

Dichotomies in Ontology-Mediated Querying with the Guarded Fragment

ANDRÉ HERNICH, University of Liverpool, UK
 CARSTEN LUTZ, University of Bremen, Germany
 FABIO PAPACCHINI, University of Liverpool, UK
 FRANK WOLTER, University of Liverpool, UK

We study ontology-mediated querying in the case where ontologies are formulated in the guarded fragment of first-order logic (GF) or extensions thereof with counting and where the actual queries are (unions of) conjunctive queries. Our aim is to classify the data complexity and Datalog rewritability of query evaluation depending on the ontology \mathcal{O} , where query evaluation w.r.t. \mathcal{O} is in PTIME (resp. Datalog rewritable) if all queries can be evaluated in PTIME w.r.t. \mathcal{O} (resp. rewritten into Datalog under \mathcal{O}), and CONP-hard if at least one query is CONP-hard w.r.t. \mathcal{O} . We identify several fragments of GF that enjoy a dichotomy between Datalog-rewritability (which implies PTIME) and CONP-hardness as well as several other fragments that enjoy a dichotomy between PTIME and CONP-hardness, but for which PTIME does not imply Datalog-rewritability. For the latter, we establish and exploit a connection to constraint satisfaction problems. We also identify fragments for which there is no dichotomy between PTIME and CONP. To prove this, we establish a non-trivial variation of Ladner's theorem on the existence of NP-intermediate problems. Finally, we study the decidability of whether a given ontology enjoys PTIME query evaluation, presenting both positive and negative results, depending on the fragment.

Additional Key Words and Phrases: Ontology-Based Data Access; Query Evaluation; Dichotomies

ACM Reference Format:

André Hernich, Carsten Lutz, Fabio Papacchini, and Frank Wolter. 2020. Dichotomies in Ontology-Mediated Querying with the Guarded Fragment. 1, 1, Article 1 (February 2020), 74 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Ontology-mediated querying is a paradigm of data access in which incomplete data is enriched with an ontology to provide domain knowledge and to enable more complete answers to queries, see [14, 47, 67] for recent surveys. In this context, an *ontology-mediated query (OMQ)* is a pair $Q = (\mathcal{O}, q)$ with \mathcal{O} an ontology and q an actual query. Relevant ontology languages include decidable fragments of first-order logic such as description logics (DLs) [7, 8] and decidable classes of tuple-generating dependencies (TGDs), also known as Datalog[±] and as existential rules [20, 64]. Prominent choices for the actual query language

An extended abstract of this paper was published in the Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017 [43].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.

XXXX-XXXX/2020/2-ART1 \$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

are conjunctive queries (CQs) and unions thereof (UCQs). A lot of research has been undertaken to understand the complexity of OMQ evaluation and the rewritability of OMQs into more conventional database query languages such as Datalog, see for example [21, 22, 38–40, 45, 65]. Regarding the former, the two most important complexity measures are combined complexity and data complexity. In combined complexity, the data and the OMQ are both considered to be inputs. In data complexity, in contrast, only the data is the input while the OMQ is fixed. It is often conceived as a minimum requirement for practical efficiency that the data complexity of OMQ evaluation should be in PTIME. Rewritability into conventional database languages is studied, on the one hand, since this can be an approach to efficient query execution in practice. On the other hand, rewritability is thoroughly intertwined with data complexity and, in particular, every ontology-mediated query that is rewritability into Datalog has PTIME data complexity.

Fine-Grained Complexity. The importance of PTIME data complexity is conflicting with the fact that many desirable features of ontology languages result in OMQ evaluation to become at least CONP-hard [17, 19, 21, 48, 70]. This has led to the design of ontology languages such as the description logics DL-Lite, \mathcal{EL} , and Horn-*SHIQ* [4, 6, 44] that have limited expressive power, but guarantee PTIME (or lower) data complexity and even rewritability into (versions of) Datalog. In practical applications, however, ontology engineers often need to use language features that are only available in computationally expensive ontology languages, but they typically do so in a way such that one may hope for hardness to be avoided by the concrete ontologies that are being designed. Initiated in [57, 59], this observation has led to studies of data complexity and rewritability that are *more fine-grained* than the analysis of entire ontology languages. The approach taken in [59] analyzes the data complexity on the level of *individual ontologies* \mathcal{O} , universally quantifying over the actual query: query evaluation w.r.t. an ontology \mathcal{O} is in PTIME if every OMQ (\mathcal{O}, q) can be evaluated in PTIME and it is CONP-hard if there is at least one OMQ (\mathcal{O}, q) that is CONP-hard to evaluate. In this way, one can identify tractable ontologies within ontology languages that are, in general, computationally hard. An even more fine-grained approach is taken in [15], where quantification over the query is avoided and one aims to classify the complexity of each OMQ. Both approaches are reasonable. In this paper, we follow the first one which is preferable when the queries to be answered are not fixed at the design time of the ontology, in line with the common view that ontologies are general purpose artifacts to be used in more than a single application.

As a concrete example for the subtlety of fine-grained complexity classification, consider the following statements in the ontology:

$$\forall x (\text{Hand}(x) \rightarrow \exists^{\neq 5} y \text{hasFinger}(x, y)) \quad (1)$$

$$\forall x (\text{Hand}(x) \rightarrow \exists y (\text{hasFinger}(x, y) \wedge \text{Thumb}(y))) \quad (2)$$

The language features used here can in principle express CONP-hard properties. But are the concrete statements (1) and (2) computationally costly? It turns out that an ontology that contains only statement (1) enjoys PTIME query evaluation and the same is true for an ontology that contains only statement (2). In contrast, an ontology that contains both (1) and (2) is CONP-hard. 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.

Aim of Paper. An overarching framework for many ontology languages, tractable or not, is provided by the *guarded fragment* of first-order logic (GF) [2, 41, 42, 66] and its

extension with counting quantifiers and other forms of counting [46, 68]. In particular, most description logics fall within GF or its extensions with counting. Sometimes, also GF itself is considered as an ontology language for ontology-mediated querying [9, 69]. Subsuming many relevant ontology languages, GF and its extensions are well suited for general studies of the data complexity and rewritability of ontology-mediated queries, in the sense of fine-grained complexity on the level of ontologies discussed above. For example, any dichotomy result obtained for the data complexity of ontology-mediated querying with GF and its extensions is inherited by all DLs that fall within these logics. The main aim of this paper is to *identify as large as possible 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 ontology languages, considering conjunctive queries (CQs) and unions thereof (UCQs) as the actual query language.* We additionally aim to provide insight into the following questions:

- (1) Which fragments of GF (with and without counting) do *not* admit a dichotomy between PTIME and CONP?
- (2) What is the relationship between PTIME data complexity and rewritability into Datalog—and into Datalog with inequality in rule bodies in case we start from GF with forms of counting?
- (3) Is it decidable whether a given ontology enjoys PTIME data complexity?

We concentrate on a fragment of GF that is invariant under disjoint union, which we call *uGF*, and on fragments thereof and their extension with forms of counting. The primary definition of uGF is syntactical: 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 sentences as subformulas and in which equality is not used as a guard. However, uGF is not just *some* fragment that is closed under disjoint union, but it is characterized by this semantic property in the following sense: we show that a sentence φ of GF is invariant under disjoint union if and only if it is equivalent to a sentence of uGF.

The reason for concentrating on uGF is two-fold. On the one hand, the expressive power of GF that does not fall within uGF seems to be of marginal importance for ontology engineering and almost all DLs that fall within GF also fall within uGF; an exception are DLs with the universal role. On the other hand, going beyond uGF brings in significant technical complications that appear to make the technical development very cumbersome. We will point out the concrete advantages of uGF over GF throughout the paper and confine ourselves to an example here: for every uGF ontology \mathcal{O} , UCQ evaluation w.r.t. \mathcal{O} is in PTIME if and only if CQ evaluation is, but there are GF ontologies for which this is not the case. We do not intend to make strong claims about the practical utility of uGF as an ontology language; its main virtue is that it encompasses many relevant DLs and thus many relevant ontologies, in this way providing a suitable and general framework for a fine-grained complexity analysis.

Obtained Results. Our dichotomy results are obtained by two different approaches: via direct, fully self-contained proofs based on a carefully designed technical machinery that centers on the notions of materializability and unraveling tolerance discussed in more detail below, and via reduction to CSP. In the latter approach, we take advantage of the recently established CSP dichotomy between PTIME and NP-complete, formerly known as the Feder-Vardi conjecture [33] and in 2017 confirmed independently in [18] and [74] using algebraic methods; see [49] for an overview of the state of the art just before the proof of the conjecture. Apart from admitting much more transparent proofs, the first approach also establishes stronger guarantees. Whenever it is applicable, it additionally allows us to

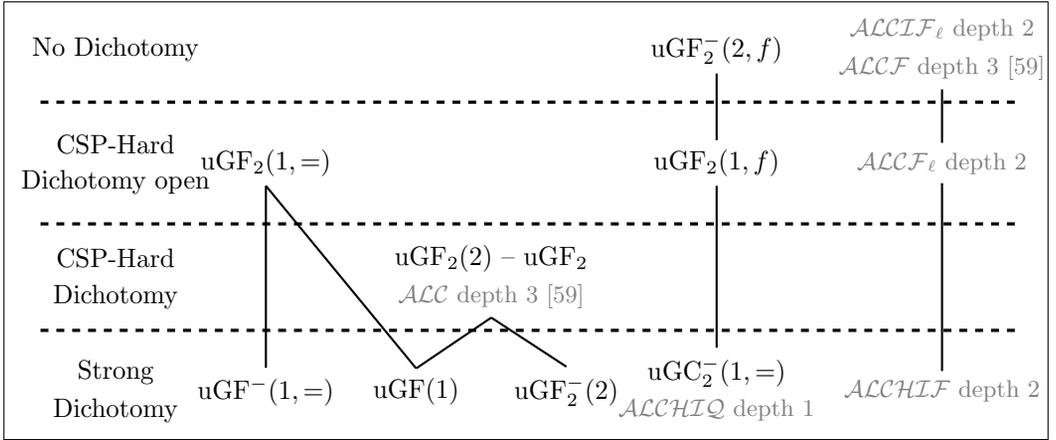


Fig. 1. Summary of results—Number in brackets indicates depth, f presence of partial functions, \cdot_2 restriction to two variables, \cdot^- restricts outermost guards to be equality, \mathcal{F} globally function roles, \mathcal{F}_ℓ concepts ($\leq 1R$).

prove that PTIME query evaluation coincides with rewritability into Datalog, admitting inequality in the rule bodies whenever we start from a fragment with equality or counting. To reflect this, we refer to dichotomies established by the first approach as *strong dichotomies*. For other fragments, we observe a close connection to CSPs—expressed via the notion of *CSP-hardness* whose (subtle) definition is omitted here and given in the paper. What is important, however, is that for CSP-hard fragments a dichotomy between PTIME and CONP-complete implies the CSP dichotomy whose proof requires highly intricate algebraic considerations. In the case of CSP-hardness (and also when we establish that there is no dichotomy between PTIME and CONP-complete), PTIME query evaluation does provably *not* coincide with rewritability into Datalog.

The main results established in this paper are summarized in Figure 1. We first explain the fragments shown in the figure and then survey the obtained results. A main parameter that we vary is the *depth* of uGF sentences $\forall \vec{x}(R(\vec{x}) \rightarrow \varphi(\vec{x}))$, defined as the quantifier depth of $\varphi(\vec{x})$ and thus not counting the outermost universal quantifier. In real world ontologies, the depth is typically very small, mostly only one and very rarely larger than two. 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, as studied for example in [46, 68]. While guarded fragments are displayed in black, description logics (DLs) are shown in gray. We use standard DL names except that ‘ \mathcal{F} ’ denotes globally functional roles while ‘ \mathcal{F}_ℓ ’ refers to (locally) counting concepts of the form ($\leq 1R$). We do not explain DL names at this point and instead refer the reader to Section 2.3 and the textbook [8].

The bottommost row of Figure 1 displays fragments for which there is a strong dichotomy between PTIME and CONP, the second row shows cases that admit a mutual reduction with the CSP dichotomy, the third row has fragments that are CSP-hard, but for which a dichotomy remains open, and the topmost part is for fragments that provably have no

dichotomy (unless $\text{PTIME} = \text{NP}$). Informally, the bottommost row thus states upper bounds while the topmost two rows state lower bounds; the second row from the bottom states both upper and lower bounds and this is why we use a range of logics there. The vertical lines indicate that the linked results are closely related, often indicating a fundamental difficulty in further generalizing an upper bound. For example, $\text{uGF}(1)$ enjoys strong dichotomy while $\text{uGF}_2(2)$ and $\text{uGF}_2(1, =)$ are CSP-hard and thus the former result is optimal in the sense that it can neither be generalized to depth two nor to the case where equality is not restricted to guards. All displayed results hold both when CQs and when UCQs are used as the actual query; here, the use of uGF rather than GF as an ontology language pays off as there are GF ontologies for which CQ evaluation is in PTIME while UCQ-evaluation is CONP -hard. The results shown in Figure 1 close a number of open problems from [59] such as that \mathcal{ALCC} ontologies of depth 2 enjoy a strong dichotomy.

We also prove a number of results that are not reflected in Figure 1. In particular we show that for \mathcal{ALCHIQ} ontologies of depth 1, it is decidable and EXPTIME -complete whether a given ontology \mathcal{O} admits PTIME query evaluation, which is equivalent to rewritability into Datalog with inequality in rule bodies, and whether query evaluation w.r.t. \mathcal{O} is CONP -hard. For $\text{uGC}_2^-(1, =)$, we show a NEXPTIME upper bound. Moreover, for ontologies formulated in $\text{uGF}_2^-(2, f)$ and in \mathcal{ALCCF}_ℓ of depth 2, we prove these problems to be undecidable.

Practical Relevance. To get a first idea of the practical relevance of our results, we have analyzed 411 ontologies from the BioPortal repository [73]. After removing all constructors that do not fall within \mathcal{ALCHIF} , a remarkable 405 ontologies turned out to have depth 2 and thus belong to a fragment with dichotomy (sometimes modulo a straightforward complexity-preserving rewriting). For \mathcal{ALCHIQ} , still 385 ontologies had depth 1 and so belonged to a fragment with dichotomy. As our initial examples (1) and (2) illustrate, ontology languages with counting induce particular subtleties regarding PTIME query evaluation. To better understand the situation regarding counting statements in practical ontologies, we have analyzed each single axiom of depth 1 that uses counting in the BioPortal ontology. We found 5081 such axioms. For the vast majority of these (4975) we established that query evaluation is in PTIME , but only 2911 are preserved under direct products, a necessary condition for being equivalent to a First-order Horn sentence [26]. Thus, at most 2911 axioms are equivalent to sentences contained in languages designed for tractable query evaluation. While the restriction to single axioms is unrealistic in practice and should be extended to whole ontologies, this nevertheless indicates that it can pay off to analyze the complexity on the level of ontologies rather than on the level of ontology languages.

Techniques Used. We briefly highlight some of the techniques used to establish our results, in particular for proving strong dichotomy. In the first approach to proving dichotomy results mentioned above, an important role is played by the notions of materializability and unraveling tolerance of an ontology \mathcal{O} , first introduced in [59]. Materializability means that for every instance \mathfrak{D} , there is a universal model \mathfrak{A} of \mathfrak{D} and \mathcal{O} in the sense that \mathfrak{A} gives exactly the same answers to all queries that are also given by \mathfrak{D} and \mathcal{O} (under the certain answer semantics commonly adopted for ontology-mediated querying). Unraveling tolerance of an ontology \mathcal{O} means that when \mathcal{O} is combined with a query that is tree-like, then the resulting ontology-mediated query cannot distinguish between an instance and its unraveling into a structure of bounded treewidth. We show that non-materializability of \mathcal{O} implies that query evaluation w.r.t. \mathcal{O} is CONP -hard (a property that fails when replacing uGF with GF) and that unraveling tolerance of \mathcal{O} implies that query evaluation w.r.t. \mathcal{O} is rewritable into Datalog and thus in PTIME . To establish strong dichotomy of a fragment, we then prove that

for the ontologies formulated in it, materializability implies unraveling tolerance; depending on the fragment, these proofs can be technically rather subtle. We also make the interesting observation that preservation of an ontology \mathcal{O} under direct products implies unraveling tolerance. As all first-order ontology languages that admit PTIME query evaluation fall within first-order Horn logic and first-order Horn logic is preserved under direct products [26], unraveling tolerance provides a uniform explanation of the good computational behavior of these languages.

In the second approach to proving dichotomy results mentioned above, we reduce ontology-mediated querying to constraint satisfaction problems with a fixed template [33]. The reduction can be rather subtle and requires the use of an extended version of CSPs that ‘admit precoloring’, that is, in which some targets of the homomorphism into the CSP template can be preassigned in the input. The same problems that make us use CSPs with precoloring also emerges when proving non-dichotomy results, where it poses serious challenges. To tackle them, we establish a variation of Ladner’s theorem, which establishes the existence of NP-intermediate problems, that is, of problems that are neither in PTIME nor NP-hard [52]. Instead of speaking about the word problem for NP Turing machines, our variation shows that there is an NP-intermediate *run fitting problem*, which is to decide whether a given partially described run of a Turing machine (that informally corresponds to a precoloring in the CSP case) can be extended to a full run which is accepting. We then prove the non-dichotomy results by first constructing an ontology that checks whether a database instance has the shape of a grid and then adding an ontology that checks whether the partial run encoded by the database instance can be extended to an accepting run. The first ontology is also used to prove that materializability, rewritability into Datalog[≠], PTIME query evaluation, and CONP query evaluation are undecidable (unless PTIME = NP), by a reduction of the finite rectangle tiling problem.

Our strategy for proving that (in some cases) it is decidable whether an ontology admits PTIME query evaluation is as follows. We first establish that it suffices to decide materializability for database instances that have the shape of a tree of depth one and are of size polynomial in the size of the ontology. We then show how partial materializations can be composed to obtain full materializations using a ‘mosaic technique’ from modal logic.

Overview of Paper. This paper is organized as follows. In Section 2, we introduce fundamental notation and the relevant ontology languages including fragments of GF and uGF as well as several description logics. We also introduce guarded bisimulations and guarded tree decompositions as essential technical tools used throughout the paper; as many of our technical notions, they come in a non-counting version and in a counting version. Section 3 introduces and studies materializability. We show that materializability does not depend on whether we use CQs, UCQs, or rAQs as actual queries where rAQs (for *rooted acyclic queries*) are a class of tree-like CQs. In contrast, whether a concrete model of the instance and ontology is a materialization or not *does* depend on the query language. We also analyze the relationship between universal models defined in terms of query answers and universal models defined in terms of homomorphisms, as well as the relationship of these notions to a certain *disjunction property*. Finally, we show that tractability of query evaluation does not depend on the query language used. Section 4 is concerned with unraveling tolerance, our main result being that unraveling tolerance implies rewritability into Datalog (with inequalities, when appropriate). In addition, we show that preservation under direct products implies unraveling tolerance. Section 5 brings together materializability and unraveling tolerance to establish strong dichotomy results. In Section 6,

we establish connections to CSP, proving both lower bounds (that is, CSP-hardness) and upper bounds (that is, dichotomy results by reduction to CSP). For the latter, we in fact make a detour via the logical generalization MMSNP of CSP introduced by Feder and Vardi [33]. In Section 7, we establish undecidability results regarding PTIME and CONP query evaluation, Datalog rewritability, and materializability. The techniques developed here are the basis for the non-dichotomy results proved in Section 8 where we also show non-dichotomy for the run-fitting problem. Section 9 is the final technical section, concerned with fragments for which PTIME query evaluation is decidable.

2 PRELIMINARIES

We start with introducing the basics of ontology-mediated querying, then define the ontology languages relevant to this paper and afterwards introduce several elementary technical notions including guarded bisimulations and guarded tree decomposition. We also establish some fundamental lemmas regarding the latter.

2.1 Basics of Ontology-Mediated Querying

We assume a countably infinite set Δ of *constants* and a set Σ of relation symbols that contains a countably infinite set of relation symbols of any arity ≥ 1 . A (*database*) *instance* \mathfrak{D} is a non-empty set of *facts* $R(a_1, \dots, a_k)$, where $R \in \Sigma$, k is the arity of R , and $a_1, \dots, a_k \in \Delta$. We generally assume that instances are finite, unless otherwise specified. An *interpretation* \mathfrak{A} is a non-empty (and potentially infinite) set of facts. We use $\text{sig}(\mathfrak{A})$ and $\text{dom}(\mathfrak{A})$ to denote the set of relation symbols and constants used in \mathfrak{A} , respectively. We generally assume that $\text{sig}(\mathfrak{A})$ is finite while $\text{dom}(\mathfrak{A})$ can be infinite. Whenever convenient, interpretations \mathfrak{A} are presented in the form $(A, (R^{\mathfrak{A}})_{R \in \text{sig}(\mathfrak{A})})$ where $A = \text{dom}(\mathfrak{A})$ and $R^{\mathfrak{A}}$ is a k -ary relation on A for each $R \in \text{sig}(\mathfrak{A})$ of arity k . While instances are syntactic objects used to represent a database, interpretations are semantic objects; though from a formal perspective, every instance is also an interpretation. Formally, an interpretation \mathfrak{A} is a *model* of an instance \mathfrak{D} , written $\mathfrak{A} \models \mathfrak{D}$, if $\mathfrak{D} \subseteq \mathfrak{A}$. By adopting this notion of being a model, we make a strong open world assumption since interpretations can make true additional facts and contain additional constants; moreover, it implies *standard names* [54], that is, every constant in \mathfrak{D} is interpreted as itself in \mathfrak{A} . While this is not the standard semantics of constants in FO, it is standard in ontology-mediated querying (and without counting, the two semantics result in the same answers to all queries).

