow)$ such that $R(a_1, \dots, a_k) \in \mathfrak{B}^{u+}$ and $a_1, \dots, a_k \in \text{dom}(\mathfrak{D}^u)$. Using Fact 5 one can show that $\mathfrak{B}^+$ is a model of $\mathcal{O}$ and $\mathfrak{D}$ such that $\mathfrak{B}^+ \not\models q(\vec{a})$ if $\mathfrak{B} \not\models q(\vec{b})$. □

G. PROOFS FOR SECTION 6

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

PROOF. We provide additional details of the proof for $\text{uGF}_2(1,=)$. Recall that $\mathcal{O}$ contains

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

where A and R range over symbols in $\text{sig}(\mathfrak{A})$ of the respective arity. We first show that $\text{coCSP}(\mathfrak{A})$ polynomially reduces to the query evaluation problem for $(\mathcal{O}, q \leftarrow N(x))$. Assume $\mathfrak{D}$ with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathfrak{A})$ is given and let

$$\mathfrak{D}' = \mathfrak{D} \cup \{R_a(d,d') \mid P_a(d) \in \mathfrak{A}\},$$

where the relations $P_a \in \text{sig}(\mathfrak{A})$ determine the precoloring and d' is a fresh labelled null for each $d \in \text{dom}(\mathfrak{D})$. We show that $\mathfrak{D} \rightarrow \mathfrak{A}$ iff $\mathcal{O}, \mathfrak{D}' \not\models q$. First let h be a homomorphism from $\mathfrak{D}$ to $\mathfrak{A}$. Define a model $\mathfrak{B}$ of $\mathfrak{D}'$ and $\mathcal{O}$ by adding to $\mathfrak{D}'$ for any $d \in \text{dom}(\mathfrak{D})$ with $h(d) = a$ and infinite chain

$$R_a(d_{0,d}, d_{1,d}), R_a(d_{1,d}, d_{2,d}), \dots$$

with $d_{0,d} = d$ and fresh labelled nulls $d_{i,d}$ for $i > 0$. Also add $R_a(d,d)$ to $\mathfrak{D}$ for all $d \in \text{dom}(\mathfrak{D})$ and all labelled nulls used in the chains. Using the definition of $\mathcal{O}$ it is not difficult to show that $\mathfrak{B}$ is a model of $\mathcal{O}$ and $\mathfrak{D}'$. Thus $\mathcal{O}, \mathfrak{D}' \not\models q$, as required. Now assume that $\mathcal{O}, \mathfrak{D}' \not\models q$. Then there is a model $\mathfrak{B}$ of $\mathcal{O}$ and $\mathfrak{D}'$ such that $\mathfrak{B} \not\models q$. Define a mapping h from $\mathfrak{D}$ to $\mathfrak{A}$ by setting $h(d) = a$ iff there exists d' with $d' \neq d$ and $R_a(d,d') \in \mathfrak{B}$. Using the definition of $\mathcal{O}$ it is not difficult to show that h is well defined and a homomorphism. This finishes the proof of the polynomial reduction of $\text{coCSP}(\mathfrak{A})$ to the query evaluation problem for $(\mathcal{O}, q \leftarrow N(x))$.

Now we show that for any rAQ q the query evaluation problem for $(\mathcal{O}, q)$ can be polynomially reduced to $\text{coCSP}(\mathfrak{A})$. We first

show that there is a polynomial reduction of the problem whether an instance $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}$ to CSP($\mathfrak{A}$). Assume $\mathfrak{D}$ is given. Let $\mathfrak{D}^\bullet$ be the sig($\mathfrak{A}$)-reduct of $\mathfrak{D}$ extended with $P_a(d)$ for any d with $R_a(d,d') \in \mathfrak{D}$ for some $d' \neq d$. Using the definition of $\mathcal{O}$ one can show that $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}$ iff $\mathfrak{D}^\bullet \to \mathfrak{A}$.

Now $\mathcal{O}, \mathfrak{D} \models q(\vec{d})$ iff $\mathfrak{D}$ is not consistent w.r.t. $\mathcal{O}$ or $\mathfrak{D}' \models q(\vec{d})$ where $\mathfrak{D}' = \mathfrak{D} \cup \{R_a(d,d) \mid a \in \text{dom}(\mathfrak{A}), d \in \text{dom}(\mathfrak{D})\}$. The latter problem is in PTIME. $\square$

H. PROOFS FOR SECTION 7

Theorem 10 (restated) *For the ontology languages $\text{uGF}_2^-(2,f)$ 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.*

As discussed in the main part of the paper, we prove Theorem 10 in two steps: we first construct an ontology $\mathcal{O}_{\text{cell}}$ that marks lower left corners of cells and then we construct an ontology $\mathcal{O}_{\mathfrak{P}}$ that marks the lower left corner of grids that represent a solution to a rectangle tiling problem $\mathfrak{P}$. We construct the ontologies in $\mathcal{ALCIF}_\ell$. Thus, in addition to $\mathcal{ALCI}$ concepts we use concepts of the form $(\leq 1R)$, $(= 1R)$, and $(\geq 2R)$. The proof is given using DL notation.

Marking the lower left corner of grid cells. Let X and Y be binary relations and X^-, Y^- their inverses in $\mathcal{ALCI}$. Using the sentences

$$\top \sqsubseteq (\leq 1Z)$$

for all $Z \in \{X, Y, X^-, Y^-\}$ we ensure that in any instance $\mathfrak{D}$ that is consistent w.r.t. our ontology the relations X and Y as well as their inverses X^- and Y^- are functional in $\mathfrak{D}$ in the sense that $R(d,d'), R(d,d'') \in \mathfrak{D}$ implies $d' = d''$ for all $R \in \{X, Y, X^-, Y^-\}$. For an instance $\mathfrak{D}$ and $d \in \text{dom}(\mathfrak{D})$ we write $\mathfrak{D} \models \text{cell}(d)$ if there exist d_1, d_2, d_3 with $X(d,d_1)$, $Y(d_1,d_3)$, $Y(d,d_2)$, $X(d_2,d_3) \in \mathfrak{D}$. Since X and Y are functional in $\mathfrak{D}$, $\mathfrak{D} \models \text{cell}(d)$ implies $d_3 = d_4$ if $X(d,d_1), Y(d_1,d_3)$, $X(d,d_2), Y(d_2,d_4) \in \mathfrak{D}$. As a marker for all d such that $\mathfrak{D} \models \text{cell}(d)$ we use the concept $(= 1P)$ for a binary relation P. For P and all binary relations R introduced below we add the inclusion $\top \sqsubseteq \exists R.\top$ to our ontology so that when building models one can only choose between having exactly one R-successor or at least two R-successors. To do the marking we use concepts $(= 1R_1)$ and $(= 1R_2)$ with binary relation symbols R_1, R_2 as 'second-order variables', ensure that all nodes in $\mathfrak{D}$ are contained in $(= 1R_1) \sqcup (= 1R_2)$, and then state (as a first attempt) that

$$\bigsqcup_{i=1,2} \exists X.\exists Y.(= 1R_i) \sqcap \exists Y.\exists X.(= 1R_i) \sqsubseteq (= 1P)$$

Clearly, if $\mathfrak{D} \models \text{cell}(d)$ then $\mathcal{O}, \mathfrak{D} \models (= 1P)(d)$ for the resulting ontology $\mathcal{O}$. Conversely, the idea is that if $\mathfrak{D} \not\models \text{cell}(d)$ and $X(d,d_1), Y(d,d_2), Y(d_1,d_3), X(d_2,d_4) \in \mathfrak{D}$ but $d_3 \neq d_4$, then one can extend $\mathfrak{D}$ by adding a single R_1-successor and two R_2-successors to d_3, a single R_2-successor and two R_1-successors to d_4, and two P-successors to d and thus obtain a model $\mathfrak{B}$ of $\mathcal{O}$ and $\mathfrak{D}$ in which $d \notin (= 1P)^{\mathfrak{B}}$, see Figure 2. In general, however, this does not work and the latter sentence has depth 3. The depth issue is easily resolved by introducing auxiliary binary relation symbols R_i^X, R_i^Y, R_i^{XY} and R_i^{YX}, $i = 1, 2$, and replacing concepts such as $\exists X.\exists Y.(= 1R_i)$ by $(= 1R_i^{XY})$ and the sentences

$$(= 1R_i^{XY}) \equiv \exists X.(= 1R_i^Y) \text{ and } (= 1R_i^Y) \equiv \exists Y.(= 1R_i)$$

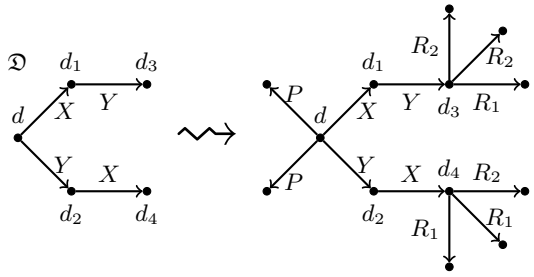


Figure 2: $\mathfrak{D} \not\models$ **cell**$(d) \Rightarrow \mathcal{O}, \mathfrak{D} \not\models (= 1P)(d)$

Details are given below. Resolving the first issue is more involved. There are two reasons why the converse does not hold. First, we might have $X(d,d_1)$, $Y(d_1,d_3)$, $X(d,d_2), Y(d_2,d_4) \in \mathfrak{D}$ with $d_3 \neq d_4$ but both d_3 and d_4 have already two R_2-successors in $\mathfrak{D}$. Then the marker $(= 1P)$ is entailed without the cell being closed at d (i.e, without $\mathfrak{D} \models \text{cell}(d)$). Second, we might have an odd cycle of mutually distinct $e_0, e_1, \ldots, e_n \in \mathfrak{D}$ such that each e_i reaches $e_{(i+1) \bmod n+1}$ via a Y^-X^-YX-path in $\mathfrak{D}$, for $i = 0, 1, \ldots, n$. Figure 3 illustrates this for $n = 2$. Then, since in at least two neighbouring $e_i, e_{(i+1) \bmod n+1}$ the same concept $(= 1R_i)$ is enforced, the marker $(= 1P)$ is enforced at some node d from which e_i and $e_{(i+1) \bmod 3}$ are reachable along XY and YX-paths, respectively, without satisfying cell(d). We resolve both problems by enforcing that $\mathfrak{D}$ is not consistent w.r.t. our ontology if such constellations appear.

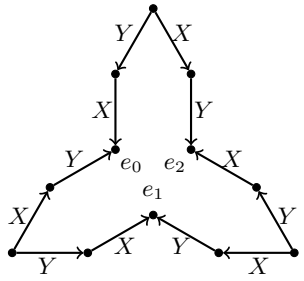


Figure 3: $\mathfrak{D} \models (= 1P)(d) \not\Rightarrow \mathfrak{D} \models$ **cell**(d)

In detail, we construct an ontology $\mathcal{O}_{\text{cell}}$ that uses in addition to X, Y, X^-, Y^- the set AUX_{cell} of binary relations P, R_i, R_i^W, where $i \in \{1,2\}$ and W ranges over a set of words over the alphabet $\{X, Y, X^-, Y^-\}$ we define below. The R_i^W serve as auxiliary symbols to avoid sentences of depth larger than two. No unary relations are used. To ensure that CQ-evaluation is Datalog$^{\neq}$-rewritable w.r.t. $\mathcal{O}_{\text{cell}}$ we include in $\mathcal{O}_{\text{cell}}$ the concept inclusions

$$\top \sqsubseteq \exists Q.\top$$

for all binary relations $Q \in \text{AUX}_{\text{cell}}$. If an instance $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}_{\text{cell}}$, then its materialization adds a certain number of Q-successors to any $d \in \text{dom}(\mathfrak{D})$ to satisfy $\top \sqsubseteq \exists Q.\top$ for $Q \in \text{AUX}_{\text{cell}}$. The remaining sentences in $\mathcal{O}_{\text{cell}}$ only influence the number of Q-successors that have to be added and thus do not influence the certain answers to CQs. In fact, we will have the following equivalence

$$\mathcal{O}_{\text{cell}}, \mathfrak{D} \models q(\vec{d}) \text{ iff } \{\top \sqsubseteq \exists Q.\top \mid Q \in \text{AUX}_{\text{cell}}\}, \mathfrak{D} \models q(\vec{d})$$

for any CQ q and $\mathfrak{D}$ that is consistent w.r.t. $\mathcal{O}_{\text{cell}}$. Define for any non-empty word W over $\{X, Y, X^-, Y^-\}$ the set $\exists^W(= 1R_i)$ of

sentences inductively by setting for $Z \in \{X, Y, X^-, Y^-\}$:

$$\begin{aligned} \exists^Z(=1R_i) &= \{(=1R_i^Z) \equiv \exists Z.(=1R_i)\} \\ \exists^{ZW}(=1R_i) &= \{(=1R_i^{ZW}) \equiv \exists Z.(=1R_i^W)\} \\ &\quad \cup \exists^W(=1R_i) \end{aligned}$$

Thus, $\exists^W(=1R_i)$ states that the unique d' reachable from d along a W-path has exactly one R_i-successor iff d has exactly one R_i^W-successor. Now $\mathcal{O}_{\text{cell}}$ is defined as follows.

1. Functionality of X, Y, X^- and Y^- is stated using
$$\top \sqsubseteq (\leq 1Z)$$
for X, Y, X^-, Y^-.
2. All nodes have exactly one R_1 successor or exactly one R_2-successor:
$$\top \sqsubseteq (=1R_1) \sqcup (=1R_2)$$
3. If all nodes reachable along an XY-path and a YX-path have exactly one R_1 and exactly one R_2-successor, then the marker $(=1P)$ is set:
$$\prod_{i=1,2} (=1R_i^{XY}) \sqcap (=1R_i^{YX}) \sqsubseteq (=1P)$$
4. For $i = 1, 2$, the concept $(=1R_i)$ is true at least at every third node on the cycles in $\mathfrak{D}$ introduced above:
$$(=1R_j^{CC}) \sqsubseteq (=1R_i) \sqcup (=1R_i^C) \sqcup (=1R_i^{CC})$$
for $\{i, j\} = \{1, 2\}$ and $C = X^-Y^-XY$
5. If $(=1R_1)$ and $(=1R_2)$ are both true in a node in $\mathfrak{D}$ then they are both true in all 'neighbouring' nodes in $\mathfrak{D}$:
$$(=1R_1^{X^-Y^-XY}) \sqcap (=1R_2^{X^-Y^-XY}) \sqsubseteq R^{12}$$
$$(=1R_1^{Y^-X^-YX}) \sqcap (=1R_2^{Y^-X^-YX}) \sqsubseteq R^{12}$$
for $R^{12} := (=1R_1) \sqcap (=1R_2)$
6. The auxiliary sentences $\exists^W(=1R_i)$ for all relations R_i^W used above.

Lemma 11 *The ontology $\mathcal{O}_{cell}$ has the following properties for all instances $\mathfrak{D}$:*

1. *for all $d \in dom(\mathfrak{D})$: $\mathcal{O}_{cell}, \mathfrak{D} \models (=1P)(d)$ iff $\mathfrak{D}$ is not consistent w.r.t. $\mathcal{O}_{cell}$ or $\mathfrak{D} \models cell(d)$; moreover, if $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}_{cell}$, then there exists a materialization $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}_{cell}$ such that $d \in (=1P)^{\mathfrak{B}}$ iff $d \in dom(\mathfrak{B})$ and $\mathfrak{D} \models cell(d)$;*
2. *If all binary relations are functional in $\mathfrak{D}$, then $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}_{cell}$;*
3. *CQ-evaluation w.r.t $\mathcal{O}_{cell}$ is Datalog$^{\neq}$-rewritable.*

PROOF. We first derive a necessary and sufficient condition for consistency of instances $\mathfrak{D}$ w.r.t. $\mathcal{O}_{\text{cell}}$. Lemma 11 then follows in a straightforward way. It is easy to see that if any of the following conditions is not satisfied, then $\mathfrak{D}$ is not consistent w.r.t. $\mathcal{O}_{\text{cell}}$:

- all X, Y, X^-, Y^- are functional in $\mathfrak{D}$;
- $\mathfrak{D}$ is consistent w.r.t. the sentences $\exists^W(=1R_i)$ in $\mathcal{O}_{\text{cell}}$;
- if $\mathfrak{D} \models \text{cell}(d)$, then d has at most one P-successor in $\mathfrak{D}$.

We thus assume in what follows that all three conditions are satisfied. Clearly, they can be encoded in Datalog$^{\neq}$. Moreover, by Point 2 we can assume that $\mathfrak{D}$ is saturated for the sentences $\exists^W(=1R)$ in the sense that if $(=1R^{ZW}) \equiv \exists Z.(=1R^W) \in \mathcal{O}_{\text{cell}}$ then for any $Z(d, d') \in \mathfrak{D}$ the following holds: d has at least two R^{ZW}-successors in $\mathfrak{D}$ iff d' has at least two R^W-successors in $\mathfrak{D}$. Now let $e_1 \leq e_2$ iff there are $X(d, d_1), Y(d_1, e_1), Y(d, d_2), X(d_2, e_2) \in \mathfrak{D}$. Let $e_1 \sim e_2$ iff $e_1 \leq e_2$ or $e_2 \leq e_1$ and let $\sim^*$ be the smallest equivalence relation containing $\sim$. For any equivalence class E w.r.t. $\sim^*$ either

- E is of the form $e_0 \leq \cdots \leq e_n$ with $e_i \neq e_j$ for all $i \neq j$, or
- E is a cycle $e_0 \leq \cdots \leq e_n$ with $e_i = e_j$ iff $\{i, j\} = \{0, n\}$ for all $i \neq j$.

Thus, if E is not a singleton $\{e\}$ with $e \leq e$, we can partition E into two sets E_1 and E_2 (with one of them possibly empty) such that

(†) there are no three $e \leq e' \leq e''$ in the same E_i.

Now set for any equivalence class E and $\{i, j\} = \{1, 2\}$,

$$E'_j = \{d \in E \mid \mathfrak{D} \models (\geq 2R_i)(d)\}$$

Claim 1. $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}_{\text{cell}}$ iff the following conditions hold for all equivalence classes E:

(a) if $E = \{e\}$ with $e \leq e$ then $e \notin E'_1 \cup E'_2$;

(b) otherwise, there exists a partition E_1, E_2 of E with $E_i \supseteq E'_i$ satisfying (†).

Moreover, if (a) and (b) hold, then a materialization $\mathfrak{B}$ satisfying the conditions of Lemma 11 (1) exists.

($\Rightarrow$) First assume that Point (a) does not hold for some $E = \{e\}$ with $e \leq e$. Then $\mathfrak{D}$ is not consistent w.r.t. $\mathcal{O}_{\text{cell}}$ by the axioms given under (2) and (4) since it is not possible to satisfy $(=1R_i)$ in e if $e \in E'_j$ $(i \neq j)$. Now assume that (b) holds. So there exists E that has either at least two elements or $e \not\leq e$ if $E = \{e\}$ but there exists no partition E_1, E_2 of E with $E_i \supseteq E'_i$ satisfying (†). Then the axioms under (4) cannot be satisfied without having at least one node in E that is in both $(=1R_1)$ and $(=1R_2)$. But then by the axioms under (5) all nodes in E are in $(=1R_1)$ and in $(=1R_2)$ which implies that $E'_1 = E'_2 = \emptyset$. This contradicts our assumption that there is no partition E_1, E_2 of E with $E_i \supseteq E'_i$ satisfying (†).

($\Leftarrow$) Assume (a) and (b) hold for every equivalence class E. For $E = \{e\}$ with $e \leq e$ we can thus construct the relevant part of a model $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}_{\text{cell}}$ such that e has exactly one R_i-successor for $i = 1, 2$ and also exactly one P-successor. Thus axiom (2) is satisfied. All e that are not members of such an equivalence class are given at least two P-successors. For any equivalence class with at least two members or $E = \{e\}$ with $e \not\leq e$ we can construct the relevant part of $\mathfrak{B}$ such that each $d \in E_i$ has exactly one R_i-successor and each $d \in E \setminus E_i$ has at least two R_i-successors. As E_1 and E_2 are mutually disjoint, the axioms under (5) are satisfied. As (†) is satisfied, the axioms under (4) are satisfied. As $E_1 \cup E_2$ contains E, the axioms under (2) are satisfied.

The conditions (a) and (b) can be encoded in a Datalog$^{\neq}$ program in a straightforward way and thus there is a Datalog$^{\neq}$-program checking consistency of an instance $\mathfrak{D}$ w.r.t. $\mathcal{O}_{\text{cell}}$. Datalog$^{\neq}$-rewritability of CQ-evaluation w.r.t. $\mathcal{O}_{\text{cell}}$ now follows from the observation that

$$\mathcal{O}_{\text{cell}}, \mathfrak{D} \models q(\vec{d}) \text{ iff } \{\top \sqsubseteq \exists Q.\top \mid Q \in \text{AUX}_{\text{cell}}\}, \mathfrak{D} \models q(\vec{d})$$

for any CQ q, $\mathfrak{D}$ that is consistent w.r.t. $\mathcal{O}_{\text{cell}}$, and any $\vec{d}$ in $\mathfrak{D}$. $\square$

This finishes the construction and analysis of $\mathcal{O}_{\text{cell}}$.