Assume that \mathfrak{A} and \mathfrak{B} are interpretations. A *homomorphism* h from \mathfrak{A} to \mathfrak{B} is a mapping from $\text{dom}(\mathfrak{A})$ to $\text{dom}(\mathfrak{B})$ such that $R(a_1, \dots, a_k) \in \mathfrak{A}$ implies $R(h(a_1), \dots, h(a_k)) \in \mathfrak{B}$ for all $a_1, \dots, a_k \in \text{dom}(\mathfrak{A})$ and $R \in \Sigma$ of arity k . We say that h *preserves a set* D of constants if $h(a) = a$ for all $a \in D$ and that h is an *isomorphic embedding* if it is injective and $R(h(a_1), \dots, h(a_k)) \in \mathfrak{B}$ implies $R(a_1, \dots, a_k) \in \mathfrak{A}$. An interpretation $\mathfrak{A} \subseteq \mathfrak{B}$ is a *subinterpretation* of \mathfrak{B} if $R(a_1, \dots, a_k) \in \mathfrak{B}$ and $a_1, \dots, a_k \in \text{dom}(\mathfrak{A})$ implies $R(a_1, \dots, a_k) \in \mathfrak{A}$; if $\text{dom}(\mathfrak{A}) = A$, 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 $\vec{x} = (x_1, \dots, x_k)$ is the tuple of *answer variables* of q , and ϕ is a conjunction of *atomic formulas* $R(y_1, \dots, y_n)$ with $R \in \Sigma$ of arity n and y_1, \dots, y_n variables. As usual, we assume that every variable in \vec{x} occurs in some atomic formula of ϕ . Any CQ $q(\vec{x}) \leftarrow \phi$ can be regarded as an instance \mathfrak{D}_q , called the *canonical database of q* , whose facts are exactly the atomic formulas of q with variables viewed as constants. A tuple $\vec{a} = (a_1, \dots, a_k)$ of constants is an *answer to $q(x_1, \dots, x_k)$ in \mathfrak{A}* , in symbols $\mathfrak{A} \models q(\vec{a})$, if there is a homomorphism h from \mathfrak{D}_q to \mathfrak{A} such

that $h(x_i) = a_i$ for $1 \leq i \leq k$. A *union of conjunctive queries (UCQ)* q takes the form $q_1(\vec{x}), \dots, q_n(\vec{x})$, 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. Let $\text{FO}(=)$ denote the set of all first-order sentences over signature Σ , admitting equality but neither constants nor function symbols. An *ontology language* \mathcal{L} is a set of $\text{FO}(=)$ sentences 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 as well as several description 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} . We use $\text{sig}(\mathcal{O})$ to denote the set of relation symbols used in \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 $\text{dom}(\mathfrak{D})$ is a *certain answer to q on \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 $\text{dom}(\mathfrak{D})^k$, whether $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$. Note that ontology-mediated querying is a form of querying under (open world) constraints, a traditional topic in database theory, see e.g. [11, 12] and references therein. It is also related to deductive databases, see e.g. the monograph [63] and to query answering under views [23, 24].

We use standard notation for Datalog programs [1, 25]. A *Datalog \neq rule* ρ takes the form

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

where S is a relation symbol from Σ , $m \geq 1$, and R_1, \dots, R_m are either relation symbols from Σ or the symbol \neq for inequality. We call $S(\vec{x})$ the *head* of ρ and $R_1(\vec{x}_1) \wedge \dots \wedge R_m(\vec{x}_m)$ its *body*. Every variable in the head of ρ is required to occur in its body. A *Datalog rule* is a Datalog \neq rule that does not use inequality. A *Datalog \neq program* is a finite set Π of Datalog \neq rules with a selected relation symbol goal that does not occur in rule bodies in Π . The *arity* of Π is the arity of its goal relation symbol; we say that Π is *Boolean* if it has arity zero. Relation symbols that occur in the head of at least one rule of Π are *intensional* and all remaining relation symbols in Π are *extensional*. Note that, by definition, goal is an intensional relation symbol. A *Datalog program* is a Datalog \neq program that does not use 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 $\forall \vec{x}_1 \dots \forall \vec{x}_m (R_1(\vec{x}_1) \wedge \dots \wedge R_m(\vec{x}_m) \rightarrow S(\vec{x}))$ with $S(\vec{x}) \leftarrow R_1(\vec{x}_1) \wedge \dots \wedge R_m(\vec{x}_m) \in \Pi$. We set $\mathfrak{D} \models \Pi(\vec{a})$ if $\text{goal}(\vec{a}) \in \mathfrak{A}$ for all models \mathfrak{A} of \mathfrak{D} and Π .

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 $\vec{a} \in \text{dom}(\mathfrak{D})$, $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ iff $\mathfrak{D} \models \Pi(\vec{a})$. *Datalog \neq -rewritability* is defined accordingly. We are mainly interested in the following properties of ontologies.

Definition 2.1. 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 data complexity.
- *\mathcal{Q} -evaluation w.r.t. \mathcal{O} is Datalog-rewritable (resp. Datalog \neq -rewritable)* if for every $q \in \mathcal{Q}$, the OMQ (\mathcal{O}, q) is Datalog-rewritable (resp. Datalog \neq -rewritable).

- *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 in data complexity.

2.2 The Guarded Fragment of FO

As ontology languages, we consider fragments of the guarded fragment of FO, the two-variable guarded fragment of FO with counting, and several description logics. We start with introducing the former.

Guarded formulas [2] 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*. To emphasize that we admit equality in non-guard positions we denote the set of all guarded formulas by $\text{GF}(=)$. The fragment in which no equality is admitted in non-guard positions is denoted GF. For example, let

$$\varphi_1 = \forall x(x = x \rightarrow \exists y, z(R(x, y, z) \wedge x = y)) \quad \text{and} \quad \varphi_2 = \forall x(x = x \rightarrow \exists y, z(R(x, y, z) \wedge S(x, y))).$$

Then φ_1 is in $\text{GF}(=)$ and not in GF while φ_2 is in GF.

As discussed in the introduction, in this article we focus on the fragment uGF of GF, defined next. The expressive power that we give up seems rather modest from an ontology engineering point of view and in the next subsection we observe that several important DLs fall within uGF.

We use *openGF* to denote the fragment of $\text{GF}(=)$ that consists of all open formulas whose subformulas are all open and in which equality is not used as a guard. The fragment $\text{uGF}(=)$ of $\text{GF}(=)$ is the set of sentences obtained from openGF by a *single guarded universal quantifier*: if $\varphi(\vec{y})$ is in openGF, then $\forall \vec{y}(\alpha(\vec{y}) \rightarrow \varphi(\vec{y}))$ is in $\text{uGF}(=)$, where $\alpha(\vec{y})$ is an atomic formula or an equality $y = y$ that contains all variables in \vec{y} . With uGF, we mean the fragment of $\text{uGF}(=)$ in which equality is only admitted as a guard for the outermost universal quantifier. For example, the formulas φ_1 and φ_2 above are $\text{uGF}(=)$ and in uGF, respectively, while the following formulas are in GF, but not in $\text{uGF}(=)$:

- (1) $\exists x(A(x) \wedge \neg B(x))$ because it starts with an existential quantifier;
- (2) $\forall x(A(x) \rightarrow \exists y(A(y) \wedge \neg B(y)))$ because it contains the sentence $\exists y(A(y) \wedge \neg B(y))$ as a proper subformula.

We often omit equality guards in $\text{uGF}(=)$ sentences of the form $\forall y(y = y \rightarrow \varphi(y))$ and simply write $\forall y \varphi$ instead although syntactically the latter formula need not even fall within GF.

The foremost technical property of $\text{uGF}(=)$ sentences φ is that they 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, $\bigcup_{i \in I} \mathfrak{B}_i \models \varphi$. Up to equivalence, this property actually exactly characterizes $\text{uGF}(=)$. The proof of the following result is given in the appendix.

THEOREM 2.2. *A sentence in $\text{GF}(=)$ (resp. GF) is invariant under disjoint unions iff it is equivalent to a sentence in $\text{uGF}(=)$ (resp. uGF).*

We next give some example ontologies that take the form of Boolean combinations of uGF sentences and are not invariant under disjoint unions. We shall come back to these ontologies later on to explain the technical convenience of $\text{uGF}(=)$ compared to $\text{GF}(=)$.

Example 2.3. Let

$$\begin{aligned}\mathcal{O}_{\text{UCQ/CQ}} &= \{\forall x(A(x) \vee B(x)) \vee \exists xE(x)\} \\ \mathcal{O}_{\text{Mat/PTime}} &= \{\forall xA(x) \vee \forall xB(x)\}.\end{aligned}$$

Then $\mathcal{O}_{\text{Mat/PTime}}$ is not preserved under disjoint unions since $\mathfrak{D}_1 = \{A(a)\}$ and $\mathfrak{D}_2 = \{B(b)\}$ are models of $\mathcal{O}_{\text{Mat/PTime}}$ but $\mathfrak{D}_1 \cup \mathfrak{D}_2$ refutes $\mathcal{O}_{\text{Mat/PTime}}$; $\mathcal{O}_{\text{UCQ/CQ}}$ 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 $\mathcal{O}_{\text{UCQ/CQ}}$ but \mathfrak{D}'_2 refutes $\mathcal{O}_{\text{UCQ/CQ}}$.

When studying uGF(=) and 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 . Thus, an openGF formula has depth 1 if no guarded quantifier occurs within the scope of another guarded quantifier. The *depth* of a sentence $\forall \vec{y}(\alpha(\vec{y}) \rightarrow \varphi(\vec{y}))$ in uGF(=) is the depth of $\varphi(\vec{y})$, thus the outermost guarded quantifier does not contribute to the depth. The *depth* of a uGF(=) ontology is the maximum depth of the sentences in it. We indicate restricted depth in brackets, writing e.g. uGF(1) to denote the set of uGF sentences of depth at most 1 and uGF(2, =) to denote the set of all uGF(=) sentences of depth at most 2. For example, the sentence

$$\varphi_3 = \forall x, y(R(x, y) \rightarrow (A(x) \vee \exists zS(y, z)))$$

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

We observe that, modulo normalization, uGF(1) has the same expressive power as GF. A GF sentence is in *Scott normal form* if it is a conjunction of sentences of one of the forms

$$\exists \vec{x}(\alpha_0(\vec{x}) \wedge \psi_0(\vec{x})) \quad \forall \vec{x}(\alpha_0(\vec{x}) \rightarrow \psi_0(\vec{x})) \quad \forall \vec{x}(\alpha_0(\vec{x}) \rightarrow \exists \vec{y}(\alpha_1(\vec{y}, \vec{z}) \wedge \psi_1(\vec{y}, \vec{z})))$$

where $\alpha_0(\vec{x})$ and $\alpha_1(\vec{y}, \vec{z})$ are atomic formulas or equalities and $\psi_0(\vec{x})$ and $\psi_1(\vec{y}, \vec{z})$ are quantifier free. It follows from [41] that for every GF sentence φ one can construct in polynomial time a GF sentence ψ in Scott normal form that is a conservative extension of φ , that is, $\psi \models \varphi$ and every model of φ can be expanded to a model of ψ by interpreting the fresh symbols in ψ [41]. Replace any sentence in ψ of the form

- $\exists \vec{x}(\alpha_0(\vec{x}) \wedge \psi_0(\vec{x}))$ by the sentence $\forall x(x = x \rightarrow \exists \vec{x}(R(x, \vec{x}) \wedge \alpha_0(\vec{x}) \wedge \psi_0(\vec{x})))$, where R is a fresh relation symbol of arity $|\vec{x}| + 1$, and x a fresh variable; and
- $\forall \vec{x}(\alpha_0(\vec{x}) \rightarrow \exists \vec{y}(\alpha_1(\vec{y}, \vec{z}) \wedge \psi_1(\vec{y}, \vec{z})))$ in which $\exists \vec{y}(\alpha_1(\vec{y}, \vec{z}) \wedge \psi_1(\vec{y}, \vec{z}))$ is closed by the sentence $\forall \vec{x}(\alpha_0(\vec{x}) \rightarrow \exists \vec{y}(R(\vec{x}, \vec{y}) \wedge \alpha_1(\vec{y}, \vec{z}) \wedge \psi_1(\vec{y}, \vec{z})))$, where R is a fresh relation symbol of arity $|\vec{x}| + |\vec{y}|$.

The set of conjuncts of the resulting formula is an ontology in uGF(1) and a conservative extension of φ . Thus, the satisfiability and CQ evaluation problems for full GF can be reduced in polynomial time 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, and uGF⁻ denotes the corresponding fragment of uGF. Thus, the sentence φ_3 above 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:

$$\forall x(\exists y(R(y, x) \wedge \neg A(y)) \rightarrow \exists zS(x, z)).$$

An example of a uGF sentence of depth 1 that is not equivalent to a uGF⁻ sentence of depth 1 is given in the following example.

Example 2.4. Let

$$\varphi = \forall x, y (R(x, y) \rightarrow (\exists z (S(x, z) \wedge A(z)) \rightarrow \exists z (S(y, z) \wedge B(z))))$$

Then φ is a uGF sentence of depth 1 but easily seen to be not equivalent to any uGF⁻ sentence of depth 1.

Informally, uGF sentences of depth 1 can be thought of as uGF⁻ sentences of ‘depth 1.5’ because giving up \cdot^- admits an additional level of ‘real’ quantification over guards that are not forced to be equality.

The two-variable fragment of uGF(=) is denoted with uGF₂(=). More precisely, in uGF₂(=) 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 uGF₂(=) and uGF₂ with forms of counting. First, uGF₂(f) denotes the extension of uGF₂ with function symbols, that is, an uGF₂(f) ontology is a finite set of uGF₂ sentences and of *functionality axioms* $\forall x \forall y_1 \forall y_2 ((R(x, y_1) \wedge R(x, y_2)) \rightarrow (y_1 = y_2))$, see also [41]. Second, we consider the extension uGC₂(=) of uGF₂(=) with counting quantifiers. More precisely, the language openGC₂ is defined in the same way as the two-variable fragment of openGF, but in addition admits *guarded counting quantifiers* as in [46, 68]: if $n \in \mathbb{N}$, $\{z_1, z_2\} = \{x, y\}$, $\alpha(z_1, z_2) \in \{R(z_1, z_2), R(z_2, z_1)\}$ for some $R \in \Sigma$, and $\varphi(z_1, z_2)$ is in openGC₂, then $\exists^{\geq n} z_1 (\alpha(z_1, z_2) \wedge \varphi(z_1, z_2))$ is in openGC₂(=). The ontology language uGC₂(=) is then defined in the same way as uGF₂(=), using openGC₂ instead of openGF₂. Whenever convenient we regard openGC₂ and uGC₂ as fragments of FO(=). The *depth* of formulas in uGC₂(=) 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, uGF₂⁻(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.

2.3 Description Logics

Description logics are a popular family of ontology languages that are closely related to the guarded fragments of FO(=) introduced above [7]. In the following, we give a brief introduction to the syntax and semantics of several relevant DLs and establish their relationship to these fragments. We concentrate on the DL \mathcal{ALC} and its extensions by inverse roles, role inclusions, qualified number restrictions, functional roles, and local functionality. \mathcal{ALC} -concepts are constructed according to the rule

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

where A and R range over unary and binary relation symbols, respectively. In DL parlance, unary relation symbols are also called *concept names* and binary relation symbols are also called *roles*, but in this paper we shall mostly speak of relation symbols.

DLs extended by *inverse roles* (denoted in the name of a DL by the letter \mathcal{I}) admit, in addition, *inverse relation symbols* denoted by R^- , with R a relation symbol. In \mathcal{ALCI} , we thus have available the additional expressions $\exists R^-.C$ and $\forall R^-.C$ for constructing concepts. DLs extended by *qualified number restrictions* (denoted by \mathcal{Q}) additionally admit concepts of the form $(\geq n R C)$ and $(\leq n R C)$, where $n \geq 0$ is a natural number, R a relation symbol or an inverse relation symbol (provided that inverse relation symbols are admitted in the original DL), and C is a concept. When extending a DL with *local functionality* (denoted by \mathcal{F}_ℓ) one can use only number restrictions of the form $(\leq 1 R \top)$. We abbreviate

the \mathcal{ALCF}_ℓ concept $(\leq 1 R \top)$ by $(\leq 1R)$, $(\exists R.\top) \sqcap (\leq 1R)$ by $(= 1R)$, and $\neg(\leq 1R)$ by $(\geq 2R)$, respectively.

In DLs, ontologies are formalized as finite sets of *concept inclusions* $C \sqsubseteq D$, where C, D are concepts. We use the *concept equivalence* $C \equiv D$ as an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. In DLs extended with *functionality* (denoted by \mathcal{F}) one can also use *functionality assertions* $\text{func}(R)$ in the ontology, where R is a relation symbol or an inverse relation symbol (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 $R \sqsubseteq S$ in the ontology, where R and S are relation symbols or inverse relation symbols (if present in the original DL), and which state that R is a subset of S . Note that while inverse roles, qualified number restrictions, and local functionality affect the concept language, functionality assertions and role inclusions only take effect on the level of ontologies. So when we work for example with \mathcal{ALCHF}_ℓ , then the concepts are formed in \mathcal{ALCF}_ℓ and, additionally, role inclusions are admitted in the ontology.

The semantics of DLs is defined in terms of interpretations \mathfrak{A} . Given \mathfrak{A} , the interpretation $C^{\mathfrak{A}}$ of a concept C , $R^{\mathfrak{A}}$ of a relation symbol R , and $(R^-)^{\mathfrak{A}}$ of an inverse relation symbol R^- is defined inductively as follows:

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

\mathfrak{A} satisfies a concept inclusion $C \sqsubseteq D$ if $C^{\mathfrak{A}} \subseteq D^{\mathfrak{A}}$, a functionality assertion $\text{func}(R)$ if $R^{\mathfrak{A}}$ is a partial function, and a rule inclusion $R \sqsubseteq S$ if $R^{\mathfrak{A}} \subseteq S^{\mathfrak{A}}$.

The semantics of DL concepts C can alternatively be given by translation to openGC formulas $C^\dagger(x)$ with one free variable x and two variables overall. For simplicity, we only give the translation explicitly for DLs without inverse roles:

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

A concept inclusion $C \sqsubseteq D$ then translates to the uGC₂ sentence $\forall x (C^\dagger(x) \rightarrow D^\dagger(x))$ and also with inverse roles and when adding role hierarchies and functionality assertions, we remain within uGC₂.

The *depth* of a concept is the maximal nesting depth of its quantifiers. The *depth* of an ontology is the maximum depth of concepts that occur in it. Thus, every \mathcal{ALC} ontology of depth n is a uGF₂⁻ ontology of depth n . When translating into uGF₂ instead of into uGF₂⁻, the depth might decrease by one because one can exploit the outermost quantifier (which does not contribute to the depth).

Example 2.5. The \mathcal{ALC} concept inclusion $\exists S.A \sqsubseteq \forall R.\exists S.B$ has depth 2, but it is equivalent to the $\text{uGF}_2(1)$ sentence

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

from Example 2.4.

In all of the DLs considered in this paper, any ontology \mathcal{O} can straightforwardly be converted in polynomial time into an ontology \mathcal{O}^* of depth 1 that is a conservative extension of \mathcal{O} . In fact, many DL algorithms for satisfiability and for query evaluation assume that the ontology is of depth one and in a normalized form [10, 71].

We observe the following relationships between DLs and fragments of GC_2 . For a DL \mathcal{L} and fragment \mathcal{L}' of GC_2 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. Then the following 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 $\text{uGF}_2^-(2)$ ontology.
- (2) Every \mathcal{ALCHIF} ontology can be written as a $\text{uGF}_2^-(f)$ 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 $\text{uGC}_2^-(1)$ ontology.

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 symbols, variable names, and so on as a single symbol and assuming that numbers in counting quantifiers and their DL counterpart, qualified number restrictions, are coded in unary. The latter assumption is relevant only for the results obtained in Section 9.

2.4 Guarded Bisimulations

We define guarded bisimulations, a standard tool for proving that two interpretations satisfy the same guarded formulas [42]. Our use of the fragment $\text{uGF}(=)$ of GF allows us to slightly modify the standard notion by considering, in the back and forth conditions, only guarded sets that overlap the current guarded set. To cover $\text{uGC}_2(=)$ we introduce *counting* guarded bisimulations.