Marking the lower left corner of grids. We now encode the rectangle tiling problem. Recall that an instance of the *finite rectangle tiling problem* is given by a triple $\mathfrak{P} = (\mathfrak{T}, H, V)$ with $\mathfrak{T}$ a non-empty, finite set of *tile types* including an *initial tile* T_{init} to be placed on the lower left corner and nowhere else and a *final tile* T_{final} to be placed on the upper right corner and nowhere else, $H \subseteq \mathfrak{T} \times \mathfrak{T}$ a *horizontal matching relation*, and $V \subseteq \mathfrak{T} \times \mathfrak{T}$ a *vertical matching relation*. A *tiling* for $(\mathfrak{T}, H, V)$ is a map $f : \{0, \dots, n\} \times \{0, \dots, m\} \to \mathfrak{T}$ such that $n, m \geq 0$, $f(0,0) = T_{\mathsf{init}}$, $f(n,m) = T_{\mathsf{final}}$, $(f(i,j), f(i+1,j)) \in H$ for all $i < n$, and $(f(i,j), f(i,j+1)) \in V$ for all $i < m$. We say that $\mathfrak{P}$ *admits a tiling* if there exists a map f that is a tiling for $\mathfrak{P}$. It is undecidable whether an instance of the finite rectangle tiling problem admits a tiling.

Now let $\mathfrak{P} = (\mathfrak{T}, H, V)$ with $\mathfrak{T} = \{T_1, \dots, T_p\}$. We regard the tile types in $\mathfrak{T}$ as unary relations and take the binary relations symbols X, Y, X^-, Y^- from above and an additional set AUX_{grid} of binary relations F, F^X, F^Y, U, R, L, D, and A. The ontology $\mathcal{O}_{\mathfrak{P}}$ is defined by taking $\mathcal{O}_{\text{cell}}$ and adding the sentences

$$\top \sqsubseteq \exists Q.\top$$

for all $Q \in \text{AUX}_{\text{grid}}$ as well as all sentences in Figure 4 to it, where (T_i, T_j, T_ℓ) range over all triples from $\mathfrak{T}$ such that $(T_i, T_j) \in H$ and $(T_i, T_\ell) \in V$:

We discuss the intuition behind the sentences of $\mathcal{O}_{\mathfrak{P}}$. The relations X and Y are used to represent horizontal and vertical adjacency of points in a rectangle. The concepts $(= 1X)$ of $\mathcal{O}_{\mathfrak{P}}$ serve the following puroposes:

- $(= 1U)$, $(= 1R)$, $(= 1L)$, and $(= 1D)$ mark the upper, right, left, and lower ('down') border of the rectangle.
- The concept $(= 1F)$ is propagated through the grid from the upper right corner where T_{final} holds to the lower left one where T_{initial} holds, ensuring that every position of the grid is labeled with at least one tile type, that the horizontal and vertical matching conditions are satisfied, and that the grid cells are closed (indicated by $(= 1P)$ from the ontology $\mathcal{O}_{\text{cell}}$).
- The relations F^X and F^Y are used to avoid depth 3 sentences in the same way as the relations R_i^W are used to avoid such sentences in the construction of $\mathcal{O}_{\text{cell}}$.
- Finally, when the lower left corner of the grid is reached, the concept $(= 1A)$ is set as a marker.

We write $\mathfrak{D} \models \text{grid}(d)$ if there is a tiling f for $\mathfrak{P}$ with domain $\{0, \dots, n\} \times \{0, \dots, m\}$ and a mapping $\gamma : \{0, \dots, n\} \times \{0, \dots, m\} \to \text{dom}(\mathfrak{D})$ with $\gamma(0,0) = d$ such that

- for all $j < n$, $k \leq m$: $T(\gamma(j,k)) \in \mathfrak{D}$ iff $T = f(j,k)$;
- for all $b_1, b_2 \in \text{dom}(\mathfrak{D})$: $X(b_1, b_2) \in \mathfrak{D}$ iff there are $j < n$, $k \leq m$ such that $(b_1, b_2) = (\gamma(j,k), \gamma(j+1,k))$;
- for all $b_1, b_2 \in \text{dom}(\mathfrak{D})$: $Y(b_1, b_2) \in \mathfrak{D}$ iff there are $j \leq n$, $k < m$ such that $(b_1, b_2) = (\gamma(j,k), \gamma(j,k+1))$.
- g is closed: if $d \in \text{ran}(\gamma)$ and $Z(d, d') \in \mathfrak{D}$ for some $Z \in \{X, Y, X^-, Y^-\}$, then $d' \in \text{ran}(\gamma)$.

We then call d *the root of the* $n \times m$*-grid with witness function* γ *for* $\mathfrak{P}$. The following result can now be proved using Lemma 11.

Lemma 12 *The ontology $\mathcal{O}_{\mathfrak{P}}$ has the following properties for all instances $\mathfrak{D}$:*

1. *for all $d \in \text{dom}(\mathfrak{D})$: $\mathcal{O}_{\mathfrak{P}}, \mathfrak{D} \models (= 1A)(d)$ iff $\mathfrak{D}$ is not consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$ or $\mathfrak{D} \models \text{grid}(d)$; moreover, if $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$, then there exists a materialization $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}_{\mathfrak{P}}$ such that $d \in (= 1A)^{\mathfrak{B}}$ iff $d \in \text{dom}(\mathfrak{B})$ and $\mathfrak{D} \models \text{grid}(d)$;*
2. *If $\mathfrak{D} \models \text{grid}(d)$ with witness γ such that $\text{dom}(\mathfrak{D}) = \text{ran}(\gamma)$, and all relations are functional in $\mathfrak{D}$ then $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$;*
3. *CQ-evaluation w.r.t $\mathcal{O}_{\mathfrak{P}}$ is Datalog$^{\neq}$-rewritable.*

We now use Lemma 12 to prove the undecidability result. Let $\mathcal{O} = \mathcal{O}_{\mathfrak{P}} \cup \{(= 1A) \sqsubseteq B_1 \sqcup B_2\}$, where B_1 and B_2 are unary relations.

Lemma 13 *(1) If $\mathfrak{P}$ admits a tiling, then $\mathcal{O}$ is not materializable and CQ-evaluation w.r.t. $\mathcal{O}$ is* coNP*-hard.*

(2) If $\mathfrak{P}$ does not admit a tiling, then CQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable.

PROOF. (1) Consider a tiling f for $\mathfrak{P}$ with domain $\{0, \dots, n\} \times \{0, \dots, m\}$. Regard the pairs in $\{0, \dots, n\} \times \{0, \dots, m\}$ as constants. Let $\mathfrak{D}$ contain

$$X((i,j),(i+1,j)),$$

for $i < n$ and $j \leq m$,

$$Y((i,j),Y(i,j+1))$$

for $i \leq n$ and $j < m$, and

$$T(i,j)$$

for $f(i,j) = T$ for $i \leq n$ and $j \leq m$. Then $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$ and $\mathcal{O}, \mathfrak{D} \models (= 1A)(0,0)$, by Lemma 12. Thus $\mathcal{O}_{\mathfrak{P}}, \mathfrak{D} \models B_1(0,0) \vee B_2(0,0)$ but $\mathcal{O}, \mathfrak{D} \not\models B_1(0,0)$ and $\mathcal{O}, \mathfrak{D} \not\models B_2(0,0)$. Thus $\mathcal{O}$ is not materializable and so CQ-evaluation is coNP-hard.

(2) Assume $\mathfrak{P}$ does not admit a tiling. Any instance $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}$ iff it is consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$ and, by Lemma 12, if $\mathcal{O}_{\mathfrak{P}}, \mathfrak{D} \models (= 1A)(d)$ for some $d \in \text{dom}(\mathfrak{D})$, then $\mathfrak{D}$ is not consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$. Thus, $\mathcal{O}, \mathfrak{D} \models q(\vec{d})$ iff $\mathcal{O}_{\mathfrak{P}}, \mathfrak{D} \models q(\vec{d})$ holds for every CQ q and all $\vec{d}$. Thus CQ-evaluation w.r.t. $\mathcal{O}$ is Datalog$^{\neq}$-rewritable by Lemma 12. □

Lemma 13 implies Theorem 10 for $\mathcal{ALCIF}_\ell$ ontologies of depth 2. For $\text{uGF}_2^-(2,f)$ we modify the construction of $\mathcal{O}_{\text{cell}}$ and $\mathcal{O}_{\mathfrak{P}}$ as follows:

- The relations X, Y, X^-, Y^- are defined as functions and it is stated that X^- and Y^- are the inverse of X and Y, respectively.
- For any relation symbol R in $\mathcal{O}_{\mathfrak{P}}$ distinct from X, Y, X^-, Y^- we introduce a function F, state $\forall x F(x,x)$, and replace

  $$\top \sqsubseteq \exists R.\top$$

  by

  $$\forall x \exists y (R(x,y) \wedge F(x,y)).$$

- We replace all occurrences of $(= 1R)$ for $R \notin \{X, Y, X^-, Y^-\}$ in $\mathcal{O}_{\mathfrak{P}}$ by

  $$\neg \exists y (R(x,y) \wedge \neg F(x,y))$$

Now Lemma 11 and Lemma 12 still hold for the resulting ontologies $\mathcal{O}_{\text{cell}}$ and $\mathcal{O}_{\mathfrak{P}}$ if $(= 1P)$ and $(= 1A)$ are replaced by $\neg \exists y (P(x,y) \wedge \neg F(x,y))$ and $\neg \exists y (A(x,y) \wedge \neg F(x,y))$, respectively.

We now come to the proof of Theorem 11. We first give a more detailed definition of the run fitting problem.

$$
\begin{aligned}
T_{\mathsf{final}} &\sqsubseteq (=1F) \sqcap (=1U) \sqcap (=1R) \\
\exists X.((=1U) \sqcap (=1F) \sqcap T_j) \sqcap T_i &\sqsubseteq (=1U) \sqcap (=1F) \\
\exists Y.((=1R) \sqcap (=1F) \sqcap T_\ell) \sqcap T_i &\sqsubseteq (=1R) \sqcap (=1F) \\
\exists Y.(=1F) &\sqsubseteq (=1F^Y) \\
\exists X.(=1F) &\sqsubseteq (=1F^X) \\
\exists X.(T_j \sqcap (=1F) \sqcap (=1F^Y)) \sqcap & \\
\exists Y.(T_\ell \sqcap (=1F) \sqcap (=1F^X)) \sqcap (=1P) \sqcap T_i &\sqsubseteq (=1F) \\
(=1F) \sqcap T_{\mathsf{init}} &\sqsubseteq (=1A) \sqcap (=1D) \sqcap (=1L) \\
\bigsqcup_{1 \le s < t \le p} T_s \sqcap T_t &\sqsubseteq \bot
\end{aligned}
$$

$$
(=1U) \sqsubseteq \forall Y.\bot \quad (=1R) \sqsubseteq \forall X.\bot \quad (=1U) \sqsubseteq \forall X.(=1U) \quad (=1R) \sqsubseteq \forall Y.(=1R)
$$

$$
(=1D) \sqsubseteq \forall Y^-.\bot \quad (=1L) \sqsubseteq \forall X^-.\bot \quad (=1D) \sqsubseteq \forall X.(=1D) \quad (=1L) \sqsubseteq \forall Y.(=1L)
$$

Figure 4: Additional Axioms of $\mathcal{O}_{\mathfrak{P}}$

We consider *non-deterministic Turing machines* (*TMs*, for short). A TM M is represented by a tuple $(Q, \Sigma, \Delta, q_0, q_a)$, where Q is a finite set of states, Σ is a finite alphabet, $\Delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{L, R\}$ is the transition relation, and $q_0, q_a \in Q$ are the start state and accepting state, respectively. The tape of M is assumed to be one-sided infinite, that is, it has a left-most cell and extends infinitely to the right. The set of strings accepted by M is denoted by $L(M)$. A *configuration* of M is represented by a string vqw, where q is the state, v is the inscription of the tape to the left of the head, and w is the inscription of the tape to the right of the head in the configuration (as usual, we omit all but possibly a finite number of trailing blanks). The configuration is *accepting* if $q = q_a$. A *run* of M is represented by a finite sequence $\gamma_0, \dots, \gamma_n$ of configurations of M with $|\gamma_0| = \dots = |\gamma_n|$. We assume that the accepting state has no successor states. A run is *accepting* if its last configuration is accepting.

Definition 7 Let $M = (Q, \Sigma, \Gamma, \Delta, q_0, q_a)$ be a TM.

- A *partial configuration* of M is a string $\tilde{\gamma}$ over $Q \cup \Sigma \cup \{\star\}$ such that there is at most one $i \in \{1, \dots, n\}$ with $\tilde{\gamma}[i] \in Q$. Here, $\gamma[i]$ denotes the symbol that occurs at the i-th position of γ. A configuration γ *matches* $\tilde{\gamma}$ if $|\gamma| = |\tilde{\gamma}|$ and for each $i \in \{1, \dots, n\}$ with $\tilde{\gamma}[i] \neq \star$ we have $\gamma[i] = \tilde{\gamma}[i]$.
- A *partial run* of M is a sequence $\tilde{\gamma} = (\tilde{\gamma}_0, \tilde{\gamma}_1, \dots, \tilde{\gamma}_m)$ of partial configurations $\tilde{\gamma}_i$ of M such that $\tilde{\gamma}_0$ starts with q_0, has no other occurrence of a state $q \in Q$, and $|\tilde{\gamma}_0| = \dots = |\tilde{\gamma}_m|$. A run $\gamma_0, \gamma_1, \dots, \gamma_n$ of M matches $\tilde{\gamma}$ if $m = n$ and γ_i matches $\tilde{\gamma}_i$, for each $i \in \{0, 1, \dots, m\}$.

Definition 8 *The* run fitting problem *for a TM M, denoted by* $\mathrm{RF}(M)$*, is defined as follows: Given a partial run $\tilde{\gamma}$ of M, decide whether there is an accepting run of M that matches $\tilde{\gamma}$.*

It is easy to see that $\mathrm{RF}(M)$ is in NP for every TM M. The following theorem states that the complexity of $\mathrm{RF}(M)$ may be intermediate between PTIME and NP-complete, provided PTIME $\neq$ NP.

Theorem 12 (restated) *If* PTIME $\neq$ NP*, then there is a TM whose run fitting problem is neither in* PTIME *nor* NP*-complete.*

To prove Theorem 12, we now construct such a TM. The construction is a modification of the construction from Impagliazzo's version of the proof of Ladner's Theorem [38], as presented in [2, Theorem 3.3].

Fix a polynomial-time TM M_{SAT} for SAT. For a monotone polynomial-time computable function $H\colon \mathbb{N} \to \mathbb{N}$ to be specified later, let M_H be a polynomial-time TM that works as follows on a given input v:

1. Check that there is an integer $n \geq 0$ such that v is the unary representation of $n^{H(n)}$ (i.e., $v = 1^{n^{H(n)}}$). If such an n does not exist, then reject v.
2. Guess an input w of length n for M_{SAT}.
3. Generate the initial configuration γ of M_{SAT} on input w.
4. Run M_{SAT} from γ, and accept v iff M_{SAT} accepts w.

We refer to the first three steps as the *initialization phase*.

We now define the function $H\colon \mathbb{N} \to \mathbb{N}$. Fix a polynomial time computable enumeration $M_0, M_1, M_2, \dots$ of deterministic TMs such that all runs of M_i on inputs of length n terminate after at most $i \cdot n^i$ steps, and for each problem A in PTIME there are infinitely many i such that $L(M_i)$ is in A.[3] Then, $H(n)$ is defined as

$$
\begin{aligned}
\min \{ i < \log\log n \mid & \text{ for all strings } z \text{ of length} \leq \log n, \\
& M_i \text{ accepts } z \text{ iff } z \in \mathrm{RF}(M_H) \},
\end{aligned}
$$

or as $\log\log n$ if the minimum does not exist. It is not hard to see that H is well-defined, and that there is a deterministic polynomial-time Turing machine that, given a positive integer n in unary, outputs $H(n)$. For details, we refer to [2].

This finishes the construction of M_H. Lemma 15 below shows that $\mathrm{RF}(M_H)$ has the desired properties, namely that $\mathrm{RF}(M_H)$ is neither in PTIME nor NP-complete, unless PTIME $=$ NP. It uses the following auxiliary lemma.

Lemma 14

- *If* $\mathrm{RF}(M_H)$ *is in* PTIME*, then* $H(n) = O(1)$.

[3] It is easy to construct a deterministic polynomial-time Turing machine that, given an integer $i \geq 0$, outputs a deterministic Turing machine M_i such that the sequence $(M_i)_{i \geq 0}$ has the desired properties. For instance, let M_i' be the i-th deterministic Turing machine in lexicographic order under some string encoding of Turing machines, and add a clock to M_i' that stops the computation of M_i' after at most $i \cdot n^i$ steps (and rejects if M_i' did not accept yet).

- *If* $\mathrm{RF}(M_H)$ *is not in* PTIME*, then* $\lim_{n\to\infty} H(n) = \infty$.

PROOF. The proof is as in [2]. We provide a proof for the sake of completeness.

Assume first that $\mathrm{RF}(M_H)$ is in PTIME. Then, there is an index i such that $L(M_i) = \mathrm{RF}(M_H)$. Now, for all $n > 2^{2^i}$, we have $i < \log\log n$ and thus $H(n) \leq i$ by the definition of H. It follows that $H(n) \leq \max\{H(m) \mid m \leq 2^{2^i} + 1\}$, and therefore $H(n) = O(1)$.

Next assume that $\mathrm{RF}(M_H)$ is not in PTIME. For a contradiction, suppose that $\lim_{n\to\infty} H(n) \neq \infty$. Since H is monotone, this means that there are integers $n_0, i \geq 0$ such that $H(n) = i$ for all integers $n \geq n_0$. Let $n \geq n_0$. By the definition of H, we have that M_i agrees with $\mathrm{RF}(M_H)$ on all strings of length at most $\log n$. Since this holds for all $n \geq n_0$, we conclude that M_i decides $\mathrm{RF}(M_H)$. But then, $\mathrm{RF}(M_H)$ is in PTIME, which contradicts our initial assumption that $\mathrm{RF}(M_H)$ is not in PTIME. □

We now prove the main lemma, which concludes the proof of Theorem 12.

Lemma 15 *If* PTIME $\neq$ NP*, then* $\mathrm{RF}(M_H)$ *is neither in* PTIME *nor* NP*-complete.*

PROOF. *"*$\mathrm{RF}(M_H)$ *is not in* PTIME*":* For a contradiction, suppose that $\mathrm{RF}(M_H)$ is in PTIME. By Lemma 14, there is a constant $c \geq 0$ such that for all integers $n \geq 0$ we have $H(n) \leq c$. Suppose that on inputs of length n, M_{SAT} makes at most $p(n) := n^k + k$ steps. Then, the following is a polynomial-time many-one reduction from SAT to $\mathrm{RF}(M_H)$, implying PTIME = NP, and therefore contradicting the lemma's assumption.

Given an input x of length n for M_{SAT}:

1. Compute $h := H(n)$ and $w := 1^{n^h}$.
2. Output the partial run $\tilde{\gamma}_0, \ldots, \tilde{\gamma}_{i+p(n)}$ of M_H such that:
   - $\tilde{\gamma}_0, \ldots, \tilde{\gamma}_i$ corresponds to the initialization phase of M_H on input w that generates the start configuration of M_{SAT} on input x. In particular, $\tilde{\gamma}_0, \ldots, \tilde{\gamma}_i$ are complete configurations of M_H, and $\tilde{\gamma}_0 = q_0 w$ and $\tilde{\gamma}_i = q_0' x$, where q_0 and q_0' are the start states of M_H and M_{SAT}, respectively;
   - $\tilde{\gamma}_{i+1}, \ldots, \tilde{\gamma}_{i+p(n)}$ consist entirely of stars.

Note that the partial run $\tilde{\gamma}_0, \ldots, \tilde{\gamma}_{i+p(n)}$ can be easily computed by simulating the initialization phase M_H on input w, where in step 2 of the initialization phase we "guess" the input string x given as input to the reduction. Then, we pad the sequence of configurations corresponding to the initialization phase by $p(n)$ partial configurations, each consisting of exactly $p(n)$ stars.

"$\mathrm{RF}(M_H)$ *is not* NP*-complete":* Suppose, to the contrary, that $\mathrm{RF}(M_H)$ is NP-complete. Then there is a polynomial-time many-one reduction f from SAT to $\mathrm{RF}(M_H)$. Using f, we construct a polynomial-time many-one reduction g from SAT to SAT such that for all sufficiently large strings x we have $|g(x)| < |x|$. This implies that SAT can be solved in polynomial time, and contradicts PTIME $\neq$ NP.

Consider an input x for SAT. Since f is a many-one reduction from SAT to $\mathrm{RF}(M_H)$, we have

$$f(x) = \tilde{\gamma}_0 \# \tilde{\gamma}_1 \# \cdots \# \tilde{\gamma}_m \tag{4}$$

for some partial run

$$\tilde{\gamma} := (\tilde{\gamma}_0, \tilde{\gamma}_1, \ldots, \tilde{\gamma}_m)$$

of M_H. Moreover, $x \in$ SAT iff there is an accepting run of M_H that matches $\tilde{\gamma}$. By the construction of M_H, an accepting run of M_H on an input y can only exist if there is an integer $n \geq 0$ such that $y = 1^{n^{H(n)}}$. Note also that the length of y has to be bounded by $|\tilde{\gamma}_0|$. Define

$$N := \{n \in \mathbb{N} \mid n^{H(n)} \leq |\tilde{\gamma}_0|\}.$$

Then, as argued above, the following are equivalent:

1. $x \in$ SAT;
2. $\tilde{\gamma}_0 \# \tilde{\gamma}_1 \# \cdots \# \tilde{\gamma}_m \in \mathrm{RF}(M_H)$;
3. there is an $n \in N$ such that there is an accepting run of M_H on input $1^{n^{H(n)}}$ that matches $\tilde{\gamma}$.

In what follows, we show how to compute in polynomial time, for each $n \in N$, a propositional formula ϕ_n such that:

- ϕ_n is satisfiable if and only if there is an accepting run of M_H on input $1^{n^{H(n)}}$ that matches $\tilde{\gamma}$;
- $|\phi_n| \leq \frac{|x|}{|N|} - 2$ for all $n \in N$ (if x is large enough).