Let \mathfrak{A} be an interpretation. It will be convenient to use the notation $[\vec{a}] = \{a_1, \dots, a_n\}$ to denote the set of components of the tuple $\vec{a} = (a_1, \dots, a_n) \in \text{dom}(\mathfrak{A})^n$. A set $G \subseteq \text{dom}(\mathfrak{A})$ is *guarded* in \mathfrak{A} if G is a singleton or there are $R \in \Sigma$ and $R(\vec{a}) \in \mathfrak{A}$ such that $G = [\vec{a}]$. By $S(\mathfrak{A})$, we denote the set of all guarded sets in \mathfrak{A} . A tuple $\vec{a} \in \text{dom}(\mathfrak{A})^n$ is *guarded* in \mathfrak{A} if $[\vec{a}]$ is a subset of some guarded set in \mathfrak{A} .

For tuples $\vec{a} = (a_1, \dots, a_n)$ in \mathfrak{A} and $\vec{b} = (b_1, \dots, b_n)$ in \mathfrak{B} we call a mapping p from $[\vec{a}]$ to $[\vec{b}]$ with $p(a_i) = b_i$ for $1 \leq i \leq n$ (written $p : \vec{a} \mapsto \vec{b}$) a *partial isomorphism* if p is an isomorphism from $\mathfrak{A}_{[\vec{a}]}$ to $\mathfrak{B}_{[\vec{b}]}$. A set I of partial isomorphisms $p : \vec{a} \mapsto \vec{b}$ from guarded tuples \vec{a} in \mathfrak{A} to guarded tuples \vec{b} in \mathfrak{B} is called a *connected guarded bisimulation* if the following hold for all $p : \vec{a} \mapsto \vec{b} \in I$:

- (i) for every guarded tuple \vec{a}' in \mathfrak{A} with $[\vec{a}] \cap [\vec{a}'] \neq \emptyset$ there exists a guarded tuple \vec{b}' in \mathfrak{B} and $p' : \vec{a}' \mapsto \vec{b}' \in I$ such that p' and p coincide on $[\vec{a}] \cap [\vec{a}']$.
- (ii) for every guarded tuple \vec{b}' in \mathfrak{B} with $[\vec{b}] \cap [\vec{b}'] \neq \emptyset$ there exists a guarded tuple \vec{a}' in \mathfrak{A} and $p' : \vec{a}' \mapsto \vec{b}' \in I$ such that p'^{-1} and p^{-1} coincide on $[\vec{b}] \cap [\vec{b}']$.

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 $p : \vec{a} \mapsto \vec{b}$. Connected guarded bisimulations differ from the standard guarded bisimulations [42] in additionally requiring $[\vec{a}] \cap [\vec{a}'] \neq \emptyset$ in Condition (i) and $[\vec{b}] \cap [\vec{b}'] \neq \emptyset$ in Condition (ii). These conditions are intimately linked to the definition of uGF and in particular to the fact that openGF formulas cannot contain sentences as subformulas.

LEMMA 2.6. *Let \mathfrak{A} and \mathfrak{B} be interpretations.*

- (1) *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 $\mathfrak{B} \models \varphi(\vec{b})$.*
- (2) *If for every guarded \vec{a} in $\text{dom}(\mathfrak{A})$ there exists a guarded \vec{b} in $\text{dom}(\mathfrak{B})$ such that (\mathfrak{A}, \vec{a}) and (\mathfrak{B}, \vec{b}) are connected guarded bisimilar and vice versa, then \mathfrak{A} and \mathfrak{B} satisfy the same GF(=) sentences.¹*

For uGC₂(=) and its fragments, we work with interpretations \mathfrak{A} such that $R^{\mathfrak{A}} = \emptyset$ for all R of arity ≥ 3 (and say that \mathfrak{A} interprets relation symbols of arity at most two). Thus, guarded sets contain at most two elements. To preserve counting guarded quantifiers we use the following modified version of guarded bisimulations. A set I of partial isomorphisms $p : \vec{a} \mapsto \vec{b}$ between guarded tuples $\vec{a} = (a_1, a_2)$ in \mathfrak{A} and $\vec{b} = (b_1, b_2)$ in \mathfrak{B} , respectively, is called a *counting connected guarded bisimulation* if the following hold for all $p : (a_1, a_2) \mapsto (b_1, b_2) \in I$:

- (i) for every finite set $X \subseteq \text{dom}(\mathfrak{A})$ such that all (a_1, a'_2) with $a'_2 \in X$ are guarded tuples in \mathfrak{A} there exists an injective mapping f from X to $\text{dom}(\mathfrak{B})$ such that $p' : (a_1, a'_2) \mapsto (b_1, f(a'_2)) \in I$ for all $a'_2 \in X$.
- (ii) for every finite set $Y \subseteq \text{dom}(\mathfrak{B})$ such that all (b_1, b'_2) with $b'_2 \in Y$ are guarded tuples in \mathfrak{B} there exists an injective mapping f from Y to $\text{dom}(\mathfrak{A})$ such that $p' : (a_1, f^{-1}(b'_2)) \mapsto (b_1, b'_2) \in I$ for all $b'_2 \in Y$.

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} that contains $p : \vec{a} \mapsto \vec{b}$. The counting is implemented by the injective mappings f in Conditions (i) and (ii). Note that we quantify only over finite sets X and Y , which corresponds to the ability of counting quantifiers to distinguish between different finite numbers of successors, but not between different infinite cardinalities.

LEMMA 2.7. *Let \mathfrak{A} and \mathfrak{B} interpret relation symbols of arity at most two.*

- (1) *If (\mathfrak{A}, \vec{a}) and (\mathfrak{B}, \vec{b}) are counting connected guarded bisimilar and $\varphi(\vec{x})$ is a formula in openGC₂, then $\mathfrak{A} \models \varphi(\vec{a})$ iff $\mathfrak{B} \models \varphi(\vec{b})$.*
- (2) *If for every guarded \vec{a} in $\text{dom}(\mathfrak{A})$ there exists a guarded \vec{b} in $\text{dom}(\mathfrak{B})$ such that (\mathfrak{A}, \vec{a}) and (\mathfrak{B}, \vec{b}) are counting connected guarded bisimilar and vice versa, then \mathfrak{A} and \mathfrak{B} satisfy the same GC₂(=) sentences.*

The proof of the above lemmas is routine and thus omitted, see [56] for closely related results.

2.5 Guarded Tree Decompositions

We introduce guarded tree decompositions as also used for example in [42] and rooted acyclic queries. A *guarded tree decomposition* of an interpretation \mathfrak{A} is a triple (T, E, bag) with

¹Although we are going to use this result only for uGF(=) sentences, it actually holds for GF(=) sentences as stated.

(T, E) an undirected tree and bag a function that assigns to every $t \in T$ a guarded set $\text{bag}(t)$ in \mathfrak{A} such that

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

When convenient, we assume that (T, E) has a designated root r which allows us to view (T, E) as a directed tree. The difference between a classical tree decomposition [36] and a guarded one is that in the latter, the elements in each bag must be a guarded set. While there is a classical tree decomposition of every interpretation, albeit of potentially high width (that is, maximum bag size), this is not the case for guarded tree decompositions. 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, $\text{bag}(t) \cap \text{bag}(t') \neq \emptyset$ for all $(t, t') \in E$. Note that an interpretation \mathfrak{A} can only have a connected guarded tree decomposition if \mathfrak{A} viewed as a hypergraph is connected.

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 $\text{bag}(r)$ is the set of answer variables of q . Note that, by definition, rAQs are non-Boolean queries.

Example 2.8. 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 define the *unraveling* of an interpretation \mathfrak{A} at a *maximally guarded set* G in \mathfrak{A} into a cg-tree decomposable interpretation \mathfrak{B} . The exact definition of unraveling depends on whether we are working with $\text{uGF}(=)$ or $\text{uGC}_2(=)$. In the former case we want to achieve that there is a connected guarded bisimulation from \mathfrak{B} to \mathfrak{A} . In the latter case we need a counting connected guarded bisimulation and can restrict our attention to relation symbols of arity at most two. Intuitively, the unravelings used here relate to (counting) connected guarded bisimulations in the same way in which the classical unravelings of Kripke structures in modal logic relate to standard bisimulations [42].

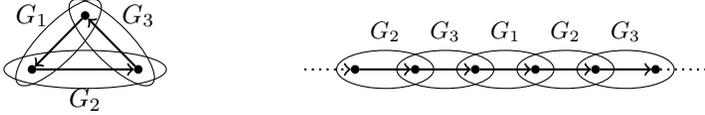
We first consider the case of $\text{uGF}(=)$. Let $T(\mathfrak{A}, G)$ be the set of nodes $t = G_0 G_1 \cdots G_n$, where G_i , $0 \leq i \leq n$, are maximally guarded sets in \mathfrak{A} , $G_0 = G$, and

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

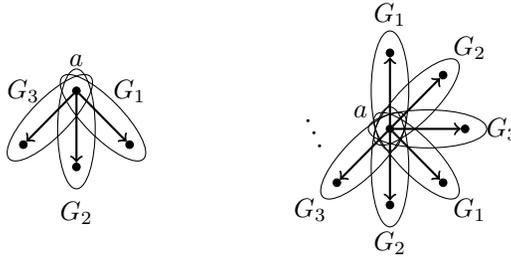
We associate with each $t \in T(\mathfrak{A}, G)$ an interpretation $\text{Bag}(t)$ with domain $\text{bag}(t)$. Then we define \mathfrak{A}_G^u , the *uGF-unraveling of \mathfrak{A} at G* , as $\bigcup_{t \in T(\mathfrak{A}, G)} \text{Bag}(t)$ and note that $(T(\mathfrak{A}, G), E, \text{bag})$ is a cg-tree decomposition of \mathfrak{A}_G^u , where $(t, t') \in E$ if $t' = tF$ for some F .

Take an infinite supply of *copies* of any $a \in \text{dom}(\mathfrak{A})$. We set $a'^{\uparrow} = a$ if a' is a copy of a . We define $\text{Bag}(t)$ and its domain $\text{bag}(t)$ by induction on the length of the sequence t . For $t = G$, $\text{Bag}(t)$ is an interpretation whose domain $\text{bag}(t)$ contains a copy a' of each $a \in G$ such that the mapping $a' \mapsto a'^{\uparrow}$ is an isomorphism from $\text{Bag}(t)$ onto the subinterpretation $\mathfrak{A}|_G$ of \mathfrak{A} induced by G . To define $\text{Bag}(t)$ when $t = G_0 \cdots G_n$ and $n > 0$, take for any $a \in G_n \setminus G_{n-1}$ a fresh copy a' of a and define $\text{Bag}(t)$ with domain $\text{bag}(t) = \{a' \mid a \in G_n \setminus G_{n-1}\} \cup \{a' \in \text{bag}(G_0 \cdots G_{n-1}) \mid a'^{\uparrow} \in G_n \cap G_{n-1}\}$ such that the mapping $a' \mapsto a'^{\uparrow}$ is an isomorphism from $\text{Bag}(t)$ onto $\mathfrak{A}|_{G_n}$. The following example illustrates the construction of \mathfrak{A}_G^u .

Example 2.9. (1) Consider the interpretation \mathfrak{A} depicted below with the maximally guarded sets G_1, G_2, G_3 . Then the uGF-unraveling $\mathfrak{A}_{G_1}^u$ of \mathfrak{A} at G_1 is given by the chain depicted on the right-hand side.



(2) Next consider the interpretation \mathfrak{A} depicted below which has the shape of a tree of depth one with root a and has three maximally guarded sets G_1, G_2, G_3 . Then the uGF-unraveling $\mathfrak{A}_{G_1}^u$ of \mathfrak{A} at G_1 consists of a tree of depth one of infinite outdegree.



For any tuple $\vec{b} = (b_1, \dots, b_n)$ in \mathfrak{A}_G^u we set $\vec{b}^\uparrow = (b_1^\uparrow, \dots, b_n^\uparrow)$. A guarded tuple \vec{b} in \mathfrak{A}_G^u is called a *copy* of a tuple \vec{a} in \mathfrak{A} if $\vec{b}^\uparrow = \vec{a}$.

LEMMA 2.10. *Let G be a maximally guarded set in \mathfrak{A} , $[\vec{a}] = G$, and let \vec{a}' be a copy of \vec{a} in $\text{bag}(G)$. Then the set I of all partial isomorphisms $p : \vec{b} \mapsto \vec{b}^\uparrow$ with $[\vec{b}]$ guarded in the uGF-unraveling \mathfrak{A}_G^u is a connected guarded bisimulation between $(\mathfrak{A}_G^u, \vec{a}')$ and (\mathfrak{A}, \vec{a}) . The mapping $\bigcup_{p \in I} p$ is a homomorphism from \mathfrak{A}_G^u onto \mathfrak{A} .*

The proof of Lemma 2.10 is straightforward and omitted. Note that Condition (b) in the construction of $T(\mathfrak{A}, G)$ corresponds to the condition that we have a *connected* guarded bisimulation. None of the Conditions (a)–(c) are required for the proof to go through. In fact, they are not part of the standard definition of guarded unravelings [41, 42]. They eliminate, however, redundancies in the standard guarded unraveling and, more importantly, ensure the existence of automorphisms of \mathfrak{A}_G^u which will be crucial in the proof of Theorem 5.2 where we work with guarded unravelings of instances to prove dichotomy results.

We now turn to unravelings for uGC_2 , which come with stronger guarantees: the unraveled interpretation is counting connected guarded bisimilar to the original interpretation. Example 2.9 (2) shows that this needs not be the case for uGF-unravelings as defined above as they may introduce too many copies of guarded sets intersecting with a given guarded set. To address this problem, assume that \mathfrak{A} only interprets relation symbols of arity at most two and define the *uGC₂-unraveling* \mathfrak{A}_G^u of \mathfrak{A} at a maximally guarded set G in the same way as the uGF-unraveling except that the Condition (c) is replaced by the stronger condition (c') $G_i \cap G_{i-1} \neq G_i \cap G_{i+1}$.

It is straightforward to prove the following analogue of Lemma 2.10.

LEMMA 2.11. *Let \mathfrak{A} interpret relation symbols of arity at most two, let G be a maximally guarded set in \mathfrak{A} , let $[\vec{a}] = G$, and let \vec{a}' be a copy of \vec{a} in $\text{bag}(G)$. Then the set I of all partial isomorphisms $p : \vec{b} \mapsto \vec{b}^\uparrow$ with $[\vec{b}]$ guarded in the uGC₂-unraveling \mathfrak{A}_G^u is a counting connected*

guarded bisimulation between $(\mathfrak{A}_G^u, \vec{a}') and (\mathfrak{A}, \vec{a}) . The mapping $\bigcup_{p \in I} p$ is a homomorphism from \mathfrak{A}_G^u onto \mathfrak{A} .$

We give a basic application of unravelings which is frequently used throughout the paper. Let \mathfrak{D} be an instance, \mathfrak{B}_G an interpretation, and G a guarded set in both \mathfrak{D} and \mathfrak{B}_G such that $\text{dom}(\mathfrak{B}_G) \cap \text{dom}(\mathfrak{D}) = G$. Then the interpretation $\mathfrak{B} = \mathfrak{D} \cup \mathfrak{B}_G$ is obtained from \mathfrak{D} by hooking \mathfrak{B}_G to \mathfrak{D} at G . If $\mathfrak{B}_G, G \in \mathcal{G}$, is a family of cg-tree decomposable interpretations satisfying the conditions above 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 $\mathfrak{B} = \mathfrak{D} \cup \bigcup_{G \in \mathcal{G}} \mathfrak{B}_G$ is called a *forest model of \mathfrak{D} defined using \mathcal{G}* . If \mathcal{G} is the set of all maximally guarded sets in \mathfrak{D} , then we call \mathfrak{B} simply a *forest model of \mathfrak{D}* .

LEMMA 2.12. *Let \mathcal{O} be a uGF(=) or uGC₂(=) 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 $\text{dom}(\mathfrak{D})$.*

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. Take for any maximally guarded set G in \mathfrak{D} the uGF-unraveling $\mathfrak{B}_G := \mathfrak{A}_G^u$ of \mathfrak{A} at G and hook it to \mathfrak{D} at G by identifying the nodes in G with their copies in $\text{bag}(G)$. It can be shown using Lemma 2.6 and Lemma 2.10 that the union \mathfrak{B} of all \mathfrak{B}_G is as required.

Assume now that \mathcal{O} is a uGC₂(=) ontology, that \mathfrak{D} is an instance, and that \mathfrak{A} is a model of \mathcal{O} and \mathfrak{D} . We may assume that \mathfrak{D} and \mathfrak{A} only interpret relation symbols of arity at most two.

To preserve counting guarded quantifiers the construction is slightly different. Let $c \in \text{dom}(\mathfrak{D})$ and consider the uGC₂-unraveling \mathfrak{A}_G^u of \mathfrak{A} at G , for every maximally guarded G in \mathfrak{A} with $G \cap \text{dom}(\mathfrak{D}) = \{c\}$. To ensure that we do not add copies of successors of c in \mathfrak{D} to the unraveling we take the modification \mathfrak{B}_G of \mathfrak{A}_G^u in which the paths $G_0 G_1 \cdots G_n$ with $G_0 = G$ satisfy $G_1 \cap \text{dom}(\mathfrak{D}) = \emptyset$. Now hook all these \mathfrak{B}_G to \mathfrak{D} at c by identifying c with its copy in \mathfrak{B}_G (in particular $\text{dom}(\mathfrak{B}_G) \cap \text{dom}(\mathfrak{D}) = \{c\}$). Define \mathfrak{B} as the union of $\mathfrak{A}_{|\text{dom}(\mathfrak{D})}$ and all \mathfrak{B}_G constructed for any $c \in \text{dom}(\mathfrak{D})$. It can be shown using Lemma 2.7 and Lemma 2.11 that \mathfrak{B} is as required. \square

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 the existence of universal models defined in terms of homomorphisms. We then prove that in uGF(=) and in uGC₂(=), non-materializability implies CONP-hardness of query evaluation. We also observe that, in contrast, an analogous statement does not hold for GF. We then use these results to establish that for ontologies formulated in uGF(=) or in uGC₂(=), PTIME query evaluation, Datalog[≠]-rewritability of query evaluation, and CONP-hardness of query evaluation does not depend on the actual query language, that is, all these properties agree for rAQs, CQs, and UCQs. Again, this is not the case for GF.

Definition 3.1 (Materializability). Let \mathcal{O} be an FO(=)-ontology and \mathcal{Q} a class of queries. 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 $\text{dom}(\mathfrak{D})$, $\mathfrak{B} \models q(\vec{a})$ iff $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$.

- Let \mathcal{M} be a class of instances. Then \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). The intuitive reason is given by Lemma 2.12, namely that we can restrict our attention to forest models, which essentially allows us to decompose CQs into rAQs.

THEOREM 3.2. *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. We show the implications (1) \Rightarrow (2), (2) \Rightarrow (3), and (3) \Rightarrow (1). The implication (3) \Rightarrow (1) follows from the fact that every rAQ is a UCQ. (2) \Rightarrow (3) follows from the observation that any interpretation \mathfrak{B} is a CQ-materialization of \mathcal{O} and an instance \mathfrak{D} if, and only if, it is UCQ-materialization of \mathcal{O} and \mathfrak{D} . We now 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 $\text{dom}(\mathfrak{D})$ (Lemma 2.12). Recall that \mathfrak{B} is obtained from \mathfrak{D} by hooking cg-tree decomposable \mathfrak{B}_G to \mathfrak{D} at G , for any maximally 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 suffices 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 $\text{dom}(\mathfrak{D}) \cap \text{dom}(\mathfrak{B}')$. Assume \mathfrak{A}' and \mathfrak{B}' are given. We may assume that $\text{dom}(\mathfrak{D}) \subseteq \text{dom}(\mathfrak{B}')$ and that $\mathfrak{B}' \cap \mathfrak{B}_G$ is connected for every maximally guarded set G in \mathfrak{D} . 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 a suitable \vec{b} with $[\vec{b}] = G$, we obtain $\mathfrak{A}' \models q_G(\vec{b})$ since \mathfrak{B} is an rAQ-materialization of \mathfrak{D} and \mathcal{O} . Let h_G be the homomorphism witnessing $\mathfrak{A}' \models q_G(\vec{b})$. 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 maximally guarded set in \mathfrak{D} , is the desired homomorphism from \mathfrak{B}' to \mathfrak{A}' preserving $\text{dom}(\mathfrak{D}) \cap \text{dom}(\mathfrak{B}')$. \square

Because of Theorem 3.2, from now on we 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 universal models defined in terms of homomorphisms as used e.g. in data exchange [29, 32]. A model of an ontology \mathcal{O} and an instance \mathfrak{D} is *hom-universal* if there is a homomorphism preserving $\text{dom}(\mathfrak{D})$ 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, we show that in $uGC_2(=)$, 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).