Then, the following function g is a polynomial-time many-one reduction from SAT to SAT:

$$g(x) := \bigvee_{n \in N} \phi_n.$$

Assuming a suitable encoding of propositional formulas, the size of $g(x)$ is then bounded by $|x| - 1$ for large enough x. Thus, g is the desired length-reducing polynomial-time self-reduction of SAT. It remains to construct ϕ_n, for all $n \in N$.

CONSTRUCTION OF ϕ_n. Fix $n \in N$. By the construction of M_H, any accepting run of M_H on input $1^{n^{H(n)}}$ has to start with the initialization phase. The first step of the initialization phase is deterministic, and checks whether the input has the form $1^{n^{H(n)}}$. Thus, we can complete $\tilde{\gamma}$ in polynomial time to a partial run of M_H where the first step of the initialization phase is completely specified. If this is not possible due to constraints imposed by $\tilde{\gamma}$, then we know that the desired accepting run does not exist, and we can output a trivial unsatisfiable formula ϕ_n. Otherwise, let

$$\tilde{\tilde{\gamma}} = (\tilde{\tilde{\gamma}}_0, \tilde{\tilde{\gamma}}_1, \ldots, \tilde{\tilde{\gamma}}_m)$$

be the resulting partial run of M_H. It remains to construct a formula ϕ_n that is satisfiable iff there is an accepting run of M_H that matches $\tilde{\tilde{\gamma}}$.

Let us take a closer look at $\tilde{\tilde{\gamma}}$. Let $i \geq 0$ be such that $\tilde{\tilde{\gamma}}_0, \ldots, \tilde{\tilde{\gamma}}_i$ corresponds to the first step of the initialization phase of M_H on input $1^{n^{H(n)}}$. In particular, for each $j \in \{0, 1, \ldots, i\}$, $\tilde{\tilde{\gamma}}_j$ is a completely specified configuration. It is possible to specify M_H in such a way that the second and third step of the initialization phase of M_H on input $1^{n^{H(n)}}$ take exactly n computation steps combined, and that any configuration after the initialization phase uses space at most n. Thus, without loss of generality we can assume:

1. $|\tilde{\tilde{\gamma}}_j| \leq n$ for all $j \in \{i+1, \ldots, m\}$;
2. $m - i - n$ is bounded by the running time of M_{SAT} on inputs of length n.

Let h be a polynomial-time computable function that, given $\tilde{\tilde{\gamma}}_{i+1} \# \cdots \# \tilde{\tilde{\gamma}}_m$, outputs a propositional formula that is satisfiable iff there is an accepting run of M_H that starts in the second step of the initialization phase of M_H in a configuration matching $\tilde{\tilde{\gamma}}_{i+1}$, and that matches $\tilde{\tilde{\gamma}}_{i+1}, \ldots, \tilde{\tilde{\gamma}}_m$. Let

$$\phi_n := h(\tilde{\tilde{\gamma}}_{i+1} \# \cdots \# \tilde{\tilde{\gamma}}_m).$$

This finishes the construction of ϕ_n.

It is immediate from the construction of ϕ_n that ϕ_n is satisfiable if and only if there is an accepting run of M_H on input $1^{n^{H(n)}}$ that matches $\tilde{\gamma}$. It remains to prove that the length of ϕ_n is bounded by $|x|/|N| - 2$.

BOUNDING THE SIZE OF ϕ_n. Let p be a polynomial such that for all strings z, M_{SAT} makes at most $p(|z|)$ steps on input z, and both $|f(z)|$ and $|h(z)|$ are bounded by $p(|z|)$. Since, as mentioned above, $m-i-n$ is bounded by the running time of M_{SAT} on inputs of length n, we have

$$m - i \;\leq\; p(n) + n.$$

Since moreover $|\tilde{\tilde{\gamma}}_j| \leq n$ for all $j \in \{i+1, \ldots, m\}$, we have

$$\begin{aligned} |\phi_n| \;&\leq\; |h(\tilde{\tilde{\gamma}}_{i+1} \# \cdots \# \tilde{\tilde{\gamma}}_m)| \\ &\leq\; p(|\tilde{\tilde{\gamma}}_{i+1} \# \cdots \# \tilde{\tilde{\gamma}}_m|) \\ &\leq\; p(\,(m-i)\cdot(n+1)\,) \\ &\leq\; p(\,(p(n)+n)\cdot(n+1)\,). \end{aligned}$$

Hence,

$$|\phi_n| \;\leq\; q(n)$$

for some polynomial q depending only on M_{SAT}, f, and h. It remains to show that for all $n \in N$ we have $q(n) \leq |x|/|N| - 2$ if x is sufficiently large.

Claim. There is a polynomial $r(\ell) > 0$ depending only on M_H such that for sufficiently large x we have $\frac{|x|}{|N|} \geq r(|x|)$.

Proof. Recall that N consists of all integers $n \geq 0$ such that $n^{H(n)} \leq |\tilde{\gamma}_0|$. Since $\tilde{\gamma}_0$ is part of $f(x)$, whose overall length is bounded by $p(|x|)$, we have $n^{H(n)} \leq p(|x|)$.

Now, for all integers $\ell \geq 0$, define

$$N(\ell) \;:=\; \{n \in \mathbb{N} \mid n^{H(n)} \leq p(\ell)\}.$$

Then, $N \subseteq N(|x|)$. We show that for all constants $c \in (0,1)$ there is an integer $\lambda_c \geq 0$ such that for all integers $\ell \geq \lambda_c$ we have $|N(\ell)| \leq \ell^c$. This implies the claim.[4]

Fix a constant $c \in (0,1)$ and $L := \{\ell \in \mathbb{N} \mid |N(\ell)| > \ell^c\}$. For each $\ell \in L$, there is an $n_\ell \in N(\ell)$ with $n_\ell \geq \ell^c$. Thus, by the definition of $N(\ell)$ and the monotonicity of H, for each $\ell \in L$ we have

$$\ell^{c \cdot H(\ell^c)} \;\leq\; n_\ell^{H(n_\ell)} \;\leq\; p(\ell). \tag{5}$$

Now, since $\lim_{\ell\to\infty} H(\ell) = \infty$ (by Lemma 14), we have $\lim_{\ell\to\infty} cH(\ell^c) = \infty$. Hence, there is an integer $\lambda_c \geq 0$ such that for all $\ell \geq \lambda_c$ we have $\ell^{c\cdot H(\ell^c)} > p(\ell)$. This implies that for all $\ell \geq \lambda_c$ we have $|N(\ell)| \leq \ell^c$ (otherwise, we would violate (5)). ⌟

Assume that $q(n) = n^k + k$. Let r be a polynomial as guaranteed by the claim. In what follows, we will assume that x is large enough so that:

1. $\frac{|x|}{|N|} \geq r(|x|)$; this can be satisfied by the previous claim.
2. $(r(|x|) - 2 - k)^{H\left((r(|x|)-2-k)^{\frac{1}{k}}\right)/k} > p(|x|)$; this is possible since $\lim_{\ell\to\infty} H(\ell) = \infty$ by Lemma 14.

[4] Set $r(\ell) := \ell^{1-c}$ for some $c \in (0,1)$ (e.g., $r(\ell) = \sqrt{\ell}$). Then, $\frac{|x|}{|N|} \geq \frac{|x|}{|N(|x|)|} \geq r(|x|)$ if $|x| \geq \lambda_c$.

Suppose that there is an $n \in N$ such that $q(n) > |x|/|N| - 2$. Then,

$$n > \left(\frac{|x|}{|N|} - 2 - k\right)^{\frac{1}{k}}.$$

This implies

$$\begin{aligned} n^{H(n)} \;&\geq\; \left(\frac{|x|}{|N|} - 2 - k\right)^{\frac{H(n)}{k}} \\ &\geq\; \left(\frac{|x|}{|N|} - 2 - k\right)^{\frac{H\left(\left(\frac{|x|}{|N|}-2-k\right)^{\frac{1}{k}}\right)}{k}} \\ &\geq\; (r(|x|) - 2 - k)^{\frac{H\left((r(|x|)-2-k)^{\frac{1}{k}}\right)}{k}} \\ &>\; p(|x|), \end{aligned}$$

where the last two inequalities follow from the monotonicity of H. Consequently,

$$\begin{aligned} |\tilde{\tilde{\gamma}}_{i+1} \# \cdots \# \tilde{\tilde{\gamma}}_m| \;&\leq\; |\tilde{\tilde{\gamma}}_0 \# \cdots \# \tilde{\tilde{\gamma}}_m| - |\tilde{\tilde{\gamma}}_0 \# \cdots \# \tilde{\tilde{\gamma}}_i| \\ &\leq\; p(|x|) - n^{H(n)} \\ &<\; p(|x|) - p(|x|), \end{aligned}$$

which is the desired contradiction. □

Lemma 4 (restated) *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.*

PROOF. We give the proof for $\mathcal{ALCIF}_\ell$ ontologies of depth 2. The proof for uGF$_2^-(2,f)$ is obtained by modifying the $\mathcal{ALCIF}_\ell$ ontology in the same way as in the proof of Theorem 10 by replacing, for example, $(\geq 2R)$ by $\exists y(R(x,y) \wedge \neg F(x,y))$.

Assume $M = (Q, \Gamma, \Delta, q_0, q_a)$ is given. The instances $\mathfrak{D}$ we use to represent partial runs and that provide the space for simulating matching runs are $n \times m$ X,Y-grids with T_{init} written in the lower left corner, T_{final} written in the upper right corner, and B (for blank) written everywhere else. To re-use the notation and results from the proof of Theorem 10 we regard such a structure as a tiling with tile types $\mathfrak{T} = \{B, T_{\mathsf{final}}, T_{\mathsf{init}}\}$. Then the ontology $\mathcal{O}_{\mathfrak{G}}$ for $\mathfrak{G} = (\mathfrak{T}, H, V)$ and

$$\begin{aligned} H \;&=\; \{(B,B), (B, T_{\mathsf{final}}), (T_{\mathsf{initial}}, B)\} \\ V \;&=\; \{(B,B), (B, T_{\mathsf{final}}), (T_{\mathsf{initial}}, B)\} \end{aligned}$$

checks whether an instance represents a grid structure. We now construct the set $\mathcal{O}_M$ of sentences that encode runs of M that match a partial run. For any $\mathfrak{D}$, the simulation of a run is triggered at a constant d exactly if $\mathcal{O}_{\mathfrak{G}}, \mathfrak{D} \models (= 1A)(d)$. $\mathcal{O}_M$ uses in addition to the binary relations in $\mathcal{O}_{\mathfrak{G}}$ binary relations $q \in Q$ that occur in concepts $(\geq 2q)$ that indicate that M is in state q. It also uses binary relations G for $G \in \Gamma$ that occur in concepts $(\geq 2G)$ that indicate that G is writte on the corresponding cell of the tage. The sentences of $\mathcal{O}_M$ are now as follows:

- The grid in which the lower left corner is marked with $(= 1A)$ is colored with $(= 1A)$, q_0 is the first symbol of the first configuration and no state occurs later in the first configuration:

$$(= 1A) \sqsubseteq \forall X.(= 1A) \sqcap \forall Y.(= 1A)\,,$$

$$(= 1A) \sqcap T_{\mathsf{init}} \sqsubseteq (\geq 2q_0)\ ,$$

$$(= 1A) \sqcap (= 1D) \sqcap (\geq 2q) \sqsubseteq (= 1L)$$

The remaining sentences are all relativized to $(= 1A)$ and so apply to constants in a grid only.