LEMMA 3.3. *A $uGC_2(=)$ ontology is materializable iff it admits hom-universal models.*

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 2.12 there exists a forest model

\mathfrak{B} of \mathfrak{D} and \mathcal{O} that is a CQ-materialization of \mathcal{O} and \mathfrak{D} . We may assume that \mathfrak{B} interprets relation symbols of arity at most two. By selecting witnesses for existential formulas and dropping subtrees that are not needed as witnesses, 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 (actually bounded by the number of existentially quantified subformulas in \mathcal{O}). 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 $\text{dom}(\mathfrak{D})$. 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, since \mathfrak{B}' is a CQ-materialization of \mathcal{O} and \mathfrak{D} , for any finite subset F of $\text{dom}(\mathfrak{B}')$ there is a homomorphism h_F from $\mathfrak{B}'|_F$ to \mathfrak{A} preserving $\text{dom}(\mathfrak{D}) \cap F$. Let F_m be the set of all $d \in \text{dom}(\mathfrak{B}')$ such that there is a sequence of at most $m + 1$ guarded sets G_0, \dots, G_m with $G_i \cap G_{i+1} \neq \emptyset$ for $i < m$, $G_0 \cap \text{dom}(\mathfrak{D}) \neq \emptyset$, and $d \in G_m$. Then each F_m is finite and $\bigcup_{m \geq 0} F_m = \text{dom}(\mathfrak{B}')$. Using a standard pigeonhole argument one can construct an infinite sequence of natural numbers $n_0 < n_1 < \dots$ such that $h_{F_{n_0}} \subseteq h_{F_{n_1}} \subseteq \dots$. Then $h = \bigcup_{i \geq 0} h_{F_{n_i}}$ is the required homomorphism from \mathfrak{B}' to \mathfrak{A} . \square

Lemma 3.3 does not hold for uGF(2) ontologies. In fact, we show in the appendix that there exists a materializable ontology \mathcal{O} in uGF(2) with three variables not admitting hom-universal models such that CQ-evaluation w.r.t. \mathcal{O} is in PTIME. Thus, admitting hom-universal models is not a necessary condition for query evaluation to be in PTIME, in contrast to materializability.

LEMMA 3.4. *There exists a materializable ontology \mathcal{O} in uGF(2) not admitting hom-universal models. Moreover, CQ-evaluation w.r.t. \mathcal{O} is in PTIME.*

We next aim to show that materializability of ontologies is a necessary condition for query evaluation to be in PTIME, unless PTIME = NP. For proving this, it is more convenient to work with a certain disjunction property instead of directly using materializability. We now introduce this property and show the equivalence of the two notions. Let \mathcal{Q} be a class of CQs, \mathcal{O} an ontology, and \mathfrak{D} an instance. For $q_1(\vec{x}_1), \dots, q_n(\vec{x}_n) \in \mathcal{Q}$ and tuples $\vec{d}_1, \dots, \vec{d}_n$ in \mathfrak{D} we write $\mathcal{O}, \mathfrak{D} \models q_1(\vec{d}_1) \vee \dots \vee q_n(\vec{d}_n)$ if for every model \mathfrak{A} of \mathcal{O} and \mathfrak{D} there exists $1 \leq i \leq n$ such that $\mathfrak{A} \models q_i(\vec{d}_i)$. An ontology \mathcal{O} has the *\mathcal{Q} -disjunction property* if for all instances \mathfrak{D} , queries $q_1(\vec{x}_1), \dots, q_n(\vec{x}_n) \in \mathcal{Q}$ and $\vec{d}_1, \dots, \vec{d}_n$ in \mathfrak{D} : if $\mathcal{O}, \mathfrak{D} \models q_1(\vec{d}_1) \vee \dots \vee q_n(\vec{d}_n)$, then there exists $1 \leq i \leq n$ such that $\mathcal{O}, \mathfrak{D} \models q_i(\vec{d}_i)$.

THEOREM 3.5. *Let \mathcal{Q} be a class of 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 ‘ \Leftarrow ’ direction, let \mathfrak{D} be an instance consistent w.r.t. \mathcal{O} such that there is no \mathcal{Q} -materialization of \mathcal{O} and \mathfrak{D} . Consider the set of FO(=) sentences Γ containing all $\neg \exists \vec{y} \phi(\vec{d}, \vec{y})$ such that $\mathcal{O}, \mathfrak{D} \not\models q(\vec{d})$ and $q(\vec{x}) \leftarrow \phi(\vec{x}, \vec{y}) \in \mathcal{Q}$. Then $\mathcal{O} \cup \mathfrak{D} \cup \Gamma$ is not satisfiable as any satisfying interpretation would be a \mathcal{Q} -materialization of \mathcal{O} and \mathfrak{D} . By compactness of FO(=), there is a finite subset $\Gamma' \subseteq \Gamma$ such that $\mathcal{O} \cup \mathfrak{D} \cup \Gamma'$ is not satisfiable. Then the set of all $q(\vec{d})$ corresponding to Γ' refutes the \mathcal{Q} -disjunction property for \mathcal{O} . \square

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 [70], a variation of a related proof from [59]. For some results established later on, it is important that we establish the following for unary rAQs.

THEOREM 3.6. *Let \mathcal{O} be an $FO(=)$ -ontology that is invariant under disjoint unions. If \mathcal{O} is not materializable, then the evaluation of unary rAQs w.r.t. \mathcal{O} is CONP-hard .*

SKETCH. It was proved in [59] 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 unary 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 [70]. The proof of Theorem 3.6 can be obtained from the proof in [59] by minor modifications, which we sketch in the following.

The proof crucially exploits that if \mathcal{O} is not rAQ-materializable, then by Theorem 3.5 it does not have the rAQ-disjunction property. In fact, we take an instance \mathfrak{D} , (not necessarily unary) rAQs $q_1(\vec{x}_1), \dots, q_n(\vec{x}_n) \in \mathcal{Q}$, and elements $\vec{d}_1, \dots, \vec{d}_n$ 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 [59] 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, one can replace these relations with k binary relations. As in the original construction in [59], one finally ends up with a query that is unary. \square

We remark that, in the proof of Theorem 3.6, we use instances and rAQs that involve additional fresh relation symbols, that is, relation symbols that do not occur in \mathcal{O} . It suffices to use binary fresh symbols and thus we stay within the assumed signature restrictions when working with uGF_2 and uGC_2 . The ontology $\mathcal{O}_{\text{Mat/PTime}}$ from Example 2.3 shows that Theorem 3.6 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 are both easy to see).

The next two theorems are the second main result of this section.

THEOREM 3.7. *For all $\text{uGF}(=)$ 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[≠]-rewritable’ and with ‘CONP-hard’ (and with ‘Datalog-rewritable’ if \mathcal{O} is a uGF ontology).

The proof is given in the appendix. For PTIME membership and for rewritability, it suffices to prove the implication (1) \Rightarrow (3). The central idea is to use a decomposition of CQs into a quantifier free CQ and a collection of rAQs that goes under various names such as splittings [55], forest decompositions [13], and squid decompositions [19], see also [9]. To achieve this, we exploit that \mathcal{O} is materializable, by Theorem 3.6. For CONP-hardness , it suffices to prove (3) \Rightarrow (1). We again use the decomposition mentioned above and Theorem 3.6.

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

The following theorem states that the equivalences of Theorem 3.7 hold for $\text{uGC}_2(=)$ ontologies as well. For the proof of Theorem 6.6 below, it will be convenient to state the equivalence also for unary rAQ -evaluation. The proof is a straightforward adaptation of the proof of Theorem 3.7.

THEOREM 3.8. *For all $\text{uGC}_2(=)$ ontologies \mathcal{O} , the statements (1) to (3) of Theorem 3.7 are equivalent and also equivalent to*

(4) *unary rAQ -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’.

Because of Theorems 3.7 and 3.8, when dealing with a $\text{uGF}(=)$ or $\text{uGC}_2(=)$ ontology we will simply speak about PTIME query evaluation, Datalog $^\neq$ -rewritability of query evaluation, and CONP-hardness of query evaluation without specifying the actual query language, as all these properties agree for rAQ s, CQ s, and UCQ s. The ontology $\mathcal{O}_{\text{UCQ}/\text{CQ}}$ from Example 2.3 shows that this 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 3.9. *CQ-evaluation w.r.t. $\mathcal{O}_{\text{UCQ}/\text{CQ}}$ is in PTIME. In contrast, UCQ-evaluation w.r.t. $\mathcal{O}_{\text{UCQ}/\text{CQ}}$ is CONP-hard.*

SKETCH. The lower bound essentially follows the construction in the proof of Theorem 3.6. For the upper bound, fix a CQ $q(\vec{x})$, and consider an input instance \mathfrak{D} and a tuple \vec{a} in \mathfrak{D} . If $\mathfrak{D} \models q(\vec{a})$, then clearly $\mathcal{O}_{\text{UCQ}/\text{CQ}}, \mathfrak{D} \models q(\vec{a})$. Otherwise, if $\mathfrak{D} \not\models q(\vec{a})$, then one can show that $\mathcal{O}_{\text{UCQ}/\text{CQ}}, \mathfrak{D} \not\models q(\vec{a})$. There are three cases to consider. If $E^\mathfrak{D}$ is nonempty, then \mathfrak{D} is a model of $\mathcal{O}_{\text{UCQ}/\text{CQ}}$ that falsifies $q(\vec{a})$. If $E^\mathfrak{D}$ is empty, then for each of the two cases – q contains an atomic formula of the form $E(y)$ or not – we can build a model of \mathfrak{D} and $\mathcal{O}_{\text{UCQ}/\text{CQ}}$ that falsifies $q(\vec{a})$. \square

4 UNRAVELING TOLERANCE

While materializability of an ontology is a necessary condition for PTIME query evaluation in $\text{uGF}(=)$ and $\text{uGC}_2(=)$, we now identify a sufficient condition for Datalog $^\neq$ -rewritability (and thus also for PTIME query evaluation) called unraveling tolerance. Unraveling tolerance is defined using the disjoint union of all unravelings of an instance at its maximally guarded sets, as defined in Section 2.5. We shall later establish strong dichotomy results by showing that for the ontology languages in question, materializability implies unraveling tolerance. We also identify a large class of unraveling tolerant ontologies by proving that ontologies whose models are preserved under direct products are unraveling tolerant. It follows, in particular, that every $\text{uGF}(=)$ and $\text{uGC}_2(=)$ ontology that is expressible in Horn $\text{FO}(=)$ is unraveling tolerant.

Similarly to the unraveling of an interpretation at a maximally guarded set, the global unraveling of an instance depends on whether we work with $\text{uGF}(=)$ or its two variable fragment with counting.

Definition 4.1 (Global Unraveling of Data Instance). Let \mathfrak{D} be an instance. The *global uGF-unraveling* (resp. *global uGC₂-unraveling*) \mathfrak{D}^u of \mathfrak{D} is the disjoint union of all uGF -unravelings (uGC_2 -unravelings) \mathfrak{D}_G^u of \mathfrak{D} at G , where G is a maximally guarded set in \mathfrak{D} (for the uGC_2 -unraveling, we assume that \mathfrak{D} only interprets relation symbols of arity at most two).

We use the notation introduced for unravelings in Section 2.5 when talking about global unravelings. Thus, for each maximally guarded set G in \mathfrak{D} we have a cg-tree decomposition $(T(\mathfrak{D}, G), E, \text{bag})$ of \mathfrak{D}_G^u where $T(\mathfrak{D}, G)$ is the set of sequences $G_0 \cdots G_n$ of maximally guarded sets in \mathfrak{D} with $G_0 = G$ and satisfying the following conditions introduced in Section 2.5: (a) $G_1 \neq G_{i+1}$, (b) $G_i \cap G_{i+1} \neq \emptyset$, and (c) $G_{i-1} \neq G_{i+1}$ or (a), (b), and (c') $G_i \cap G_{i-1} \neq G_i \cap G_{i+1}$, respectively. $T(\mathfrak{D})$ denotes the union of all $T(\mathfrak{D}, G)$, where G is a maximally guarded set in \mathfrak{D} . We set $\text{tail}(G_0 \cdots G_n) = G_n$ and call G_n the *tail* of $G_0 \cdots G_n \in T(\mathfrak{D})$. For $t \in T(\mathfrak{D})$, every $a \in \text{bag}(t)$ is a copy of a unique $a^\uparrow \in \text{tail}(t)$.

Definition 4.2 (Unraveling Tolerance). A uGF(=) (resp. uGC₂(=)) ontology \mathcal{O} is *unraveling tolerant* if for every instance \mathfrak{D} , every rAQ $q(\vec{x})$, and every tuple \vec{a} in \mathfrak{D} such that $G = [\vec{a}]$ 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 $\text{bag}(G)$

where \mathfrak{D}^u is the global uGF-unraveling (resp. the global uGC₂-unraveling) of \mathfrak{D} .

In Definition 4.2, one can equivalently replace the equivalence (1) \Leftrightarrow (2) by the implication (1) \Rightarrow (2). In fact, the following observation is shown in the appendix.

LEMMA 4.3. *The implication (2) \Rightarrow (1) in Definition 4.2 holds for every uGF(=) and uGC₂(=) ontology and every rAQ.*

Note that it is pointless to define unraveling tolerance using UCQs or CQs in place of rAQs since the former query languages can trivially separate database instances from their (global) unraveling. Conversely, it might seem that rAQs are not sufficiently powerful to achieve the separation. We use the instance introduced in Example 2.9 to illustrate how rAQs are used to refute unraveling tolerance.

Example 4.4. Consider the uGF ontology \mathcal{O} that contains the sentences

$$\begin{aligned} \forall x (A(x) \rightarrow (\exists y (R(x, y) \wedge A(y)) \rightarrow E(x))) \\ \forall x (\neg A(x) \rightarrow (\exists y (R(x, y) \wedge \neg A(y)) \rightarrow E(x))) \\ \forall x (E(x) \rightarrow ((R(x, y) \vee R(y, x)) \rightarrow E(y))). \end{aligned}$$

Assume that \mathfrak{D} is an instance with $A(b) \notin \mathfrak{D}$ for any b . Then $\mathcal{O}, \mathfrak{D} \models E(a)$ iff there is a $R \cup R^{-1}$ -path from a to some c in an odd R -cycle in \mathfrak{D} . Thus, for the instance \mathfrak{D} from Example 2.9 (1) we have $\mathcal{O}, \mathfrak{D} \models E(a)$ for every $a \in \text{dom}(\mathfrak{D})$, but $\mathcal{O}, \mathfrak{D}^u \not\models E(a)$ for any $a \in \text{dom}(\mathfrak{D}^u)$.

We now show that unraveling tolerance implies that query evaluation is Datalog $^\neq$ -rewritable.

THEOREM 4.5. *If \mathcal{O} is an unraveling tolerant uGF(=) or uGC₂(=) ontology, then rAQ-evaluation w.r.t. \mathcal{O} is Datalog $^\neq$ -rewritable (resp., Datalog-rewritable if \mathcal{O} is formulated in uGF).*

A detailed proof is given in the appendix. For the case that \mathcal{O} is a uGF(=) ontology, the basic idea is as follows. Suppose that \mathcal{O} is unraveling tolerant, and that $q(\vec{x})$ is a rAQ. We have to construct a Datalog $^\neq$ program Π that on any instance \mathfrak{D} computes the certain answers \vec{a} of q on \mathfrak{D} given \mathcal{O} . We can w.l.o.g. restrict our attention to answers \vec{a} such that the set of elements of \vec{a} is maximally guarded in \mathfrak{D} . By unraveling 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 $\text{bag}(G)$ and \mathfrak{D}^u is the uGF-unraveling of

\mathfrak{D} . The Datalog[≠] program Π assigns to each maximally guarded tuple $\vec{a} = (a_1, \dots, a_k)$ in \mathfrak{D} a set of *types* where, roughly, a type is maximally consistent set of uGF formulas with free variables in x_1, \dots, x_k where variable x_i represents the element a_i and that can be realized in some model of \mathcal{O} . To achieve finiteness of types (and of Π), we restrict our attention to subformulas of \mathcal{O} and to a certain finite set of formulas induced by q . The Datalog[≠] program Π ensures the following:

- (1) for any two maximally guarded tuples $\vec{a} = (a_1, \dots, a_k)$, $\vec{b} = (b_1, \dots, b_l)$ 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 $\vec{a} = (a_1, \dots, a_k)$ is an answer to Π if all types assigned to \vec{a} contain $q(x_1, \dots, x_k)$, 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 $\mathcal{O}, \mathfrak{D}^u \models q(\vec{a})$. A similar idea works for uGC₂(=) ontologies, but the program is more complex.

We next show that ontologies whose models are preserved under direct products are unraveling tolerant. This covers all ontologies in uGF(=) and uGC₂(=) that can be expressed in Horn FO(=) [26] and all so-called Horn description logics, syntactically defined fragments of expressive DLs that enjoy PTIME query evaluation and that fall within Horn FO(=), see for example [31, 44, 50].

The *direct product* $\mathfrak{A} = \prod_{i \in I} \mathfrak{A}_i$ of a family \mathfrak{A}_i , $i \in I$, of interpretations is defined by setting

$$\begin{aligned} \text{dom}(\mathfrak{A}) &= \{f : I \rightarrow \bigcup \text{dom}(\mathfrak{A}_i) \mid \forall i \in I : f(i) \in \text{dom}(\mathfrak{A}_i)\} \\ \mathfrak{A} &= \{R(f_1, \dots, f_k) \mid \forall i \in I : R(f_1(i), \dots, f_k(i)) \in \mathfrak{A}_i\} \end{aligned}$$

We regard the functions $f \in \text{dom}(\mathfrak{A})$ as constants and identify the constant function f_a mapping every $i \in I$ to $a \in \bigcap_{i \in I} \text{dom}(\mathfrak{A}_i)$ with the constant a . An ontology \mathcal{O} is *preserved under direct products* if $\prod_{i \in I} \mathfrak{A}_i$ is a model of \mathcal{O} whenever \mathfrak{A}_i , $i \in I$, is a family of models of \mathcal{O} . We show that if \mathcal{O} is either a uGF(=) or uGC₂(=) ontology preserved under direct products, then \mathcal{O} is unraveling tolerant. First we introduce a natural equivalence relation and automorphisms on the global unraveling \mathfrak{D}^u of an instance \mathfrak{D} .

Let \mathfrak{D} be an instance. Define equivalence relations \sim on $T(\mathfrak{D})$ and \sim^u on \mathfrak{D}^u by setting $t \sim t'$ if $\text{tail}(t) = \text{tail}(t')$ and $a \sim^u b$ if $a^\uparrow = b^\uparrow$, respectively. For any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ the mapping $h_{t,t'}$ that sends every $a \in \text{bag}(t)$ to the unique $b \in \text{bag}(t')$ with $a \sim^u b$ is an isomorphism from $\mathfrak{D}_{|\text{bag}(t)}$ to $\mathfrak{D}_{|\text{bag}(t')}$, called the *canonical isomorphism*. Using the Conditions (a)–(c) from the construction of $(T(\mathfrak{D}), E)$ one can readily show that 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 by t and t' on the connected component $T(\mathfrak{D}, G)$ of t in $T(\mathfrak{D})$ and induces the mapping $\hat{h}_{t,t'}$ from \mathfrak{D}^u into \mathfrak{D}^u defined by setting $\hat{h}_{t,t'} = \bigcup_{s \in T(\mathfrak{D})} h_{s, i_{t,t'}(s)}$.

LEMMA 4.6. *Let $t, t' \in T(\mathfrak{D})$ such that $t \sim t'$. Then $\hat{h}_{t,t'}$ is an automorphism of \mathfrak{D}^u .*

Call $\hat{h}_{t,t'}$ the *canonical automorphism of \mathfrak{D}^u* induced by t, t' . Lemma 4.6 shall be a fundamental tool for the constructions in Section 5. We apply it here to prove the announced result that preservation under direct products implies unraveling tolerance.

THEOREM 4.7. *Let \mathcal{O} be a uGF(=) or uGC₂(=) ontology preserved under direct products. Then \mathcal{O} is unraveling tolerant.*

PROOF. Let \mathfrak{D} be an instance, G_0 a maximally guarded set in \mathfrak{D} , and assume that $[\vec{a}] = G_0$, \vec{b} is a copy of \vec{a} in $\text{bag}(G_0)$, and that $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$ for an rAQ $q(\vec{x})$. We have to show that $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$. Let $\mathfrak{A}_i, i \in I$, be the family of at most countable forest models of \mathcal{O} and \mathfrak{D} (up to isomorphisms). Then $\mathfrak{A}^* = \prod_{i \in I} \mathfrak{A}_i$ is a model of \mathcal{O} and \mathfrak{D} (recall that we identify for every $a \in \text{dom}(\mathfrak{D}^u)$ the constant function f_a mapping all $i \in I$ to $f(i) = a$ with a). Moreover, $\mathfrak{A}^* \not\models q(\vec{b})$ since there exists $i \in I$ such that $\mathfrak{A}_i \not\models q(\vec{b})$ (and the projection is a homomorphism from \mathfrak{A}^* to \mathfrak{A}_i). Next observe that for any $t, t' \in T(\mathfrak{D})$ such that $t \sim t'$, the automorphism $\hat{h}_{t,t'}$ of \mathfrak{D}^u from Lemma 4.6 can be lifted to an automorphism $h_{t,t'}^I$ of \mathfrak{A}^* . In particular, for any t, t' with $t \sim t'$ there is an isomorphism from the interpretation hooked to \mathfrak{D}^u at $\text{bag}(t)$ in \mathfrak{A}^* onto the interpretation hooked to \mathfrak{D}^u at $\text{bag}(t')$ in \mathfrak{A}^* mapping every $a \in \text{bag}(t)$ to the unique a' with $a \sim^u a'$ in $\text{bag}(t')$.

Assume first that \mathcal{O} is a uGF(=) ontology. Define a model \mathfrak{A} of \mathfrak{D} by taking for every maximally guarded G in \mathfrak{D} a maximally guarded G' in \mathfrak{D}^u with $G'^\uparrow = G$ and hooking to \mathfrak{D} at G a copy of the interpretation $\mathfrak{A}_{G'}^*$, hooked to \mathfrak{D}^u in \mathfrak{A}^* at G' by identifying every $a \in G'$ with $a^\uparrow \in G$. Assume $[\vec{c}] = G'$. Using Lemma 2.10 and the automorphisms $h_{t,t'}^I$ of \mathfrak{A}^* one can readily check that there is a connected guarded bisimulation between $(\mathfrak{A}^*, \vec{c})$ and $(\mathfrak{A}, \vec{c}^\uparrow)$. Thus, by Lemma 2.6, \mathfrak{A} is a model of \mathcal{O} . Moreover, as we can regard every rAQ as a formula in openGF, we also have $\mathfrak{A} \not\models q(\vec{a})$.

Assume now \mathcal{O} is a uGC₂(=) ontology. Define a model \mathfrak{A} of \mathfrak{D} by hooking to \mathfrak{D} at every $c^\uparrow \in \text{dom}(\mathfrak{D})$ a copy of the interpretation \mathfrak{A}_c hooked to \mathfrak{D}^u in \mathfrak{A}^* at c by identifying c with c^\uparrow . In addition, add $\{R(c_1^\uparrow, c_2^\uparrow) \mid R(c_1, c_2) \in \mathfrak{A}_{|\text{dom}(\mathfrak{D}^u)}^*\}$ to \mathfrak{D} . Using Lemma 2.11 and the automorphisms $h_{t,t'}^I$ of \mathfrak{A}^* one can check that for every maximally guarded G' in \mathfrak{D}^u and \vec{c} with $[\vec{c}] = G'$ there is a counting connected guarded bisimulation between $(\mathfrak{A}^*, \vec{c})$ and $(\mathfrak{A}, \vec{c}^\uparrow)$. Thus, by Lemma 2.7, \mathfrak{A} is a model of \mathcal{O} . As we can regard q as a formula in openGF, we also have $\mathfrak{A} \not\models q(\vec{a})$. \square

5 STRONG DICHOTOMIES

We prove dichotomies between Datalog[≠]-rewritability and coNP-hardness of query evaluation in the six ontology languages displayed in the bottommost row of Figure 1.³ This also implies that, unless PTIME = NP, Datalog[≠]-rewritability coincides with PTIME query evaluation. The proof consists of showing that in the ontology languages under consideration, materializability implies unraveling tolerance. It follows that PTIME query evaluation also coincides with unraveling tolerance and with materializability (again unless PTIME = NP).

Let \mathfrak{D} be an instance. In what follows we make intense use of Lemma 4.6. In particular, we use the following straightforward consequence.

LEMMA 5.1. *Let $t, t' \in T(\mathfrak{D})$, $t \sim t'$, and let \mathcal{O} be an FO(=) ontology. Then the following hold.*

- (1) *If $[\vec{a}] \subseteq \text{bag}(t)$, then $\mathcal{O}, \mathfrak{D}^u \models q(\vec{a})$ iff $\mathcal{O}, \mathfrak{D}^u \models q(h_{t,t'}(\vec{a}))$ holds for all rAQs $q(\vec{x})$;*
- (2) *If \mathfrak{A} is a materialization of \mathcal{O} and \mathfrak{D}^u , then $\hat{h}_{t,t'}$ is an automorphism of $\mathfrak{A}_{|\text{dom}(\mathfrak{D}^u)}$.*

In fact, Point (1) of Lemma 5.1 is an immediate consequence of Lemma 4.6 and Point (2) is a consequence of Point (1) by the definition of materializations and since every fact in $\mathfrak{A}_{|\text{dom}(\mathfrak{D}^u)} \setminus \mathfrak{D}^u$ can be viewed as an answer to a rAQ.

³In what follows we do not explicitly consider \mathcal{ALCHIQ} ontologies of depth 1 since they can be equivalently rewritten into ontologies formulated in uGC₂⁻(1, =).

We now establish the main result of this section. In anticipation of the decidability results to be proved in Section 9, we actually state it in a form that is slightly stronger than announced: already when \mathcal{O} is materializable for the class of (possibly infinite) cg-tree decomposable instances with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$, it must be unraveling tolerant. It can be established by an easy compactness argument that materializability implies materializability for the mentioned class of instances, even within full $\text{FO}(=)$, so we also obtain the result announced originally.

THEOREM 5.2. *Let \mathcal{O} be an ontology formulated in one of $\text{uGF}(1)$, $\text{uGF}^-(1, =)$, $\text{uGF}_2^-(2)$, $\text{uGC}_2^-(1, =)$, or an *ALCHIF* ontology of depth 2. If \mathcal{O} is materializable for the class of (possibly infinite) cg-tree decomposable instances \mathfrak{D} with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$, then \mathcal{O} is unraveling tolerant.*

PROOF. We give the proof for $\text{uGC}_2^-(1, =)$. The remaining cases are considered in the appendix; they are more involved but based on the same ideas. We first observe that if \mathcal{O} is materializable for the class of cg-tree decomposable instances \mathfrak{D} with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$, then it is materializable for the class of all cg-tree decomposable instances without any signature restrictions. To show this, assume that the former holds and let \mathfrak{D} be an arbitrary cg-tree decomposable instance consistent w.r.t. \mathcal{O} . Let $\text{red}(\mathfrak{D}^u)$ be the $\text{sig}(\mathcal{O})$ -reduct of \mathfrak{D} . As \mathcal{O} is invariant under disjoint unions and materializable for the class of cg-tree decomposable instances \mathfrak{D} with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$ there exists a materialization $\mathfrak{B}_{\text{red}}$ of $\text{red}(\mathfrak{D}^u)$. Clearly $\{R \mid R(\vec{a}) \in \mathfrak{B}_{\text{red}}\} \subseteq \text{sig}(\mathcal{O})$. 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} . This finishes the proof of the observation.

Now let \mathcal{O} be an ontology in $\text{uGC}_2^-(1, =)$ and let \mathfrak{D} be an instance interpreting relation symbols of arity at most two. Let \mathfrak{D}^u be the global uGC_2 -unraveling of \mathfrak{D} . Let G_0 be a maximal guarded set in \mathfrak{D} , $[\vec{a}] = G_0$, \vec{b} the copy of \vec{a} in $\text{bag}(G_0)$, and q_0 an rAQ. Assume that $\mathcal{O}, \mathfrak{D}^u \not\models q_0(\vec{b})$. We aim to show that $\mathcal{O}, \mathfrak{D} \not\models q_0(\vec{a})$.

By the observation above, there exists a materialization \mathfrak{B}^u of \mathfrak{D}^u . We may assume that \mathfrak{B}^u is a forest model. Take for any $c \in \text{dom}(\mathfrak{D}^u)$ the cg-tree decomposable interpretation \mathfrak{B}_c hooked to \mathfrak{D}^u at c . In particular, $\text{dom}(\mathfrak{D}^u) \cap \text{dom}(\mathfrak{B}_c) = \{c\}$. Fix for every equivalence class $\{d \mid c \sim^u d\}$ in \mathfrak{D}^u a $c^* \sim^u c$. We define a model \mathfrak{B} of \mathfrak{D} by

- hooking to \mathfrak{D} at every $c^\uparrow \in \text{dom}(\mathfrak{D})$ a copy $\mathfrak{B}_{c^*}^\uparrow$ of the interpretation \mathfrak{B}_{c^*} hooked to \mathfrak{D}^u at c^* in \mathfrak{B}^u (we assume $\text{dom}(\mathfrak{D}) \cap \text{dom}(\mathfrak{B}_{c^*}^\uparrow) = \{c^\uparrow\}$) and
- adding the atoms $\{R(c_1^\uparrow, c_2^\uparrow) \mid R(c_1, c_2) \in \mathfrak{B}_{|\text{dom}(\mathfrak{D}^u)}^u\}$.

We show that \mathfrak{B} is a model of \mathcal{O} and \mathfrak{D} such that $\mathfrak{B} \not\models q_0(\vec{a})$, as required. For the proof we *uniformize* \mathfrak{B}^u . Define \mathfrak{B}^* by hooking to \mathfrak{D}^u at c a copy $\mathfrak{B}_c^{c^*}$ of the interpretation \mathfrak{B}_{c^*} hooked to \mathfrak{D}^u at c^* in \mathfrak{B}^u , for every $c \in \text{dom}(\mathfrak{D}^u)$, and adding $\mathfrak{B}_{|\text{dom}(\mathfrak{D}^u)}$. We assume $\text{dom}(\mathfrak{D}^u) \cap \text{dom}(\mathfrak{B}_c^{c^*}) = \{c\}$. We show that \mathfrak{B}^* is also a materialization of \mathcal{O} and \mathfrak{D}^u . By Lemma 5.1, $\mathfrak{B}^* \models q(\vec{a})$ iff $\mathfrak{B}^u \models q(\vec{a})$ holds for all guarded \vec{a} in \mathfrak{D}^u and all rAQs q . It remains to prove that \mathfrak{B}^* is a model of \mathcal{O} . For this to hold, the restriction to sentences in $\text{uGC}_2^-(1, =)$ is crucial. Let $\varphi \in \mathcal{O}$. Then φ is of the form $\forall x(x = x \rightarrow \psi(x))$, where $\psi(x)$ is a formula of depth 1 in openGC_2 . Consider $a \in \text{dom}(\mathfrak{B}^*)$. We have to show that $\mathfrak{B}^* \models \psi(a)$. We distinguish two cases:

Case 1. $a \in \text{dom}(\mathfrak{B}_{c^*}^c) \setminus \{c\}$ for some $c \in \text{dom}(\mathfrak{D}^u)$. Let a' be the element corresponding to a in \mathfrak{B}_{c^*} . As ψ has depth 1 and the interpretations $\mathfrak{B}_{\{c\}}^u$ and $\mathfrak{B}_{\{c^*\}}^u$ are isomorphic by Lemma 5.1, we have the following equivalences:

$$\mathfrak{B}^* \models \psi(a) \Leftrightarrow \mathfrak{B}_{c^*}^c \models \psi(a) \Leftrightarrow \mathfrak{B}_{c^*} \models \psi(a') \Leftrightarrow \mathfrak{B}^u \models \psi(a')$$

and the claim follows from the assumption that \mathfrak{B}^u is a model of \mathcal{O} .

Case 2. $a \in \text{dom}(\mathfrak{D}^u)$. Let $N(c) = \{c\} \cup \{d \mid R(c, d) \in \mathfrak{D}^u \text{ or } R(d, c) \in \mathfrak{D}^u\}$, for any $c \in \text{dom}(\mathfrak{D}^u)$. By Lemma 5.1, the interpretations $\mathfrak{B}_{N(c)}^u$ and $\mathfrak{B}_{N(c^*)}^u$ are isomorphic for every $c \in \text{dom}(\mathfrak{D}^u)$. Thus, as ψ has depth 1:

$$\mathfrak{B}^* \models \psi(a) \Leftrightarrow \mathfrak{B}_{N(a)}^* \cup \mathfrak{B}_{a^*}^a \models \psi(a) \Leftrightarrow \mathfrak{B}_{N(a^*)}^u \cup \mathfrak{B}_{a^*} \models \psi(a^*) \Leftrightarrow \mathfrak{B}^u \models \psi(a^*)$$

and the claim follows from the assumption that \mathfrak{B}^u is a model of \mathcal{O} .

We have shown that \mathfrak{B}^* is a materialization of \mathcal{O} and \mathfrak{D}^u . Thus $\mathfrak{B}^* \not\models q_0(\vec{b})$. Now let I be the union of the set of partial isomorphisms between \mathfrak{D}^u and \mathfrak{D} from Lemma 2.11 and the partial isomorphisms between guarded sets induced by the obvious isomorphisms between $\mathfrak{B}_{c^*}^c$, $c \in \mathfrak{D}^u$, and the copy of $\mathfrak{B}_{c^*}^{\uparrow}$ hooked to \mathfrak{D} at c^{\uparrow} in \mathfrak{B} . I is a counting connected guarded bisimulation between \mathfrak{B}^* and \mathfrak{B} . Thus, by Lemma 2.11, \mathfrak{B} is a model of \mathcal{O} . Also $(\mathfrak{B}^*, \vec{b})$ and (\mathfrak{B}, \vec{a}) are connected guarded bisimilar and so $\mathfrak{B} \not\models q_0(\vec{a})$ since $\mathfrak{B}^* \not\models q_0(\vec{b})$, by Lemma 2.11 and since q can be regarded as a formula in openGF_2 . We have shown that $\mathcal{O}, \mathfrak{D} \not\models q_0(\vec{a})$, as required. \square

We can now prove the announced strong dichotomy result.

THEOREM 5.3. *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 $\text{PTIME} = \text{NP}$):*

- (1) \mathcal{O} is materializable;
- (2) \mathcal{O} is materializable for the class of cg-tree decomposable instances \mathfrak{D} with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$;
- (3) \mathcal{O} is unraveling 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) \Rightarrow (2) is by a compactness argument. (2) \Rightarrow (3) is Theorem 5.2. (3) \Rightarrow (4) follows from Theorem 4.5 and Theorems 3.7 and 3.8. (4) \Rightarrow (5) is folklore. (5) \Rightarrow (1) is Theorem 3.6 (assuming $\text{PTIME} \neq \text{NP}$). \square

The qualification ‘with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$ ’ in Point 2 of Theorem 5.3 can be dropped without compromising the correctness of the theorem, and the same is true for Theorem 5.2. It will be useful, though, in the decision procedures developed in Section 9.

6 CONNECTION TO CSP AND MMSNP

We establish the four CSP-hardness results displayed in the middle two rows of Figure 1 as well as the dichotomy result stated in the second lowest row. In contrast to the dichotomies proved in the previous section, this dichotomy is not ‘strong’ in the sense explained in the introduction, that is, it is established using a reduction to the dichotomy of CSPs (via a detour through the logical generalization MMSNP of CSP) rather than elementary proofs

and it does not establish that PTIME query evaluation coincides with Datalog[≠] rewritability. In fact, we use results on CSPs to show that the latter two notions do not coincide for the ontology languages considered here.

Let \mathfrak{A} be an instance. The *constraint satisfaction problem* $\text{CSP}(\mathfrak{A})$ is to decide, given an instance \mathfrak{D} with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathfrak{A})$, whether there is a homomorphism from \mathfrak{D} to \mathfrak{A} , which we denote with $\mathfrak{D} \rightarrow \mathfrak{A}$. In this context, \mathfrak{A} is called the *template* of $\text{CSP}(\mathfrak{A})$. The complement of $\text{CSP}(\mathfrak{A})$ is denoted $\text{coCSP}(\mathfrak{A})$. We will generally assume that \mathfrak{A} interprets relation symbols of arity at most two and that the template \mathfrak{A} *admits precoloring*, that is, for each $a \in \text{dom}(\mathfrak{A})$, there is a unary relation symbol P_a such that $P_a(b) \in \mathfrak{A}$ iff $b = a$ [28]. It is known that for every template \mathfrak{A} , there is a template \mathfrak{A}' of this form such that $\text{CSP}(\mathfrak{A})$ has the same complexity as $\text{CSP}(\mathfrak{A}')$ up to polynomial time reductions [5, 28]. Moreover, $\text{coCSP}(\mathfrak{A})$ is Datalog definable iff $\text{coCSP}(\mathfrak{A}')$ is Datalog definable [53].

Definition 6.1. 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} that admits precoloring and interprets relation symbols of arity at most two, there exists an \mathcal{L} ontology \mathcal{O} such that

- (1) there is a Boolean query $q_0 \in \mathcal{Q}$ such that for every instance \mathfrak{D} with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathfrak{A})$:
 $\mathfrak{D} \not\rightarrow \mathfrak{A}$ iff $\mathcal{O}, \mathfrak{D} \models q_0$.
- (2) for every $q \in \mathcal{Q}$, evaluating the OMQ (\mathcal{O}, q) reduces in polynomial time to $\text{coCSP}(\mathfrak{A})$.

Observe that it follows from Point 1 that $\text{coCSP}(\mathfrak{A})$ reduces in polynomial time to evaluating the OMQ (\mathcal{O}, q_0) .

The following theorem summarizes our results on CSP-hardness.

THEOREM 6.2. *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{ALCCF}_ℓ ontologies of depth 2.*

PROOF. We give the proof for $uGF_2(1, =)$ and then indicate the modifications needed for $uGF_2(1, f)$ and \mathcal{ALCCF}_ℓ ontologies of depth 2. For $uGF_2(2)$, the result follows from a proof of the corresponding result in [59] for \mathcal{ALC} ontologies of depth 3.

Let \mathfrak{A} be a template admitting precoloring and interpreting relation symbols of arity at most two. Let R_a be a binary relation symbol for each $a \in \text{dom}(\mathfrak{A})$, and set

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

Then \mathcal{O} contains

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

where A and R range over symbols in $\text{sig}(\mathfrak{A})$ 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 $a \in \text{dom}(\mathfrak{A})$ 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.

We show that \mathcal{O} satisfies Conditions 1 and 2 from Definition 6.1, where the query q_0 used in Condition 1 is $q_0 \leftarrow N(x)$ with N a fresh unary relation symbol.

For Condition 1, assume \mathfrak{D} with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathfrak{A})$ is given. We show that $\mathfrak{D} \rightarrow \mathfrak{A}$ iff $\mathcal{O}, \mathfrak{D} \not\models q_0$. First let h be a homomorphism from \mathfrak{D} to \mathfrak{A} . Define a model \mathfrak{B} of \mathcal{O} and \mathfrak{D} by adding to \mathfrak{D} for any $d \in \text{dom}(\mathfrak{D})$ with $h(d) = a$ an 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 constants $d_{i,d}$ for all $i > 0$. Also add $R_a(d, d)$ to \mathfrak{D} for all $a \in \text{dom}(\mathfrak{A})$, $d \in \text{dom}(\mathfrak{D})$, and all constants 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} . There is no b with $N(b) \in \mathfrak{B}$ and thus $\mathcal{O}, \mathfrak{D} \not\models q_0$, as required. Now assume that $\mathcal{O}, \mathfrak{D} \not\models q_0$. Then there is a model \mathfrak{B} of \mathcal{O} and \mathfrak{D} such that $\mathfrak{B} \not\models q_0$. 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 Condition 1.

For Condition 2, let q be a CQ. We show that the query evaluation problem for (\mathcal{O}, q) can be reduced in polynomial time to $\text{coCSP}(\mathfrak{A})$. We first show that there is a polynomial time reduction of the problem whether an instance \mathfrak{D} is consistent w.r.t. \mathcal{O} to $\text{CSP}(\mathfrak{A})$. Assume \mathfrak{D} is given. Let \mathfrak{D}^\bullet be the $\text{sig}(\mathfrak{A})$ -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 $\mathfrak{D}^\bullet \rightarrow \mathfrak{A}$. This provides the polynomial time reduction of consistency to $\text{CSP}(\mathfrak{A})$. Now let $\mathfrak{D}' = \mathfrak{D} \cup \{R_a(d, d) \mid a \in \text{dom}(\mathfrak{A}), d \in \text{dom}(\mathfrak{D})\}$. Clearly, the evaluation problem $\mathfrak{D}' \models q(\vec{d})$ is in PTIME. Observe that if an instance \mathfrak{D} is consistent w.r.t. \mathcal{O} , then one can construct a CQ-materialization \mathfrak{B} of \mathcal{O} and \mathfrak{D} such that there is a homomorphism from \mathfrak{B} to \mathfrak{D}' preserving $\text{dom}(\mathfrak{D})$ and vice versa. It follows that $\mathcal{O}, \mathfrak{D} \models q(\vec{d})$ iff \mathfrak{D} is not consistent w.r.t. \mathcal{O} or $\mathfrak{D}' \models q(\vec{d})$ and we have obtained the polynomial time reduction of query evaluation for (\mathcal{O}, q) to $\text{coCSP}(\mathfrak{A})$.