- every grid point is colored with some $(\geq 2G)$ for $G \in \Gamma$ or $(\geq 2q)$ for $q \in Q$:

$$(= 1A) \sqsubseteq \bigsqcup_{G \in \Gamma} (\geq 2G) \sqcup \bigsqcup_{q \in Q} (\geq 2q)$$

- The machine M cannot be in two different states at the same time and no two distinct symbols can be written on the same cell. For all mutually distinct $H_1, H_2 \in Q \cup \Gamma$:

$$(= 1A) \sqcap (\geq 2H_1) \sqcap (\geq 2H_2) \sqsubseteq \bot,$$

- For any triple $G_0qG_1 \in \Gamma \times Q \times \Gamma$ let $S(G_0qG_1)$ denote the set of all possible successor triples $S_1S_2S_3 \in (Q \times \Gamma \times \Gamma) \cup (\Gamma \times \Gamma \times Q)$ according to the transition relation Δ of M. To avoid sentences of depth larger than two we introduce for the words $W \in \{X, XX\}$ and for $S \in Q \cup \Gamma$ fresh binary relations S^W and the sentences

$$(= 1A) \sqcap (\geq 2S^X) \equiv (= 1A) \sqcap \exists X.(\geq 2S)\ ,$$

$$(= 1A) \sqcap (\geq 2S^{XX}) \equiv (= 1A) \sqcap \exists X.(\geq 2S^X)$$

and then add

$$(= 1A) \sqcap (\geq 2G_0) \sqcap (\geq 2q^X) \sqcap (\geq 2G_1^{XX}) \sqsubseteq$$

$$\bigsqcup_{S_1S_2S_3 \in S(G_0qG_1)} \exists Y.(\geq 2S_1) \sqcap (\geq 2S_2^X) \sqcap (\geq 2S_3^{XX})$$

to $\mathcal{O}_M$.

- the symbol written on a cell does not change if the head is more than one cell away. For all $G, G_1, G_2 \in \Gamma$:

$$(= 1A) \sqcap \forall X.G_1^{\geq 2} \sqcap \forall X^-.G_2^{\geq 2} \sqcap G^{\geq 2} \sqsubseteq \forall Y.G^{\geq 2}$$

for $G_i^{\geq 2} := (\geq 2G_i)$ with $i \in \{1, 2\}$ or empty.

- the final state cannot be a state distinct from the accepting state q_a. For all $q \in Q \setminus \{q_a\}$:

$$(= 1A) \sqcap (\geq 2q) \sqsubseteq \exists Y.\top$$

- Finally, let AUX_M denote the set of fresh binary relations used above and add

$$\top \sqsubseteq \exists Q.\top$$

to $\mathcal{O}_M$ for all $Q \in \mathrm{AUX}_M$.

This finishes the definition of $\mathcal{O}_M$. Let $\mathcal{O} = \mathcal{O}_{\mathfrak{G}} \cup \mathcal{O}_M$. We show that $\mathcal{O}$ is as required.

Let N be a fresh unary relation symbol. Then an instance $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}$ if $\mathcal{O}, \mathfrak{D} \not\models q$ for the Boolean query $q \leftarrow N(x)$. It therefore suffices to provide a polynomial reduction of the run fitting problem for M to the problem whether an instance $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}$.

Assume that a partial run $\tilde{\gamma} = (\tilde{\gamma}_0, \tilde{\gamma}_1, \ldots, \tilde{\gamma}_m)$ of partial configurations $\tilde{\gamma}_i$ of M such that $\tilde{\gamma}_0$ starts with q_0 and $|\tilde{\gamma}_0| = \cdots = |\tilde{\gamma}_m| = n + 1$ is given. We define an instance $\mathfrak{D}$ with $\mathfrak{D} \models \mathrm{grid}(0, 0)$ which encodes the partial run. Thus we regard (i, j) with $0 \leq i \leq n$ and $0 \leq j \leq m$ as constants and $\mathfrak{D}$ contains the assertions

$$X((i,j),(i+1,j)), \qquad Y((i,j),(i,j+1)),$$
$$T_{\mathsf{init}}(0,0), \qquad T_{\mathsf{final}}(n,m)$$

and $B(i, j)$ for $(i, j) \notin \{(0, 0), (n, m)\}$. In addition, we include in $\mathfrak{D}$ the atoms

$$S((i,j), d^1_{i,j}), \quad S((i,j), d^2_{i,j})$$

for distinct fresh constants $d^1_{i,j}$ and $d^2_{i,j}$ for all i, j such that $\tilde{\gamma}_j[i] = S$ and $S \neq \star$. It is now straightforward to show that $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}$ iff there is an accepting run of M that matches $\tilde{\gamma}$.

For the converse direction, we have to provide for every UCQ q a polynomial reduction of the query evaluation problem for $(\mathcal{O}, q)$ to the complement of the run fitting problem for M. To this end observe that the following two conditions are equivalent for any UCQ $q(\vec{x})$, instance $\mathfrak{D}$, and tuple $\vec{a}$:

1. $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$
2. $\mathfrak{D}$ is not consistent w.r.t. $\mathcal{O}$ or

$$\{\top \sqsubseteq \exists Q.\top \mid Q \in \mathrm{AUX}\}, \mathfrak{D} \models q(\vec{a})$$

where $\mathrm{AUX} = \mathrm{AUX}_{\mathrm{cell}} \cup \mathrm{AUX}_{\mathrm{grid}} \cup \mathrm{AUX}_M$.

As the latter problem is in PTIME it suffices to provide a polynomial reduction of the problem whether an instance $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}$ to the run fitting problem for M. Assume $\mathfrak{D}$ is given. First decide in polynomial time whether $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}_{\mathfrak{G}}$ (Lemma 12). If not, we are done. If yes, let G be the set of all $d \in \mathrm{dom}(\mathfrak{D})$ such that $\mathfrak{D} \models \mathrm{grid}(d)$. Assume without loss of generality that $G = \{0, \ldots, k\}$. Then we find natural numbers n_i, m_i such that each $i \in G$ is the root of an $n_i \times m_i$-grid with witness function β_i for $\mathfrak{G}$. By Lemma 12, there is a materialization $\mathfrak{B}$ of $\mathfrak{D}$ and $\mathcal{O}_{\mathfrak{G}}$ such that $d \in G$ iff $d \in (= 1A)^{\mathfrak{B}}$. Next we check in polynomial time that $\mathfrak{D}$ is consistent w.r.t. the union of $\mathcal{O}_{\mathfrak{G}}$ and the sentences of the form

$$(= 1A) \sqcap (\geq 2S^X) \equiv (= 1A) \sqcap \exists X.(\geq 2S),$$

$$(= 1A) \sqcap (\geq 2S^{XX}) \equiv (= 1A) \sqcap \exists X.(\geq 2S^X)$$

in $\mathcal{O}_M$. If this is not the case we are done. If this is the case we assume that $\mathfrak{D}$ is saturated in the sense that if, for example, $(= 1A) \sqcap (\geq 2S^X) \equiv (= 1A) \sqcap \exists X.(\geq 2S)$ is in $\mathcal{O}_M$ then for any $d \in (= 1A)^{\mathfrak{B}}$ and $X(d, d') \in \mathfrak{D}$ the following holds: d has at least two S^X-successors in $\mathfrak{D}$ iff d' has at least two S-successors in $\mathfrak{D}$. Now for each $i \in G$ we define the sequences of strings

$$\tilde{\gamma}^i = (\tilde{\gamma}^i_0, \ldots, \tilde{\gamma}^i_{m_i})$$

by setting for $0 \leq r \leq n_i$ and $S \in \Gamma \cup Q$,

$$\tilde{\gamma}^i_r[j] = S \text{ iff } \beta_i(j, r) \text{ has at least two } S\text{-successors in } \mathfrak{D}$$

If any of these strings is not a partial configuration, then $\mathfrak{D}$ is not consistent w.r.t. $\mathcal{O}$ and we are done. Otherwise each $\tilde{\gamma}^i$ is a partial run of M. It is now straightforward to show that $\mathfrak{D}$ is consistent w.r.t. $\mathcal{O}$ iff for each $i \in G$ there exists an accepting run of M that matches $\tilde{\gamma}^i$ which provides us with a polynomial reduction of the consistency problem for instances $\mathfrak{D}$ to the run fitting problem for M. $\square$

I. PROOFS FOR SECTION 8

We start by introducing the basic notions used in this section in more detail. An interpretation $\mathfrak{D}$ is *$\mathcal{O}$-saturated* if for all atoms $R(\vec{a})$ 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* $\mathfrak{D}$. The following lemma states basic properties of $\mathcal{O}$-saturated instances.

Lemma 16 *Let $\mathfrak{D} \subseteq \mathfrak{D}'$ be instances with $\mathfrak{D}'_{|dom(\mathfrak{D})} = \mathfrak{D}$ and let $\mathcal{O}$ be a uGC$_2(=)$ ontology.*

1. *There exists a materialization of $\mathcal{O}$ and $\mathfrak{D}$ iff there exists a materialization of $\mathcal{O}$ and the $\mathcal{O}$-saturation of $\mathfrak{D}$.*
2. *If $\mathfrak{B}$ is a materialization of $\mathcal{O}$ and $\mathfrak{D}$ and $\mathfrak{D}$ is $\mathcal{O}$-saturated, then $\mathfrak{B}_{|dom(\mathfrak{D})} = \mathfrak{D}$.*
3. *If $\mathfrak{D}'$ is $\mathcal{O}$-saturated, then $\mathfrak{D}$ is $\mathcal{O}$-saturated.*

We also require a lemma about stronger forms of unravelling tolerance and forest models that one can employ for $\mathcal{ALCHIQ}$ ontologies. Call a model $\mathfrak{B}$ of an instance $\mathfrak{D}$ an *irreflexive forest model* if it is obtained from $\mathfrak{D}$ by adding atoms of the form $R(a, b)$ with $a \neq b$ in dom($\mathfrak{D}$) to $\mathfrak{D}$ and hooking irreflexive tree interpretations $\mathfrak{B}_a$ to every $a \in \text{dom}(\mathfrak{D})$. The following variations of Lemma 1 and Theorem 7 can be proved by modifying the proofs in a straightforward way taking into account the limited expressive power of $\mathcal{ALCHIQ}$.