For $\text{uGF}_2(1, f)$, we modify the ontology \mathcal{O} defined above as follows. First, we state that a binary relation symbol F is a partial function satisfying $\forall x F(x, x)$. 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 $\exists y (R_a(x, y) \wedge F(x, y))$, respectively. The resulting ontology is in $\text{uGF}_2(1, f)$ and one can prove in exactly the same way as above that it is as required.

To construct an \mathcal{ALCF}_ℓ ontology of depth 2 with the required properties, replace in \mathcal{O} the formulas $\varphi_a^\neq(x)$ by $\exists^{\geq 2} y R_a(x, y)$ and $\varphi_a^=(x)$ by $\exists y R_a(x, y)$, respectively. The resulting ontology is equivalent to an \mathcal{ALCF}_ℓ ontology of depth 2 and one can prove in almost the same way as above that it is as required. \square

It is known that there exist templates \mathfrak{A} such that $\text{CSP}(\mathfrak{A})$ is in PTIME while $\text{coCSP}(\mathfrak{A})$ is not Datalog definable [33]. By the reductions provided in [5, 53], there also exists such a template \mathfrak{A} that additionally admits precoloring and interprets only relation symbols of arity at most two. As a consequence of the results in [34], $\text{coCSP}(\mathfrak{A})$ is not Datalog $^\neq$ definable either. It now follows directly from the definition that if a language \mathcal{L} is CSP-hard, then there exists an ontology \mathcal{O} in \mathcal{L} such that CQ-evaluation w.r.t. \mathcal{O} is in PTIME but not Datalog $^\neq$ -rewritable.

THEOREM 6.3. *In any of the following ontology languages \mathcal{L} there exist ontologies that enjoy PTIME CQ-evaluation but are not Datalog $^\neq$ -rewritable: $\text{uGF}_2(1, =)$, $\text{uGF}_2(2)$, $\text{uGF}_2(1, f)$, and the class of \mathcal{ALCF}_ℓ ontologies of depth 2.*

The ontology languages in Theorem 4.5 thus behave differently from the languages for which we proved a dichotomy in Section 5.

We next establish a dichotomy between PTIME and CONP-hardness for query evaluation in uGF_2 . Our proof proceeds via reduction to the logical generalization MMSNP of CSP introduced by Feder and Vardi [33], see also [16, 60, 62]. While MMSNP has higher expressive power than CSP, it has the same complexity: for every MMSNP sentence φ , there is a template \mathfrak{A} such that evaluating φ has the same complexity as $CSP(\mathfrak{A})$, up to polynomial time reductions [33, 51]. In particular, the dichotomy between PTIME and NP of CSPs is thus inherited by MMSNP. It is well-known that MMSNP has the same expressive power as the complement of Boolean monadic disjunctive Datalog [15]. Here, we prefer to work with the latter.

Monadic disjunctive Datalog (MDDLog) is a variation of Datalog (without inequality) in which all intensional relation symbols are monadic and where rule heads might be disjunctive. Thus, a *monadic disjunctive Datalog (MDDLog) rule* ρ has the form

$$S_1(x_1) \vee \cdots \vee S_m(x_m) \leftarrow R_1(\vec{y}_1) \wedge \cdots \wedge R_n(\vec{y}_n)$$

with $n > 0$ and $m \geq 0$ and where S_1, \dots, S_m are monadic relation symbols and R_1, \dots, R_n are relation symbols of unrestricted arity. As expected, we refer to $S_1(\vec{x}_1) \vee \cdots \vee S_m(\vec{x}_m)$ as the *head* of ρ , and to $R_1(\vec{y}_1) \wedge \cdots \wedge R_n(\vec{y}_n)$ as the *body*. As in Datalog, every variable that occurs in the head of ρ is required to also occur in the body of ρ .

A *monadic disjunctive Datalog (MDDLog) program* Π is a finite set of monadic disjunctive Datalog rules with a selected relation symbol goal that does not occur in rule bodies and appears only in non-disjunctive rules of the form $\text{goal}(\vec{x}) \leftarrow R_1(\vec{x}_1) \wedge \cdots \wedge R_n(\vec{x}_n)$. The arity of programs and intensional and extensional relation symbols are defined as for Datalog, and so is the semantics. When all extensional relation symbols in Π are from the signature Σ , we say that Π is *over extensional signature* Σ and assume that no other relation symbols occur in instances over which Π is evaluated. We refer to [30] for more information on disjunctive Datalog. We will sometimes use body atoms of the form $\text{true}(x)$ that are vacuously true for all elements of the active domain. This is just syntactic sugar since any rule with body atom $\text{true}(x)$ can equivalently be replaced by a set of rules obtained by replacing $\text{true}(x)$ in all possible ways with an atom $R(x_1, \dots, x_n)$ where R is a relation symbol from the extensional signature Σ and where $x_i = x$ for some i and all other x_i are fresh variables.

The problem to *evaluate* Π is to decide, given a Σ -instance \mathfrak{D} and $a_1, \dots, a_n \in \text{dom}(\mathfrak{D})$, whether $\mathfrak{D} \models \Pi(a_1, \dots, a_n)$. This problem is in CONP for every MDDLog program Π . We note the following dichotomy.

THEOREM 6.4. *Let Π be an MDDLog program. Then evaluating Π is in PTIME or CONP-complete.*

As explained above, for Boolean programs Theorem 6.4 is a consequence of the dichotomy between PTIME and NP for CSPs [18, 74] and the fact that Boolean MDDLog has the same expressive power as the complement of MMSNP. Moreover, it was observed in [35] that a dichotomy for Boolean MDDLog programs implies a dichotomy for MDDLog programs of unrestricted arity.

Our aim in this section is to establish the following result.

THEOREM 6.5. *Let \mathcal{O} be a uGF_2 ontology. Then query evaluation w.r.t. \mathcal{O} is in PTIME or CONP-complete.*

Due to Theorem 3.8, Theorem 6.5 can be established by proving a dichotomy for the class of all OMQs (\mathcal{O}, q) with \mathcal{O} from uGF_2 and q a unary rAQ. This is what we do in the following. Note, however, that Theorem 6.6 below even applies to GF_2 ontologies instead of only to uGF_2 ontologies. While this is stronger than what we need to establish Theorem 6.5, it remains open whether the strong result in Theorem 6.6 also holds for CQs and UCQs rather than only for rAQs.

THEOREM 6.6. *For every OMQ (\mathcal{O}, q) with \mathcal{O} a GF_2 ontology and q a unary rAQ, evaluation is in PTIME or CONP-complete.*

Let (\mathcal{O}, q) be an OMQ with \mathcal{O} a GF_2 ontology and q a rAQ with one answer variable. To show that evaluating (\mathcal{O}, q) is in PTIME or CONP-hard, we construct an MDDLog program Π that is equivalent to the OMQ (\mathcal{O}, q) in the sense that for all instances \mathfrak{D} , the certain answers to (\mathcal{O}, q) coincide with the answers to Π . Together with Theorem 6.4, this yields Theorem 6.6.

Let Σ be the set of relation symbols that occur in \mathcal{O} and q . Clearly, it suffices to consider evaluation of (\mathcal{O}, q) on instances that only contain symbols from Σ . From now on, we fix two variables x and y and assume that x is the answer variable of q . We denote by $\text{cl}(\mathcal{O}, q)$ the smallest set of formulas with at most x as their free variable that satisfies the following conditions: it contains all subformulas of \mathcal{O} with all free variables replaced with x and all subformulas of q which have exactly one free variable, renamed to x , and it is closed under applying single negation. Note that $\text{cl}(\mathcal{O}, q)$ contains q and all sentences from \mathcal{O} . A *type for \mathcal{O} and q* is a maximal satisfiable subset $t \subseteq \text{cl}(\mathcal{O}, q)$. We use $\text{tp}(\mathcal{O}, q)$ to denote the set of all types for \mathcal{O} and q . For an interpretation \mathfrak{A} and an $a \in \text{dom}(\mathfrak{A})$, we use $t^{\mathfrak{A}}(a)$ to denote the type realized at a in \mathfrak{A} , that is, $t^{\mathfrak{A}}(a) = \{\varphi(x) \in \text{cl}(\mathcal{O}, q) \mid \mathfrak{A} \models \varphi(a)\} \cup \{\varphi() \in \text{cl}(\mathcal{O}, q) \mid \mathfrak{A} \models \varphi()\}$. Note that the types defined here are similar but not identical to those used in the proof of Theorem 3.7.

A *link* is a (potentially empty) set of atomic formulas of the form $R(x, y)$ and $R(y, x)$. Let \mathfrak{A} be an interpretation. Then all $a, b \in \text{dom}(\mathfrak{A})$ give rise to a link

$$\mathcal{R}^{\mathfrak{A}}(a, b) := \{R(x, y) \mid R(a, b) \in \mathfrak{A}\} \cup \{R(y, x) \mid R(b, a) \in \mathfrak{A}\}.$$

A *typed link* is a triple t_1, \mathcal{R}, t_2 with t_1, t_2 types and \mathcal{R} a link. We say that t_1, \mathcal{R}, t_2 is *realizable* if there is a model \mathfrak{A} of \mathcal{O} and (not necessarily distinct) $a, b \in \text{dom}(\mathfrak{A})$ with $t^{\mathfrak{A}}(a) = t_1$, $\mathcal{R}^{\mathfrak{A}}(a, b) \supseteq \mathcal{R}$, and $t^{\mathfrak{A}}(b) = t_2$.

We now construct the desired MDDLog program Π . Introduce a fresh unary relation symbol P_t for every type t , to be used as intensional relation symbols in Π . The program comprises the following rules:

$$\begin{array}{ll} \bigvee_{t \in \text{tp}(\mathcal{O}, q)} P_t(x) \leftarrow \text{true}(x) & \\ \text{goal}(x) \leftarrow P_t(x) & \text{whenever } q \in t \\ \perp \leftarrow P_{t_1}(x) \wedge \mathcal{R}(x, y) \wedge P_{t_2}(y) & \text{for all typed links } t_1, \mathcal{R}, t_2 \\ & \text{that are not realizable} \end{array}$$

where $\mathcal{R}(x, y)$ denotes the conjunction over all atoms in the link \mathcal{R} .

Informally, these rules ‘guess’ of a model \mathfrak{A} of \mathcal{O} and \mathfrak{D} that is partial in the sense that we only explicitly represent the restriction of \mathfrak{A} to the constants in $\text{dom}(\mathfrak{D})$ while all relevant information about other constants in $\text{dom}(\mathfrak{A})$ is summarized in the types that we assign to the constants in $\text{dom}(\mathfrak{D})$. The type t guessed via P_t in the first line determines which formulas

from $\text{cl}(\mathcal{O}, q)$ are made true at a constant in $\text{dom}(\mathfrak{D})$. The second line ensures that whenever the query is true at a constant a in the guessed model, then a is returned as an answer. And the third line guarantees that the guessed types ‘fit together’; as an example note that, when we have guessed $P_{t_1}(a)$ with $A(x) \in t_1$, $R(b, a) \in \mathfrak{D}$, and $\text{cl}(\mathcal{O}, q) \ni \vartheta := \exists y R(x, y) \wedge A(y)$, then we must guess a type t_2 for b with $\vartheta \in t_2$. Theorem 6.6 follows from the following lemma which we prove in detail in the appendix.

LEMMA 6.7. *For all Σ -instances \mathfrak{D} and $a \in \text{dom}(\mathfrak{D})$, $\mathcal{O}, \mathfrak{D} \models q(a)$ iff $\mathfrak{D} \models \Pi(a)$.*

7 UNDECIDABILITY

We show that ontology languages which admit both sentences of depth 2 and relation symbols interpreted as partial functions tend to be computationally problematic. In particular, the languages considered here 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[≠]-rewritable, or CONP-hard, and whether \mathcal{O} is materializable. In this section, we establish the undecidability results. The technique introduced here is used in the subsequent section to prove non-dichotomy results.

THEOREM 7.1. *For the ontology languages $uGF_2^-(2, f)$ and \mathcal{ALCIF}_ℓ 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[≠]-rewritable, or CONP-hard (unless PTIME = NP);*
- (2) *\mathcal{O} is materializable.*

The remainder of this section is devoted to the proof of Theorem 7.1. The proof is by reduction of the undecidable rectangle tiling problem, defined as follows. An instance $\mathfrak{P} = (\mathfrak{T}, H, V)$ of the *rectangle tiling problem* is given by a finite non-empty set \mathfrak{T} of *tile types* including an *initial tile type* T_{init} to be placed only on the lower left corner and a *final tile type* T_{final} to be placed only on the upper right corner, a *horizontal matching relation* $H \subseteq \mathfrak{T} \times \mathfrak{T}$ and a *vertical matching relation* $V \subseteq \mathfrak{T} \times \mathfrak{T}$. A *tiling* for (\mathfrak{T}, H, V) is a map $f : \{0, \dots, n\} \times \{0, \dots, m\} \rightarrow \mathfrak{T}$ such that $n, m \geq 0$, $f(0, 0) = T_{\text{init}}$, $f(n, m) = T_{\text{final}}$, $f(i, j) \in \mathfrak{T} \setminus \{T_{\text{init}}, T_{\text{final}}\}$ for all $(i, j) \notin \{(0, 0), (n, m)\}$, $(f(i, j), f(i+1, j)) \in H$ for all $i < n$ and $j \leq m$, and $(f(i, j), f(i, j+1)) \in V$ for all $i \leq n$ and $j < m$. 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 [72].

To establish Theorem 7.1, it suffices to construct, 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 $\mathcal{O}_{\mathfrak{P}}$ is materializable and query evaluation w.r.t. $\mathcal{O}_{\mathfrak{P}}$ is Datalog[≠]-rewritable.

The rectangle to be tiled is represented in input instances using the binary relation symbols X and Y , and $\mathcal{O}_{\mathfrak{P}}$ declares these relation symbols and their inverses to be functional. The basic 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[≠]-rewritable. 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 parts of the input. In \mathcal{ALCCIF}_ℓ , we use formulas of the form $\exists^{=1}yP(x, y)$ while additionally stating in $\mathcal{O}_{\mathfrak{P}}$ that $\forall x\exists yP(x, y)$. 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 as being true at some constant in the input instance. 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 various technical problems we have to address in the construction below, cannot be avoided. On $\text{uGF}_2^-(2, f)$, we work with the marker $\neg\exists y(P(x, y) \wedge \neg F(x, y))$, where F is a functional relation symbol for which we state $\forall xF(x, x)$. Also here, we can preset the marker negatively but not positively.

We now provide the detailed construction of $\mathcal{O}_{\mathfrak{P}}$, in two steps: we first construct an ontology $\mathcal{O}_{\text{cell}}$ that marks the lower left corner 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} . The ontologies are formulated in \mathcal{ALCCIF}_ℓ . Thus, in addition to \mathcal{ALCI} concepts we use concepts of the form $(\leq 1R)$, $(= 1R)$, and $(\geq 2R)$. As formulas can be written more succinctly in DL notation compared to FO notation, we use the former.

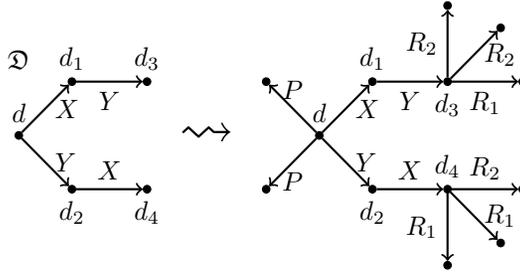
Marking the lower left corner of grid cells. Let X and Y be binary relation symbols and X^-, Y^- their inverses in \mathcal{ALCI} . Using the sentences

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

for all $Z \in \{X, Y, X^-, Y^-\}$ we ensure that in any instance \mathfrak{D} consistent w.r.t. our ontology the relation symbols 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 symbol P . For P and all binary relation symbols 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 set the marker $(= 1P)$ 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

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

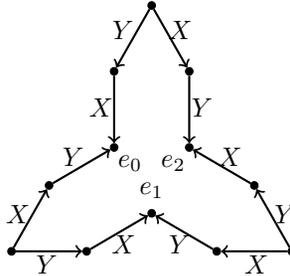
Clearly, if $\mathfrak{D} \models \text{cell}(d)$ then $\mathcal{O}, \mathfrak{D} \models (= 1P)(d)$ for the resulting ontology \mathcal{O} . Conversely, the idea is that if $\mathfrak{D} \not\models \text{cell}(d)$ and $X(d, d_1), Y(d, d_2), Y(d_1, d_3), X(d_2, d_4) \in \mathfrak{D}$ but $d_3 \neq d_4$, 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 inclusion does not work yet. First, the inclusion has depth 3 and we are aiming at an inclusion of depth 2. This 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 $(= 1R_i^{XY})$


 Fig. 2. $\mathfrak{D} \not\models \text{cell}(d) \Rightarrow \mathcal{O}, \mathfrak{D} \not\models (= 1P)(d)$

and the sentences

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

Details are given below. More importantly, the implication ‘ $\mathfrak{D} \not\models \text{cell}(d) \Rightarrow \mathcal{O}, \mathfrak{D} \not\models (= 1P)(d)$ ’ does not hold. There are two reasons for this. 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 . Second, d_3, d_4 might be on an odd cycle of mutually distinct $e_0, e_1, \dots, e_n \in \mathfrak{D}$ such that each e_i reaches $e_{(i+1) \bmod n+1}$ via a Y^-X^-YX -path in \mathfrak{D} , for $i = 0, 1, \dots, n$. Figure 3 illustrates this for $n = 2$. Then, since in at least two neighboring $e_i, e_{(i+1) \bmod n+1}$ 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 $\text{cell}(d)$. The first problem is easily dealt with by demanding the implication to be true only if \mathfrak{D} is consistent w.r.t. our ontology. The second problem is resolved by adding appropriate axioms enforcing that also in this case \mathfrak{D} is not consistent w.r.t. our ontology.


 Fig. 3. $\mathfrak{D} \models (= 1P)(d) \not\Rightarrow \mathfrak{D} \models \text{cell}(d)$

In detail, the ontology $\mathcal{O}_{\text{cell}}$ 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 relation symbols 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 relation symbols $Q \in \text{AUX}_{\text{cell}}$. 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

$$\mathcal{O}_{\text{cell}}, \mathfrak{D} \models q(\vec{d}) \quad \text{iff} \quad \{\top \sqsubseteq \exists Q.\top \mid Q \in \text{AUX}_{\text{cell}}\}, \mathfrak{D} \models q(\vec{d}) \quad (3)$$

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 $Z \in \{X, Y, X^-, Y^-\}$:

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

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 $\mathcal{O}_{\text{cell}}$ contains the following axioms in addition to $\top \sqsubseteq \exists Q.\top$ for $Q \in \text{AUX}_{\text{cell}}$:

- (1) Functionality of X, Y, X^- and Y^- is stated using

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

for $Z \in \{X, Y, X^-, Y^-\}$.

- (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 neighboring nodes on the cycles in \mathfrak{D} introduced above:

$$(= 1R_1^{X^-Y^-XY}) \sqcap (= 1R_2^{X^-Y^-XY}) \sqsubseteq (= 1R_1) \sqcap (= 1R_2)$$

$$(= 1R_1^{Y^-X^-YX}) \sqcap (= 1R_2^{Y^-X^-YX}) \sqsubseteq (= 1R_1) \sqcap (= 1R_2)$$

- (6) The auxiliary sentences $\exists^W(= 1R_i)$ for all relation symbols R_i^W used above.

LEMMA 7.2. *The ontology $\mathcal{O}_{\text{cell}}$ has the following properties for all instances \mathfrak{D} :*

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

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. 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 relation symbols and take the binary relation symbols X, Y, X^-, Y^- from above and an additional

set AUX_{grid} of binary relation symbols F, F^X, F^Y, U, R, L, B , and A . The ontology $\mathcal{O}_{\mathfrak{P}}$ is defined by adding to $\mathcal{O}_{\text{cell}}$ the sentences

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

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

$$\begin{aligned} T_{\text{final}} &\sqsubseteq (= 1F) \sqcap (= 1U) \sqcap (= 1R) \\ \exists X. ((= 1U) \sqcap (= 1F) \sqcap T_j) \sqcap T_i &\sqsubseteq (= 1U) \sqcap (= 1F) \\ \exists Y. ((= 1R) \sqcap (= 1F) \sqcap T_\ell) \sqcap T_i &\sqsubseteq (= 1R) \sqcap (= 1F) \\ \exists Y. (= 1F) &\sqsubseteq (= 1F^Y) \\ \exists X. (= 1F) &\sqsubseteq (= 1F^X) \\ \exists X. (T_j \sqcap (= 1F) \sqcap (= 1F^Y)) \sqcap & \\ \exists Y. (T_\ell \sqcap (= 1F) \sqcap (= 1F^X)) \sqcap (= 1P) \sqcap T_i &\sqsubseteq (= 1F) \\ (= 1F) \sqcap T_{\text{init}} &\sqsubseteq (= 1A) \sqcap (= 1B) \sqcap (= 1L) \\ \bigsqcup_{1 \leq s < t \leq p} T_s \sqcap T_t &\sqsubseteq \perp \\ (= 1U) \sqsubseteq \forall Y. \perp \quad (= 1R) \sqsubseteq \forall X. \perp \quad (= 1U) \sqsubseteq \forall X. (= 1U) \quad (= 1R) \sqsubseteq \forall Y. (= 1R) \\ (= 1B) \sqsubseteq \forall Y^-. \perp \quad (= 1L) \sqsubseteq \forall X^-. \perp \quad (= 1B) \sqsubseteq \forall X. (= 1B) \quad (= 1L) \sqsubseteq \forall Y. (= 1L) \end{aligned}$$

Fig. 4. Additional Axioms of $\mathcal{O}_{\mathfrak{P}}$

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

- $(= 1U)$, $(= 1R)$, $(= 1L)$, and $(= 1B)$ mark the upper, right, left, and bottom 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_{init} 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 relation symbols F^X and F^Y are used to avoid depth 3 sentences in the same way as the relation symbols 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 $\rho: \{0, \dots, n\} \times \{0, \dots, m\} \rightarrow \text{dom}(\mathfrak{D})$ with $\rho(0, 0) = d$ such that

- for all $j \leq n$, $k \leq m$, and all tile types T : $T(\rho(j, k)) \in \mathfrak{D}$ 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 $(b_1, b_2) = (\rho(j, k), \rho(j + 1, k))$;
- 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 $(b_1, b_2) = (\rho(j, k), \rho(j, k + 1))$;

- the range of ρ is a maximally connected component in the graph $(\text{dom}(\mathfrak{D}), X \cup X^- \cup Y \cup Y^-)$: if $d \in \text{ran}(\rho)$ and $Z(d, d') \in \mathfrak{D}$ for some $Z \in \{X, Y, X^-, Y^-\}$, then $d' \in \text{ran}(\rho)$.

We then call d the *root* of the $n \times m$ -grid with witness function ρ for \mathfrak{P} . The following result can now be proved using Lemma 7.2.

LEMMA 7.3. *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 ρ such that $\text{dom}(\mathfrak{D}) = \text{ran}(\rho)$, and all relation symbols 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 use Lemma 7.3 to prove the undecidability result. Let $\mathcal{O} = \mathcal{O}_{\mathfrak{P}} \cup \{ (=1A) \sqsubseteq E_1 \sqcup E_2 \}$, where E_1 and E_2 are unary relation symbols.

LEMMA 7.4. (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 \mathcal{O} is materializable and 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 all $i < n$ and $j \leq m$, $Y((i, j), (i, j+1))$, for all $i \leq n$ and $j < m$, and, for every tile type T , $T(i, j)$ if $f(i, j) = T$, for all $i \leq n$ and $j \leq m$. Then \mathfrak{D} is consistent w.r.t. \mathcal{O} and $\mathcal{O}, \mathfrak{D} \models (=1A)(0, 0)$, by Lemma 7.3. Thus $\mathcal{O}, \mathfrak{D} \models E_1(0, 0) \vee E_2(0, 0)$ but $\mathcal{O}, \mathfrak{D} \not\models E_1(0, 0)$ and $\mathcal{O}, \mathfrak{D} \not\models E_2(0, 0)$. Thus \mathcal{O} is not materializable and CQ-evaluation is coNP-hard.

(2) Assume \mathfrak{P} does not admit a tiling. Clearly, any instance \mathfrak{D} is consistent w.r.t. \mathcal{O} iff it is consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$. Thus, by Lemma 7.3, if $\mathcal{O}, \mathfrak{D} \models (=1A)(d)$ for some $d \in \text{dom}(\mathfrak{D})$, then \mathfrak{D} is not consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$. It follows that $\mathcal{O}, \mathfrak{D} \models q(\vec{d})$ iff $\mathcal{O}_{\mathfrak{P}}, \mathfrak{D} \models q(\vec{d})$ for every CQ q and \vec{d} in $\text{dom}(\mathfrak{D})$. Thus, by Points 1 and 3 of Lemma 7.3, \mathcal{O} is materializable and CQ-evaluation w.r.t. \mathcal{O} is Datalog $^{\neq}$ -rewritable, as required. \square

Lemma 7.4 entails Theorem 7.1 for \mathcal{ALCF}_{ℓ} 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 relation symbols X, Y, X^-, Y^- are defined as functions and it is stated that X^- and Y^- are the inverse relations 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 the axiom $\top \sqsubseteq \exists R. \top$ 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 7.2 and Lemma 7.3 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.

8 NON-DICHOTOMY

We prove the two non-dichotomy results shown in the topmost section of Figure 1, reusing some of the techniques from the previous section. Ideally, we would like to use the existence

of NP-intermediate word problems of Turing machines as asserted by Ladner's theorem [52] to establish our results. However, this does not appear to be easily possible. In fact, in Section 7 it was important to use CSPs that admit precoloring rather than standard CSPs and, very informally spoken, in this section we need a version of Ladner's theorem that in a similar sense admits precoloring. We find it in the form of the run fitting problem for Turing machines which asks whether a given partially described run of a Turing machine (that corresponds to a precoloring) can be extended to a full run which is accepting. We use an adaptation of the proof of Ladner's theorem to show that there are NP-intermediate run fitting problems.

THEOREM 8.1. *For the ontology languages $uGF_2^-(2, f)$ and \mathcal{ALCLF}_ℓ of depth 2, there is no dichotomy between PTIME and CONP (unless PTIME = CONP).*

We consider *non-deterministic Turing machines* (TMs, for short) with a single one-sided infinite tape. 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. A *configuration* of M is represented by a string vwq , where q is the state, v is the inscription of the tape to the left of the tape head, and w is the inscription of the tape to the right of the tape 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 $|\gamma_0| = \dots = |\gamma_n|$ such that $\gamma_0 = q_0w$ for some string w that may contain blanks. Note that, since w may contain blanks, γ_0 does not necessarily correspond to the initial configuration of M for a given input string. A run is *accepting* if its last configuration is accepting. We assume that the accepting state has no successor states.

Definition 8.2. 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, $\tilde{\gamma}[i]$ denotes the symbol that occurs at the i -th position of $\tilde{\gamma}$. 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 $|\tilde{\gamma}_0| = \dots = |\tilde{\gamma}_m|$. 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 8.3. The *run fitting problem* for a TM 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 the run fitting problem for a TM M is in NP. We prove the following result in Appendix G by a careful adaptation of the proof of Ladner's theorem given in [3].

THEOREM 8.4. *There is a TM whose run fitting problem is neither in PTIME nor NP-hard, unless PTIME = NP.*

Now Theorem 8.1 is a consequence of the following lemma.

LEMMA 8.5. *For every Turing machine M , there is a $uGF_2^-(2, f)$ ontology \mathcal{O} and an \mathcal{ALCLF}_ℓ ontology \mathcal{O} of depth 2 such that the following hold, where N is a distinguished unary relation symbol:*

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

(2) for every CQ q , evaluating the OMQ (\mathcal{O}, q) is reducible in polynomial time to the complement of the run fitting problem for M .

PROOF. We use a grid construction and marker formulas as in the proof of Theorem 7.1, with the grid providing the space in which the run of the TM is simulated and markers represent TM states and tape symbols. In fact, we re-use the ontology $\mathcal{O}_{\mathfrak{F}}$ from the proof of Theorem 7.1, for 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}_ℓ and $\text{uGF}_2^-(2, f)$, we represent states q and tape symbols G using the same formulas as in the CSP encoding. Thus, for \mathcal{ALCIF}_ℓ we use formulas $\exists^{\geq 2}yq(x, y)$ and $\exists^{\geq 2}yG(x, y)$, respectively, using q and G as binary relation symbols. Note that here the encoding $\exists^{\geq 1}yq(x, y)$ 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.

We give the detailed proof for \mathcal{ALCIF}_ℓ ontologies of depth 2. The proof for $\text{uGF}_2^-(2, f)$ is obtained by modifying the \mathcal{ALCIF}_ℓ ontology in the same way as in the proof of Theorem 7.1 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 E written everywhere else. To re-use the notation and results from the proof of Theorem 7.1 we regard such a structure as a tiling with tile types $\mathfrak{T} = \{E, T_{\text{final}}, T_{\text{init}}\}$. Then the ontology $\mathcal{O}_{\mathfrak{F}}$ for $\mathfrak{F} = (\mathfrak{T}, H, V)$ and

$$\begin{aligned} H &= \{(E, E), (E, T_{\text{final}}), (T_{\text{init}}, E)\} \\ V &= \{(E, E), (E, T_{\text{final}}), (T_{\text{init}}, E)\} \end{aligned}$$

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 $\mathcal{O}_{\mathfrak{F}}, \mathfrak{D} \models (=1A)(d)$. \mathcal{O}_M uses in addition to the binary relation symbols in $\mathcal{O}_{\mathfrak{F}}$ binary relation symbols $q \in Q$ and $G \in \Gamma$ that occur in concepts $(\geq 2q)$ and $(\geq 2G)$ and indicate that M is in state q and that G is written on the tape, respectively. The sentences of \mathcal{O}_M are now as follows:

- (a) The grid in which the lower left corner is marked with $(=1A)$ is completely colored with $(=1A)$:

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

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

- (b) q_0 is the first symbol of the first configuration and no $q \in Q$ occurs later in the first configuration:

$$(=1A) \cap T_{\text{init}} \sqsubseteq (\geq 2q_0), \quad (=1A) \cap (=1B) \cap (\geq 2q) \sqsubseteq (=1L)$$

- (c) Every grid point is colored with exactly one $(\geq 2H)$ for $H \in \Gamma \cup Q$:

$$(=1A) \sqsubseteq \bigsqcup_{H \in \Gamma \cup Q} ((\geq 2H) \cap \bigsqcap_{H' \in (\Gamma \cup Q) \setminus \{H\}} (=1H'))$$

- (d) To avoid sentences of depth larger than two we introduce for $W \in \{X, X^-\}$ and $S \in Q \cup \Gamma$ fresh binary relation symbols S^W and the sentences

$$(=1A) \cap (\geq 2S^W) \equiv (=1A) \cap \exists W.(\geq 2S)$$

- (e) 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 . Then add the following sentence to \mathcal{O}_M

$$\begin{aligned} & (= 1A) \sqcap \exists X^-. (\geq 2G_0) \sqcap (\geq 2q) \sqcap \exists X. (\geq 2G_1) \sqsubseteq \\ & \bigsqcup_{S_1S_2S_3 \in S(G_0qG_1)} \exists Y. ((\geq 2S_1^{X^-}) \sqcap (\geq 2S_2) \sqcap (\geq 2S_3^X)) \end{aligned}$$

- (f) Symbols written on cells with distance at least two from the position of the head are not changed. For all $G, G_1, G_2 \in \Gamma$:

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

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

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

- (h) Finally, for AUX_M the set of fresh binary relation symbols used above \mathcal{O}_M contains:

$$\{ \top \sqsubseteq \exists Q. \top \mid Q \in \text{AUX}_M \}$$

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

(1) 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 time 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 $\tilde{\gamma} = (\tilde{\gamma}_0, \tilde{\gamma}_1, \dots, \tilde{\gamma}_m)$ of partial configurations $\tilde{\gamma}_i$ of M such that $\tilde{\gamma}_0$ starts with q_0 and $|\tilde{\gamma}_0| = \dots = |\tilde{\gamma}_m| = n + 1$ is given. We define an instance \mathfrak{D} with $\mathfrak{D} \models \text{grid}(0, 0)$ 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

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

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

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

for distinct fresh constants $d_{i,j}^1$ and $d_{i,j}^2$ 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 $\tilde{\gamma}$.

(2) We have to provide for every CQ $q(\vec{x})$ a polynomial time 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 CQ $q(\vec{x})$, 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 $\{ \top \sqsubseteq \exists Q. \top \mid Q \in \text{AUX} \}, \mathfrak{D} \models q(\vec{a})$, where $\text{AUX} = \text{AUX}_{\text{cell}} \cup \text{AUX}_{\text{grid}} \cup \text{AUX}_M$.

As the second problem in Point (2) is in PTIME it suffices to provide a polynomial time reduction of the consistency problem for instances \mathfrak{D} 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. $\mathcal{O}_{\mathfrak{F}}$ (Lemma 7.3). If not, we are done. If yes, let

$$I = \{ d \in \text{dom}(\mathfrak{D}) \mid \mathfrak{D} \models \text{grid}(d) \}.$$

For each $d \in I$ we find natural numbers n_d, m_d such that d is the root of an $n_d \times m_d$ -grid with witness function ρ_d for the tiling problem \mathfrak{P} . By Lemma 7.3, there is a materialization \mathfrak{B} of \mathfrak{D} and $\mathcal{O}_{\mathfrak{P}}$ such that $d \in I$ iff $d \in (=1A)^{\mathfrak{B}}$.

Next we check in polynomial time that \mathfrak{D} is consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$ and the axioms from (a) and (d). To check consistency w.r.t. $\mathcal{O}_{\mathfrak{P}}$ and the axiom from (a), it suffices to check that no e in the range of some ρ_d has two or more A -successors. To check that \mathfrak{D} is consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$ and the sentences $(=1A) \cap (\geq 2S^W) \equiv (=1A) \cap \exists W.(\geq 2S)$ with $W \in \{X, X^-\}$ from (d) it suffices to check that every e in the range of some ρ_d with at least two S^W -successors in \mathfrak{D} has a W -successor in \mathfrak{D} . This can be done in polynomial time. If the answer is yes, we may assume that \mathfrak{D} is saturated in the sense that if, for example, $(=1A) \cap (\geq 2S^X) \equiv (=1A) \cap \exists X.(\geq 2S)$ is in \mathcal{O}_M then for any e in the range of some ρ_d 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, if \mathfrak{D} is not consistent w.r.t. $\mathcal{O}_{\mathfrak{P}}$ and the axioms from (a) and (d), then we are done. Moreover, if there exist $d \in I$ and natural numbers j, r with $0 \leq r \leq n_d$ and $0 \leq j \leq m_d$ such that there are distinct $S, S' \in \Gamma \cup Q$ such that $\rho_d(j, r)$ has at least two S -successors and at least two S' -successors in \mathfrak{D} , then \mathfrak{D} is not consistent w.r.t. \mathcal{O} (as this condition contradicts the axioms in (c)) and we are done as this can clearly be checked in polynomial time). Otherwise, define for every $d \in I$ the sequences of strings

$$\tilde{\gamma}^d = (\tilde{\gamma}_0^d, \dots, \tilde{\gamma}_{m_d}^d)$$

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

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

By construction, the sequences $\tilde{\gamma}^d$, $d \in I$, are well-defined partial runs of M . It is now straightforward to show that \mathfrak{D} is consistent w.r.t. \mathcal{O} iff for each $d \in I$ there exists an accepting run of M that matches $\tilde{\gamma}^d$. This provides a polynomial time reduction of the consistency problem for instances \mathfrak{D} w.r.t. \mathcal{O} to the run fitting problem for M . \square

9 DECIDABILITY RESULTS

In Section 7, we have shown that it is undecidable whether a given ontology admits PTIME query evaluation when the ontologies are formulated in $\text{uGF}_2^-(2, f)$ or are \mathcal{ALCIF} ontologies of depth 2. The aim of this section is to identify cases where this problem is decidable. In fact, we show decidability and EXPTIME-completeness for \mathcal{ALCHIQ} ontologies of depth 1 and a NEXPTIME upper bound for $\text{uGC}_2^-(1, =)$ ontologies. In both languages, PTIME query evaluation coincides with rewritability into Datalog^{\neq} and thus our results can also be viewed as establishing decidability of Datalog^{\neq} rewritability. As discussed in the introduction, we have carried out experiments which show that a large majority of real world ontologies are \mathcal{ALCHIQ} ontologies of depth 1 or can be transformed into such ontologies in a complexity preserving way, which demonstrates the practical relevance of the obtained results.

THEOREM 9.1. *For $\text{uGC}_2^-(1, =)$ ontologies, it is in NEXPTIME to decide whether query evaluation is in PTIME (equivalently: rewritable into Datalog^{\neq}). For \mathcal{ALCHIQ} ontologies of depth 1, this problem is EXPTIME-complete and the lower bound already holds for \mathcal{ALC} TBoxes of depth 1.*

We remark that the satisfiability problem for \mathcal{ALCHIQ} ontologies is also EXPTIME-complete, both for ontologies of depth 1 and for ontologies of unrestricted depth. Thus, the EXPTIME upper bound in Theorem 9.1 is the best one can hope for. Also note that because of [27] (Corollary 6.9) and [18], it is decidable and NP-complete whether a given CSP has

PSPACE complexity. This, however, does not imply any of the results in Theorem 9.1 in an obvious way because a CSP corresponds to an ontology with a fixed query while we quantify over all possible queries.

The proof of Theorem 9.1 proceeds through a series of lemmas that are of independent interest. Since the ontology languages considered here admit at most binary relation symbols, 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 thus speak of *tree interpretations* rather than of cg-tree decomposable interpretations. *Tree instances* are defined likewise. A tree interpretation is *irreflexive* if there exists no fact of the form $R(b, b)$ in \mathfrak{B} . The *outdegree* of \mathfrak{B} is the outdegree of $G_{\mathfrak{B}}$. The main insight underlying the proof of Theorem 9.1 is that for ontologies formulated in the mentioned languages, materializability (which by Theorem 5.3 coincides with PSPACE 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-neighborhood* $\mathfrak{B}_a^{\leq 1}$ of a in \mathfrak{B} as $\mathfrak{B}|_X$, where X is the union of all guarded sets in \mathfrak{B} that contain a . We say that \mathfrak{B} is a *bouquet with root a* if $\mathfrak{B}_a^{\leq 1} = \mathfrak{B}$.

LEMMA 9.2. *Let \mathcal{O} be a $\text{uGC}_2^-(1, =)$ ontology (resp. an ALCHIQ ontology of depth 1). Then \mathcal{O} is materializable iff \mathcal{O} is materializable for instances \mathfrak{D} that are bouquets (resp. irreflexive bouquets) of outdegree $\leq |\mathcal{O}|$ and satisfy $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$.*

PROOF. We start by introducing the basic notions used in the proof. An interpretation \mathfrak{D} is \mathcal{O} -saturated if $\{R(\vec{a}) \mid \mathcal{O}, \mathfrak{D} \models R(\vec{a}), \vec{a} \text{ tuple in } \text{dom}(\mathfrak{D})\} \subseteq \mathfrak{D}$. For every \mathcal{O} and instance \mathfrak{D} there exists a unique minimal (w.r.t. set-inclusion) \mathcal{O} -saturated instance $\mathfrak{D}^\bullet \supseteq \mathfrak{D}$ called *the \mathcal{O} -saturation of \mathfrak{D}* . Observe that $\text{dom}(\mathfrak{D}^\bullet) = \text{dom}(\mathfrak{D})$. We list the basic properties of \mathcal{O} -saturated instances. Let $\mathfrak{D} \subseteq \mathfrak{D}'$ be instances with $\mathfrak{D}'|_{\text{dom}(\mathfrak{D})} = \mathfrak{D}$ and let \mathcal{O} be a $\text{uGC}_2(=)$ ontology. Assume \mathfrak{D}' is consistent w.r.t. \mathcal{O} . Then the following hold:

- (a) 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} ; moreover, the materializations are the same.
- (b) If \mathfrak{B} is a materialization of \mathcal{O} and \mathfrak{D} and \mathfrak{D} is \mathcal{O} -saturated, then $\mathfrak{B}|_{\text{dom}(\mathfrak{D})} = \mathfrak{D}$.
- (c) If \mathfrak{D}' is \mathcal{O} -saturated, then \mathfrak{D} is \mathcal{O} -saturated.

We first give the proof of Lemma 9.2 for $\text{uGC}_2^-(1, =)$ ontologies, starting without the condition on the outdegree. Let $\Sigma_0 = \text{sig}(\mathcal{O})$ and assume that \mathcal{O} is materializable for the class of all Σ_0 -bouquets. By Theorem 5.3 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} consistent w.r.t. \mathcal{O} . By Point (a), we may assume that \mathfrak{D} is \mathcal{O} -saturated. Note that a forest model materialization \mathfrak{B} of \mathcal{O} and any \mathcal{O} -saturated instance \mathfrak{F} can be obtained by taking the union of \mathfrak{F} and the tree interpretations \mathfrak{B}_a , $a \in \text{dom}(\mathfrak{F})$, hooked to \mathfrak{F} at a in \mathfrak{B} . 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 in the interpretation \mathfrak{B}_a hooked to $\mathfrak{D}_a^{\leq 1}$ at a in a forest model materialization $\mathfrak{B}(a)$ of \mathcal{O} and $\mathfrak{D}_a^{\leq 1}$ (such a forest model materialization exists since $\mathfrak{D}_a^{\leq 1}$ is materializable). Let $\mathfrak{A} = \mathfrak{D} \cup \bigcup_{a \in \text{dom}(\mathfrak{D})} \mathfrak{B}_a$ be the resulting interpretation. We show that \mathfrak{A} is a materialization of \mathcal{O} and \mathfrak{D} . Clearly \mathfrak{A} is a model of \mathfrak{D} . It is a model of \mathcal{O} since the axioms in \mathcal{O} have depth at most one and since $\mathfrak{D}_a^{\leq 1} = \mathfrak{B}(a)|_{\text{dom}(\mathfrak{D}_a^{\leq 1})}$ for every $a \in \text{dom}(\mathfrak{D})$, by Points (b) and (c). The condition $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ iff $\mathfrak{A} \models q(\vec{a})$, for every CQ $q(\vec{x})$ and \vec{a} in $\text{dom}(\mathfrak{D})$, follows directly from the condition that the interpretations $\mathfrak{B}(a)$ are materializations of \mathcal{O} and $\mathfrak{D}_a^{\leq 1}$.