Lemma 17 *Let $\mathcal{O}$ be an $\mathcal{ALCHIQ}$ ontology of depth 1. Then the following hold:*

1. *Let $\mathfrak{A}$ be a model of $\mathcal{O}$ and $\mathfrak{D}$. Then there exists an irreflexive forest model $\mathfrak{B}$ of $\mathcal{O}$ and $\mathfrak{D}$ such that there exists a homomorphism h from $\mathfrak{B}$ to $\mathfrak{A}$ that preserves dom($\mathfrak{D}$).*
2. *$\mathcal{O}$ is materializable iff $\mathcal{O}$ is materializable for the class of all irreflexive tree instance $\mathfrak{D}$ with dom($\mathfrak{D}$) $\subseteq$ sig($\mathcal{O}$).*

To characterize materializability in uGC$_2^-(1,=)$ we admit 1-materializability witnesses with loops and require that they satisfy additional closure conditions which ensure that one can construct a materialization in a step-by-step fashion, as in the proof of Lemma 6. To formulate the conditions we employ a 'mosaic technique' and introduce mosaic pieces consisting of a 1-materializability witness and a homomorphism into a bouquet. Given a set of such mosaic pieces satisfying certain closure conditions we can then construct the materialization of a bouquet step-by-step using the 1-materializability witnesses and, simultaneously, construct a homomorphism into any other model of the bouquet (and thereby prove that we have indeed constructed a materialization). The resulting procedure for checking materializability will be in NEXPTIME as we can first guess an exponential size set of mosaic pieces and then confirm in exponential time that it satisfies our conditions.

A *1-model pair* is a tuple $(\mathfrak{F}, a, \mathfrak{B})$ such that

- $\mathfrak{F}$ is a bouquet with root a that is consistent w.r.t. $\mathcal{O}$, of outdegree $\leq |\mathcal{O}|$ and such that sig($\mathfrak{D}$) $\subseteq$ sig($\mathcal{O}$);
- $\mathfrak{B}$ is a bouquet with root a of outdegree $\leq 2|\mathcal{O}|$ that is a model of $\mathfrak{F}$ such that there exists a model $\mathfrak{A}$ of $\mathfrak{F}$ and $\mathcal{O}$ with $\mathfrak{A}_a^{\leq 1} = \mathfrak{B}$.

A 1-model pair $(\mathfrak{F}, a, \mathfrak{B})$ is a *1-materializability witness* if $\mathfrak{B}$ is a 1-materialization of $\mathfrak{F}$ w.r.t. $\mathcal{O}$. We require two different types of mosaic pieces. First, an *injective hom-pair* takes the form $(\mathfrak{F}, a, \mathfrak{B}) \to_h (\mathfrak{F}', a', \mathfrak{B})$ such that

- $(\mathfrak{F}, a, \mathfrak{B})$ is a 1-materializability witness and $(\mathfrak{F}', a', \mathfrak{B}')$ is a 1-model pair;
- h is a homomorphism from $\mathfrak{B}$ to $\mathfrak{B}'$ and its restriction to dom($\mathfrak{F}$) is a bijective homomorphism onto $\mathfrak{F}'$ mapping a to a'.

Thus, in an injective hom-pair the 1-materializability witness is a piece of the materialization we wish to construct and the 1-model pair is a piece of the model into which we wish to homomorphically embed the materialization. Injective hom-pairs assume the homomorphic embedding one wants to extend (i.e., the restriction of h to dom($\mathfrak{F}$)) is injective. To deal with non-injective embeddings (as indicated by Example 7) we also consider *contracting hom-pairs* which take the form $(\mathfrak{F}, a, \mathfrak{B}) \to_h (\mathfrak{B}', a')$ where

- $(\mathfrak{F}, a, \mathfrak{B})$ is a 1-materializability witness with dom($\mathfrak{F}$) $= \{a, b\}$;
- $(\mathfrak{B}', a')$ is a bouquet with root a' of outdegree at most $2|\mathcal{O}|$ such that there exists a model $\mathfrak{A}$ of $\mathcal{O}$ with $\mathfrak{A}_{a'}^{\leq 1} = \mathfrak{B}'$;
- h is a homomorphism from $\mathfrak{B}$ to $\mathfrak{B}'$ with $h(a) = h(b) = a'$.

Similarly to injective hom-pairs, in a contracting hom-pair the 1-materializability witness is a piece of the materialization we wish to construct and the second component is a piece of the model into which we wish to homomorphically embed the materialization. The following lemma now provides a NEXPTIME decision procedure for materializability of uGC$_2^-(1,=)$ ontologies.

Lemma 18 *Let $\mathcal{O}$ be a uGC$_2^-(1,=)$ ontology. Then $\mathcal{O}$ is materializable iff there exist*

1. *a set M of 1-materializability witnesses containing exactly one 1-materializability witness for every bouquet $\mathfrak{F}$ of outdegree $\leq |\mathcal{O}|$ with sig($\mathfrak{F}$) $\subseteq$ sig($\mathcal{O}$) that is consistent relative to $\mathcal{O}$;*
2. *sets H of injective hom-pairs and E of contracting hom-pairs whose first components are all in M*

such that the conditions of Table 1 hold.

1. If $(\mathfrak{F}, a, \mathfrak{B}) \in M$ and $\mathfrak{F}'$ is a bouquet with root a' such that there is a bijective homomorphism h_0 from $\mathfrak{F}$ to $\mathfrak{F}'$ mapping a to a', and $\mathfrak{B}'$ is an interpretation such that $(\mathfrak{F}', a', \mathfrak{B}')$ is a 1-model pair, then there exists an extension h of h_0 such that $(\mathfrak{F}, a, \mathfrak{B}) \to_h (\mathfrak{F}', a', \mathfrak{B}') \in H$.
2. If $(\mathfrak{F}, a, \mathfrak{B}) \to_h (\mathfrak{F}', a', \mathfrak{B}') \in H$ or $(\mathfrak{F}, a, \mathfrak{B}) \to_h (\mathfrak{B}', a') \in E$ with $b \in \text{dom}(\mathfrak{B}) \setminus \text{dom}(\mathfrak{F})$ and $h(a) = h(b) = a'$, then there exist h' and $\mathfrak{B}''$ such that $(\mathfrak{B}_{|\{a,b\}}, b, \mathfrak{B}'') \to_{h'} (\mathfrak{B}', a') \in E$.

Table 1: Conditions for M, H, and E in materializability check for uGC$_2^-(1,=)$

PROOF. Using Lemma 2 one can show that sets M, H, and E satisfying the conditions of Table 1 exist if $\mathcal{O}$ is materializable. Conversely, let the sets M, H, and E satisfy the conditions given in Lemma 18. Assume a bouquet $\mathfrak{D}$ of outdegree $\leq |\mathcal{O}|$ with sig($\mathfrak{D}$) $\subseteq$ sig($\mathcal{O}$) and root a is given. Take the 1-materializability witness $(\mathfrak{D}, a, \mathfrak{B}) \in M$. As in the proof of Lemma 6 we construct a sequence of interpretations $\mathfrak{B}^0, \mathfrak{B}^1, \ldots$ and sets $F_i \subseteq \text{dom}(\mathfrak{B}^i)$ of frontier elements (but now using only 1-materializability witnesses in M): set $\mathfrak{B}^0 := \mathfrak{B}$ and $F_0 = \text{dom}(\mathfrak{B}) \setminus \text{dom}(\mathfrak{D})$. If $\mathfrak{B}^i$ and F_i have been constructed, then take for any $b \in F_i$ its (unique) predecessor a and a 1-materializability witness $(\mathfrak{B}^i_{|\{a,b\}}, b, \mathfrak{B}_b) \in M$ and set $\mathfrak{B}_{i+1} := \mathfrak{B}^i \cup \bigcup_{b \in F_i} \mathfrak{B}_b$. Let $\mathfrak{B}^*$ be the union of all $\mathfrak{B}_i$. We show that $\mathfrak{B}$ is a materialization. $\mathfrak{B}$ is a model of $\mathcal{O}$ by construction since $\mathcal{O}$ is a uGC$_2^-(1,=)$ ontology. Consider a model $\mathfrak{A}$ of $\mathcal{O}$ and $\mathfrak{D}$. We construct a homomorphism h from $\mathfrak{B}^*$ to $\mathfrak{A}$ preserving dom($\mathfrak{D}$) as the limit of a sequence $h_0, \ldots$ of homomorphisms from $\mathfrak{B}^i$ to $\mathfrak{A}$. We may assume that the outdegree of $\mathfrak{A}$ is

$\leq 2|\mathcal{O}|$. By definition, there exists a homomorphism from $\mathfrak{B}^0$ to $\mathfrak{A}_a^{\leq 1}$ preserving $\mathfrak{D}$. Now, inductively, we ensure in each step that the homomorphisms h_i satisfy the following conditions for all $\mathfrak{B}^i$ and F_i, all $b \in F_i$ and the predecessor a of b in $\mathfrak{B}^{i-1}$:

(a) Assume $h_i(a) \neq h_i(b)$. Then, for the 1-materializability witness $(\mathfrak{B}^*_{|\{a,b\}}, b, \mathfrak{B}_b) \in M$ there exists a homomorphism h_b such that

$$(\mathfrak{B}^*_{|\{a,b\}}, b, \mathfrak{B}_b) \to_{h_b} (\mathfrak{A}_{|\{h_i(a),h_i(b)\}}, h_i(b), \mathfrak{A}^{\leq 1}_{h_i(b)}) \in H$$

(b) Assume $h_i(a) = h_i(b)$. Then, for the 1-materializability witness $(\mathfrak{B}^*_{|\{a,b\}}, b, \mathfrak{B}_b) \in M$ there exists a homomorphism h_b such that

$$(\mathfrak{B}^*_{|\{a,b\}}, b, \mathfrak{B}_b) \to_{h_b} (\mathfrak{A}^{\leq 1}_{h_i(b)}, h_i(b)) \in E$$

Assume h_i with the properties (a) and (b) has been constructed. Then we take for all $b \in F_i$ and the predecessor a of b the homomorphism h_b determined by (a) and, respectively, (b) and set

$$h_{i+1} := h_i \cup \bigcup_{b \in F_i} h_b$$

Using the properties of M, H, and E given in Table 1 it is not difficult to show that h_{i+1} again has the properties (a) and (b) above. $\square$

We prove Theorem 14, that is, for $\mathcal{ALC}$ ontologies of depth 2, deciding whether query-evaluation is in PTIME is NEXPTIME-hard (unless PTIME = CONP). The proof is by reduction of the complement of a NEXPTIME-complete tiling problem in which the aim is to tile a $2^n \times 2^n$-grid. An instance of this problem is given by a tuple $P = (\mathcal{T}, t_0, H, V)$ where $\mathcal{T}$ is a set of *tile types*, $t_0 \in \mathcal{T}$ is a distinguished tile to be placed on position $(0, 0)$ of the grid, and H and V are horizontal and vertical matching conditions. A *solution* to P is a function $\tau : 2^n \times 2^n \to \mathcal{T}$ such that

- if $\tau(i, j) = t$ and $\tau(i + 1, j) = t'$ then $(t, t') \in H$, for all $i < 2^n - 1, j < 2^n$,
- if $\tau(i, j) = t$ and $\tau(i, j + 1) = t'$ then $(t, t') \in V$, for all $i < 2^n, j < 2^n - 1$,
- $\tau(0, 0) = t_0$.

Let $P = (\mathcal{T}, t_0, H, V)$. We construct an $\mathcal{ALC}$ ontology $\mathcal{O}$ of depth 2 such that P has a solution iff $\mathcal{O}$ is not materializable.

The idea is that $\mathcal{O}$ verifies the existence of an r-chain in the input instance whose elements represent the grid positions along with a tiling, row by row from left to right, starting at the lower left corner and ending at the upper right corner. The positions in the grid are represented in binary by the concept names $X_1, \dots, X_{2n}$ in the instance where $X_1, \dots, X_n$ indicate the horizontal position and $X_{n+1}, \dots, X_{2n}$ the vertical position. The tiling is represented by unary relations T_t, $t \in \mathcal{T}$. $\mathcal{O}$ verifies the chain by propagating a marker bottom up and while doing this, it verifies the horizontal matching condition. When the top of the chain is reached, $\mathcal{O}$ generates another r-chain using existential quantifiers. On that chain, a violation of the vertical matching condition is guessed using disjunction, that is, the position where the violation occurs and the tiles involved in it. If the tiling on the first chain is defective, the guesses can be made such that the second chain homomorphically maps into the first one, and then the second chain is 'invisible' to the query. Otherwise, the query can 'see' the second chain and because of the disjunctions involved in building it, the instance has no materialization w.r.t. $\mathcal{O}$. Clearly, the second case can occur only if P has a solution.

We now assemble the ontology $\mathcal{O}$. To avoid that $\mathcal{O}$ is trivially not materializable, we have to hide the marker propagated up the chain in the input and all markers used on the existentially generated chain. We thus use concepts of the form $\forall s.A$ instead of concept names A, additionally stating in $\mathcal{O}$ that $\top \sqsubseteq \exists s.A$. For any concept name A, we use H_A as an abbreviation of $\forall s.A$.

1. Set up hiding of concept names: for all $A \in \{V, \mathsf{ok}_1, \dots, \mathsf{ok}_{2n}, D, L, Y_1, \dots, Y_{2n}, \overline{Y}_1, \dots, \overline{Y}_{2n}, Z_1, \dots, Z_n, \overline{Z}_1, \dots, \overline{Z}_n\} \cup \{S_t \mid t \in \mathcal{T}\}$,
$$\top \sqsubseteq \exists s.A$$

2. The initial position starts the propagation: for all $t \in \mathcal{T}$,
$$\overline{X}_1 \sqcap \dots \sqcap \overline{X}_{2n} \sqcap T_t \sqsubseteq H_V$$

3. Tiles are mutually exclusive: for all distinct $t, t' \in \mathcal{T}$:
$$T_t \sqcap T_{t'} \sqsubseteq \bot$$

4. The propagation proceeds upwards, checking the horizontal matching condition:
$$\begin{aligned}
X_i \sqcap \exists r.X_i \sqcap \textstyle\bigsqcup_{1 \leq j < i} \exists r.X_j &\sqsubseteq H_{\mathsf{ok}_i} \\
\overline{X}_i \sqcap \exists r.\overline{X}_i \sqcap \textstyle\bigsqcup_{1 \leq j < i} \exists r.X_j &\sqsubseteq H_{\mathsf{ok}_i} \\
X_i \sqcap \exists r.\overline{X}_i \sqcap \textstyle\bigsqcap_{1 \leq j < i} \exists r.\overline{X}_j &\sqsubseteq H_{\mathsf{ok}_i} \\
\overline{X}_i \sqcap \exists r.X_i \sqcap \textstyle\bigsqcap_{1 \leq j < i} \exists r.\overline{X}_j &\sqsubseteq H_{\mathsf{ok}_i} \\
H_{\mathsf{ok}_1} \sqcap \dots \sqcap H_{\mathsf{ok}_n} \sqcap \overline{X}_i \sqcap & \\
\exists r.(H_V \sqcap T_t) \sqcap T_{t'} &\sqsubseteq H_V \\
\exists r.X_i \sqcap \exists r.\overline{X}_i &\sqsubseteq \bot
\end{aligned}$$
where i ranges over $1..2n$ and (t, t') over H; the use of $\overline{X}_i$ in the third last line prevents the counter from wrapping around at maximum value. The last inclusion is necessary to avoid that multiple successors that carry different concept name labels interact in undesired ways.

5. When the maximum value is reached, we make an extra step in the instance and then generate the first object on an existential chain that represents a tiling defect:
$$\exists r.(X_1 \sqcap \dots \sqcap X_{2n} \sqcap H_V) \sqsubseteq \exists r.C$$
where
$$C = H_M \sqcap H_{Y_1} \sqcap \dots \sqcap H_{Y_{2n}}$$

6. We continue building the chain; in every step, we decide whether we have a defect here (H_D) or defer it to later (H_L); we can defer at most to the left-most position of the second row:
$$\begin{aligned}
H_M &\sqsubseteq (H_D \sqcap C_D) \sqcup H_L \\
H_M \sqcap H_{\overline{Y}_1} \sqcap \dots \sqcap H_{\overline{Y}_n} \sqcap & \\
H_{Y_{n+1}} \sqcap H_{\overline{Y}_{n+2}} \sqcap \dots \sqcap H_{\overline{Y}_{2n}} &\sqsubseteq H_D
\end{aligned}$$
where C_D is a concept to be defined later.

7. When defering the defect to later, we keep going and maintain the Y-counter, decrementing it:
$$\begin{aligned}
H_L &\sqsubseteq \exists r.H_M \\
H_L \sqcap \textstyle\bigsqcap_{1 \leq j \leq i} H_{\overline{Y}_i} &\sqsubseteq \forall r.H_{Y_i} \\
H_L \sqcap H_{Y_i} \sqcap \textstyle\bigsqcap_{1 \leq j < i} H_{\overline{Y}_i} &\sqsubseteq \forall r.H_{\overline{Y}_i} \\
H_L \sqcap H_{Y_i} \sqcap \textstyle\bigsqcup_{1 \leq j < i} H_{Y_i} &\sqsubseteq \forall r.H_{Y_i} \\
H_L \sqcap H_{\overline{Y}_i} \sqcap \textstyle\bigsqcup_{1 \leq j < i} H_{Y_i} &\sqsubseteq \forall r.H_{\overline{Y}_i}
\end{aligned}$$
where i ranges over $1..2n$.

8. When implementing the defect, we guess two V-incompatible tiles, start a new counter, travel exactly 2^n steps, and verify the tiles. The above concept C_D is

$$C_D = H_{Z_1} \sqcap \cdots \sqcap H_{Z_n} \sqcap \bigsqcup_{(t,t') \notin V} (T_t \sqcap H_{S_{t'}})$$

and we add the following concept inclusions:

$$\begin{aligned}
H_D \sqcap H_{Z_i} &\sqsubseteq \exists r.(H_M \sqcap H_D) \\
H_D \sqcap \textstyle\bigsqcap_{1 \leq j \leq i} H_{\overline{Z}_i} &\sqsubseteq \forall r.H_{Z_i} \\
H_D \sqcap H_{Z_i} \sqcap \textstyle\bigsqcap_{1 \leq j < i} H_{\overline{Z}_i} &\sqsubseteq \forall r.H_{\overline{Z}_i} \\
H_D \sqcap H_{Z_i} \sqcap \textstyle\bigsqcup_{1 \leq j < i} H_{Z_i} &\sqsubseteq \forall r.H_{Z_i} \\
H_D \sqcap H_{\overline{Z}_i} \sqcap \textstyle\bigsqcup_{1 \leq j < i} H_{Z_i} &\sqsubseteq \forall r.H_{\overline{Z}_i} \\
H_D \sqcap H_{Z_i} \sqcap H_{S_t} &\sqsubseteq \forall r.H_{S_t} \\
H_D \sqcap H_{\overline{Z}_1} \sqcap \cdots \sqcap H_{\overline{Z}_n} \sqcap H_{S_t} &\sqsubseteq T_t
\end{aligned}$$

where i ranges over $1..n$.

We now establish correctness of the reduction. A *tiling chain* is an instance $\mathfrak{D}$ that consists of the following assertions:

- $r(a_i, a_{i+1})$ for $0 \leq i < 2^{2n} - 1$
- $X_j(a_i)$ whenever the j-th bit of i is one
- $\overline{X}_j(a_i)$ whenever the j-th bit of i is zero
- exactly one $T_t(a_i)$ for each i, $t \in \mathcal{T}$, such that when $T_t(a_i), T_{t'}(a_{i+1}) \in \mathfrak{D}$, then $(t, t') \in H$.

We say that the tiling chain $\mathfrak{D}$ is *defective* if there is an $i \leq 2^{2n} - (2^n + 1)$ such that $T_t(a_i), T_{t'}(a_{i+2^n}) \in \mathfrak{D}$ and $(t, t') \notin V$.

Lemma 19 *P has a solution iff $\mathcal{O}$ is not materializable.*

PROOF. (sketch) "if". Assume that P has no solution. Take an instance $\mathfrak{D}$. We have to construct a CQ-materialization $\mathfrak{A}$ of $\mathfrak{D}$ and $\mathcal{O}$. To do this, start with $\mathfrak{D}$ viewed as an interpretation $\mathfrak{A}$. To satisfy the concept inclusions in Points 1-4, which all fall within monadic Datalog, apply a standard chase procedure. To satisfy the concept inclusions in Point 5 to 8, consider every $a \in (\exists r.(X_1 \sqcap \cdots \sqcap X_{2n} \sqcap H_V))^{\mathcal{I}}$. The concept $H_V = \forall s.V$ cannot be made true by an atom in an instance and, consequently, it was made true at a by the chase. Analyzing the concept inclusions in $\mathcal{O}$, one can show that there must thus be a tiling chain $\mathcal{C} \subseteq \mathfrak{D}$ whose last element $a_{2^{2n}-1}$ is a. Since there is no solution, $\mathcal{C}$ must be defective. When generating the existential chain, we can thus make the guesses in a way such that the chain homomorphically maps into $\mathcal{C}$, thus into $\mathfrak{D}$. It can be verified that this yields the desired CQ-materialization of $\mathcal{O}$ and $\mathfrak{D}$.

"only if". Assume that P has a solution. Let $\mathfrak{D}$ be the tiling chain that represents it. We show that there is no materialization of $\mathcal{O}$ and $\mathfrak{D}$. In fact, $\mathcal{O}$ propagates the H_V marker all the way up to the last element $a = a_{2^{2n}-1}$ of $\mathfrak{D}$, where then an existential chain is generated. Because of the use of disjunctions to generate all different kinds of violations of the vertical tiling conditions, there are clearly at least two chains that can be generated and that are incomparable in terms of homomorphisms. Moreover, since $\mathfrak{D}$ represents a proper tiling, none of the chains homomorphically maps to $\mathfrak{D}$. Consequently, we can find CQs $q_1(x), \ldots, q_m(x)$ such that $\mathcal{O}, \mathfrak{D} \models q_1(a) \vee \cdots \vee q_m(a)$, but $\mathcal{O}, \mathfrak{D} \not\models q_i(a)$ for any i. Thus, $\mathcal{O}$ does not have the disjunction property and consequently is not materializable. $\square$