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). By Point (a), we may assume that \mathfrak{D} is \mathcal{O} -saturated. \mathcal{O} consists of sentences of the form $\forall x(x = x \rightarrow \psi(x))$, where $\psi(x)$ is a formula in openGC₂ of depth 1. Take for any subformula $\chi = \exists^{\geq n} z(\alpha(z, x) \wedge \varphi(z, x))$ occurring in such a ψ from \mathcal{O} the set

$$Z_\chi = \{b \neq a \mid \mathfrak{D} \models \alpha(b, a) \wedge \varphi(b, a)\}$$

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 \mathcal{O} and \mathfrak{D}' . Assume for a proof by contradiction that there is a materialization \mathfrak{B} of \mathcal{O} and \mathfrak{D}' . Let \mathfrak{B}' be the union of $\mathfrak{D} \cup \mathfrak{B}$ and the interpretations \mathfrak{B}_b , $b \in \text{dom}(\mathfrak{D}) \setminus (Z \cup \{a\})$, hooked to $\mathfrak{D}|_{\{a,b\}}$ at b in a forest model materialization of \mathcal{O} and $\mathfrak{D}|_{\{a,b\}}$ (here we use the condition that for tree instances with at most two constants materializations exist). 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 and Points (b) and (c), one can show that the restriction $\mathfrak{B}'|_{\text{dom}(\mathfrak{D})}$ of \mathfrak{B}' to $\text{dom}(\mathfrak{D})$ 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} . This finishes the proof for $\text{uGC}_2^-(1, =)$.

The proof for \mathcal{ALCHIQ} ontologies of depth 1 is similar. To show, however, that it suffices to consider irreflexive bouquets one has to replace the notion of unraveling introduced above by irreflexive counterparts (which are well known from previous work on \mathcal{ALCHIQ} [37]). For every interpretation \mathfrak{A} (interpreting at most binary relation symbols) and $a \in \text{dom}(\mathfrak{A})$ one can construct the irreflexive unraveling \mathfrak{A}_a^u of \mathfrak{A} at a into an irreflexive tree interpretation satisfying the same \mathcal{ALCHIQ} concept inclusions as \mathfrak{A} and also satisfying in a the same \mathcal{ALCHIQ} concepts as \mathfrak{A} . Irreflexive unraveling can then be used to define the *irreflexive global unraveling* of a data instance which behaves, when restricted to \mathcal{ALCHIQ} ontologies, in exactly the same way as the global unraveling defined above. In what follows we use the following easily proved variations of Lemma 2.12 and Theorem 5.3. Let \mathcal{O} be an \mathcal{ALCHIQ} ontology of depth 1. Then the following hold:

- (d) Let \mathfrak{A} be a model of \mathcal{O} and an irreflexive tree instance \mathfrak{D} . Then there exists an irreflexive tree model \mathfrak{B} of \mathcal{O} and \mathfrak{D} such that there exists a homomorphism h from \mathfrak{B} to \mathfrak{A} preserving $\text{dom}(\mathfrak{D})$.
- (e) \mathcal{O} is materializable iff \mathcal{O} is materializable for the class of all irreflexive tree instances \mathfrak{D} with $\text{dom}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$.

To prove the claim for \mathcal{ALCHIQ} and irreflexive bouquets in Lemma 9.2, one can now use Point (e) and modify in a straightforward way the proof given for $\text{uGC}_2^-(1, =)$. \square

We now develop algorithms that decide PTIME query evaluation by checking materializability of instances that are bouquets as stated in Lemma 9.2. Let \mathfrak{D} be an instance that is a bouquet with root a . An interpretation $\mathfrak{B} \supseteq \mathfrak{D}$ is a *1-materialization of \mathcal{O} and \mathfrak{D} with root a* if it is a bouquet and

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

For brevity, we say that \mathfrak{D} is *\mathcal{O} -relevant* if it is consistent w.r.t. \mathcal{O} , of outdegree at most $|\mathcal{O}|$, and satisfies $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$.

We show that when checking materializability of *ALCHIQ* ontologies of depth 1, not only is it sufficient to consider irreflexive \mathcal{O} -relevant bouquets instead of unrestricted instances, but additionally one can concentrate on 1-materializations of such bouquets.

LEMMA 9.3. *Let \mathcal{O} be an *ALCHIQ* ontology of depth 1. Then \mathcal{O} is materializable iff for all irreflexive \mathcal{O} -relevant bouquets \mathcal{D} there is a 1-materialization of \mathcal{O} and \mathcal{D} .*

PROOF. Let \mathcal{D} be an irreflexive \mathcal{O} -relevant bouquet with root a . Assume that for all irreflexive \mathcal{O} -relevant bouquets \mathfrak{F} with root b there exists a 1-materialization \mathfrak{B} of \mathcal{O} and \mathfrak{F} with root b (we then call the triple $(\mathfrak{F}, b, \mathfrak{B})$ a *1-materializability witness*). It suffices to prove that there exists a materialization of \mathcal{O} and \mathcal{D} . Note that it follows from Point (d) in the proof of Lemma 9.2 that any \mathfrak{B} in any 1-materializability witness $(\mathfrak{F}, b, \mathfrak{B})$ is an irreflexive tree interpretation. We construct the desired materialization step-by-step using these 1-materializability witnesses and also memorizing sets of frontier elements that have to be expanded in the next step. We start with a 1-materializability witness $(\mathcal{D}, a, \mathfrak{B}^0)$ and set $F_0 = \text{dom}(\mathfrak{B}^0) \setminus \{a\}$. Then we construct a sequence of irreflexive tree interpretations $\mathfrak{B}^0 \subseteq \mathfrak{B}^1 \subseteq \dots$ 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 b' in \mathfrak{B}^i and a 1-materializability witness $(\mathfrak{B}_{\{b',b\}}^i, b, \mathfrak{B}_b)$ and set

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

Let $\mathfrak{B}^* := \bigcup_{i \geq 0} \mathfrak{B}^i$. We show that \mathfrak{B}^* is a materialization of \mathcal{O} and \mathcal{D} . \mathfrak{B}^* is a model of \mathcal{O} by construction since \mathcal{O} is an *ALCHIQ* ontology of depth 1. We show that \mathfrak{B}^* is hom-universal. Consider a model \mathfrak{A} of \mathcal{O} and \mathcal{D} . It suffices to construct a homomorphism h from \mathfrak{B}^* to \mathfrak{A} preserving $\text{dom}(\mathcal{D})$. By Point (d) in the proof of Lemma 9.2, we may assume that \mathfrak{A} is an irreflexive tree interpretation. We construct h as the limit of a sequence $h_0 \subseteq h_1 \subseteq \dots$ of homomorphisms h_i from \mathfrak{B}^i to \mathfrak{A} . By definition, there exists a homomorphism h_0 from \mathfrak{B}^0 to $\mathfrak{A}_{a \leq 1}^{\leq 1}$ preserving $\text{dom}(\mathcal{D})$. 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 b' 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 1-materializability witness $(\mathfrak{B}_{\{b',b\}}^i, b, \mathfrak{B}_b)$. But then, as \mathfrak{B}_b is a 1-materialization of $\mathfrak{B}_{\{b',b\}}^i$ and h_i is injective on $\mathfrak{B}_{\{b',b\}}^i$ (since \mathfrak{A} is irreflexive), we can expand the homomorphism h_i to a homomorphism from domain $\text{dom}(\mathfrak{B}^i) \cup \{c\}$ into \mathfrak{A} . Thus, we can expand h_i to a homomorphism from \mathfrak{B}^{i+1} to \mathfrak{A} . \square

Lemma 9.3 implies that an *ALCHIQ* ontology \mathcal{O} of depth 1 enjoys PTIME query evaluation if and only if for all irreflexive \mathcal{O} -relevant bouquets \mathcal{D} there exists a 1-materialization of \mathcal{O} and \mathcal{D} . The latter condition can be checked in deterministic exponential time, as follows.

We enumerate all irreflexive \mathcal{O} -relevant bouquets \mathcal{D} . There are only single exponentially many candidates for \mathcal{D} and for each of them we can check in EXPTIME whether it is indeed \mathcal{O} -relevant. Note that this involves checking whether \mathcal{D} is consistent w.r.t. \mathcal{O} and that consistency of instances w.r.t. an ontology can be decided in EXPTIME in *ALCHIQ* [7].

For each \mathcal{O} -relevant bouquet \mathcal{D} , we have to check the existence of a 1-materialization. Using a straightforward selective filtration argument one can show that if an interpretation \mathfrak{A} is an irreflexive tree model of \mathcal{O} and \mathcal{D} , then there exists a subinterpretation \mathfrak{A}' of \mathfrak{A} of outdegree at most $2|\mathcal{O}|$ that satisfies \mathcal{O} and \mathcal{D} . It follows that we can concentrate on 1-materializations \mathfrak{B} of outdegree at most $2|\mathcal{O}|$. We can also assume that $\text{sig}(\mathfrak{B}) \subseteq \text{sig}(\mathcal{O})$. Thus, there are

only single exponentially many candidates for \mathfrak{B} . We enumerate all of them and have to verify Conditions (1) and (2) of 1-materializations. For Condition (1), we have to check whether there is a model \mathfrak{A} of \mathcal{O} and \mathfrak{D} such that $\mathfrak{B} = \mathfrak{A}_a^{\leq 1}$. This can be done in EXPTIME by a straightforward polynomial time reduction to the consistency of an instance w.r.t. an \mathcal{ALCHIQ} ontology. For Condition (2), it suffices to check whether for every bouquet $\mathfrak{A} \supseteq \mathfrak{D}$ with root a and outdegree $\leq 2|\mathcal{O}|$ and with $\text{sig}(\mathfrak{A}) \subseteq \text{sig}(\mathcal{O})$ such that there exists a model \mathfrak{C} of \mathcal{O} with $\mathfrak{C}_a^{\leq 1} = \mathfrak{A}$ there exists a homomorphism from \mathfrak{B} to \mathfrak{A} preserving \mathfrak{D} . This can again be achieved by combining enumeration with consistency checks.

We have thus established the EXPTIME upper bound in Theorem 9.1. A matching lower bound (even for \mathcal{ALC} ontologies of depth 1) can be proved by a straightforward reduction of the (un)satisfiability problem for ontologies: an \mathcal{ALC} ontology \mathcal{O} is unsatisfiable if and only if $\mathcal{O} \cup \{\top \sqsubseteq B_1 \sqcup B_2\}$ is materializable, where B_1, B_2 are fresh unary relation symbols.

The proof of Lemma 9.3 makes intense use of irreflexive tree interpretations to define appropriate unravelings for \mathcal{ALCHIQ} ontologies. This does not work for $\text{uGC}_2^-(1, =)$. In fact, the following example shows that there, the existence of 1-materializations does not guarantee the materializability of bouquets.

Example 9.4. Let S, S', R, R' be binary relation symbols and consider the ontology \mathcal{O} that consists of the sentences

$$\begin{aligned} \forall x ((S(x, x) \wedge R(x, x)) \rightarrow \exists y (R(x, y) \wedge (x \neq y)) \vee \exists y (S(x, y) \wedge (x \neq y))) \\ \forall x (\exists y (W(y, x) \wedge (y \neq x)) \rightarrow \exists y W'(x, y)) \end{aligned}$$

where (W, W') ranges over $\{(R, R'), (S, S')\}$. Note that \mathcal{O} is equivalent to an $\text{uGF}_2^-(1, =)$ ontology. \mathcal{O} is not materializable since for $\mathfrak{D} = \{S(a, a), R(a, a)\}$, every model of \mathcal{O} and \mathfrak{D} contains an atom of the form $R'(c, c')$ or $S'(c, c')$, but not necessarily both. It is, however, not too difficult to verify that for every bouquet \mathfrak{D} there exists a 1-materialization of \mathcal{O} and \mathfrak{D} .

In $\text{uGC}_2^-(1, =)$, we thus have to check unrestricted materializability of \mathcal{O} -relevant bouquets, instead of 1-materializability. To achieve this, we use a mosaic approach. In each mosaic piece, we record a 1-neighborhood of the materialization of the bouquet, a 1-neighborhood of a tree-model of the bouquet and the 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 9.3. We actually introduce two different kinds of mosaic pieces, one kind of piece explicitly addressing reflexive loops which, as illustrated by Example 9.4, 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.

A *bounded 1-materializability witness* for \mathcal{O} is a tuple $(\mathfrak{F}, a, \mathfrak{B})$ such that

- \mathfrak{F} is an \mathcal{O} -relevant bouquet with root a ;
- \mathfrak{B} is a 1-materialization of \mathcal{O} and \mathfrak{F} with root a and of outdegree $\leq 2|\mathcal{O}|$.

A pair (a, \mathfrak{B}) with \mathfrak{B} a bouquet with root a is called a *1-model* for \mathcal{O} if $\text{sig}(\mathfrak{B}) \subseteq \text{sig}(\mathcal{O})$, the outdegree of \mathfrak{B} is $\leq 2|\mathcal{O}|$, and there exists a model \mathfrak{A} of \mathcal{O} with $a \in \text{dom}(\mathfrak{A})$ such that $\mathfrak{A}_a^{\leq 1} = \mathfrak{B}$. We require the following two types of mosaic pieces. First, an *injective hom-pair* takes the form $(\mathfrak{F}, a, \mathfrak{B}) \rightarrow_h^i (a', \mathfrak{B}')$ such that

- $(\mathfrak{F}, a, \mathfrak{B})$ is a bounded 1-materializability witness for \mathcal{O}
- (a', \mathfrak{B}') is a 1-model of \mathcal{O} ;
- h is a homomorphism from \mathfrak{B} to \mathfrak{B}' mapping a to a' which is injective on \mathfrak{F} .

In an injective hom-pair, the 1-materializability witness is a piece of the materialization we wish to construct and the 1-model is a piece of the model into which we wish to homomorphically embed the materialization. Injective hom-pairs cover the case in which the homomorphism one wants to extend (i.e., the restriction of h to $\text{dom}(\mathfrak{F})$) is injective. To deal with non-injective homomorphisms (as indicated by Example 9.4) we also consider *contracting hom-pairs* which take the form $(\mathfrak{F}, a, \mathfrak{B}) \rightarrow_h^c (a', \mathfrak{B}')$ where

- $(\mathfrak{F}, a, \mathfrak{B})$ is a bounded 1-materializability witness for \mathcal{O} with $\text{dom}(\mathfrak{F}) = \{a, b\}$ for some constant b ;
- (a', \mathfrak{B}') is a 1-model of \mathcal{O} ;
- h is a homomorphism from \mathfrak{B} to \mathfrak{B}' with $h(a) = h(b) = a'$.

The following lemma now provides a NEXPTIME decision procedure checking materializability of $\text{uGC}_2^-(1, =)$ ontologies.

LEMMA 9.5. *Let \mathcal{O} be a $\text{uGC}_2^-(1, =)$ ontology. Then \mathcal{O} is materializable iff there exist*

- (1) *a set M of bounded 1-materializability witnesses containing exactly one bounded 1-materializability witness $(\mathfrak{F}, a, \mathfrak{B})$ for every \mathcal{O} -relevant bouquet \mathfrak{F} with root a and*
- (2) *sets H of injective hom-pairs and E of contracting hom-pairs whose first components are all in M*

such that the following conditions hold:

- (a) *if $(\mathfrak{F}, a, \mathfrak{B}) \in M$, (a', \mathfrak{B}') is a 1-model of \mathcal{O} , and h_0 an injective homomorphism from \mathfrak{F} to \mathfrak{B}' with $h_0(a) = a'$, then there is an extension h of h_0 with $(\mathfrak{F}, a, \mathfrak{B}) \rightarrow_h^i (a', \mathfrak{B}') \in H$;*
- (b) *if $(\mathfrak{F}, a, \mathfrak{B}) \rightarrow_h^i (a', \mathfrak{B}') \in H$ or $(\mathfrak{F}, a, \mathfrak{B}) \rightarrow_h^c (a', \mathfrak{B}') \in E$ with $b \in \text{dom}(\mathfrak{B}) \setminus \text{dom}(\mathfrak{F})$ and $h(a) = h(b) = a'$, then there are h' and \mathfrak{B}'' with $(\mathfrak{B}_{\{a,b\}}, b, \mathfrak{B}'') \rightarrow_{h'}^c (a', \mathfrak{B}') \in E$.*

PROOF. Using Lemma 3.3 and selective filtrations as in the EXPTIME proof above, one can easily show that sets M , H , and E satisfying the Conditions (a) and (b) exist if \mathcal{O} is materializable. Conversely, let the sets M , H , and E satisfy the Conditions (a) and (b). Assume an \mathcal{O} -relevant bouquet \mathfrak{D} with root a is given. It suffices to prove that there exists a materialization of \mathfrak{D} and \mathcal{O} . Take the bounded 1-materializability witness $(\mathfrak{D}, a, \mathfrak{B}) \in M$. As in the proof of Lemma 9.3 we construct a sequence of interpretations $\mathfrak{B}^0 \subseteq \mathfrak{B}^1 \subseteq \dots$ and sets $F_i \subseteq \text{dom}(\mathfrak{B}^i)$ of frontier elements (but now using only bounded 1-materializability witnesses in M): set $\mathfrak{B}^0 := \mathfrak{B}$ and $F_0 = \text{dom}(\mathfrak{B}) \setminus \{a\}$. If \mathfrak{B}^i and F_i have been constructed, then take for any $b \in F_i$ its predecessor b' and a bounded 1-materializability witness $(\mathfrak{B}_{\{b',b\}}^i, b, \mathfrak{B}_b) \in M$ and set $\mathfrak{B}^{i+1} := \mathfrak{B}^i \cup \bigcup_{b \in F_i} \mathfrak{B}_b$ and $F_{i+1} := \bigcup_{b \in F_i} \text{dom}(\mathfrak{B}_b) \setminus \{b\}$. Let $\mathfrak{B}^* := \bigcup_{i \geq 0} \mathfrak{B}^i$. 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 a $\text{uGC}_2^-(1, =)$ ontology. We show that \mathfrak{B}^* is hom-universal. Consider a model \mathfrak{A} of \mathcal{O} and \mathfrak{D} . We construct a homomorphism h from \mathfrak{B}^* to \mathfrak{A} preserving $\text{dom}(\mathfrak{D})$ as the limit of a sequence $h_0 \subseteq h_1 \subseteq \dots$ of homomorphisms from \mathfrak{B}^i to \mathfrak{A} . As argued above, we may assume that the outdegree of \mathfrak{A} is $\leq 2|\mathcal{O}|$. By definition, there exists a homomorphism h_0 from \mathfrak{B}^0 to $\mathfrak{A}_{\leq 1}^a$ 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 b' of b in \mathfrak{B}^{i-1} :

- (1) *if $h_i(b') \neq h_i(b)$, then for the bounded 1-materializability witness $(\mathfrak{B}_{\{b',b\}}^*, b, \mathfrak{B}_b) \in M$ there exists a homomorphism h_b which coincides with h_i on $\{b, b'\}$ such that*

$$(\mathfrak{B}_{\{b',b\}}^*, b, \mathfrak{B}_b) \rightarrow_{h_b}^i (h_i(b), \mathfrak{A}_{h_i(b)}^{\leq 1}) \in H$$

- (2) if $h_i(b') = h_i(b)$, then for the bounded 1-materializability witness $(\mathfrak{B}_{\{b',b\}}^*, b, \mathfrak{B}_b) \in M$ there exists a homomorphism h_b which coincides with h_i on $\{b, b'\}$ such that

$$(\mathfrak{B}_{\{b',b\}}^*, b, \mathfrak{B}_b) \rightarrow_{h_b}^c (h_i(b), \mathfrak{A}_{h_i(b)}^{\leq 1}) \in E$$

Assume h_i with the properties (1) and (2) has been constructed. Then we take for all $b \in F_i$ and the predecessor b' of b the homomorphism h_b given by Condition (1) and, respectively, (2) and set $h_{i+1} := h_i \cup \bigcup_{b \in F_i} h_b$. Using the Conditions (a) and (b) for M , H , and E it is straightforward to show that h_{i+1} again satisfies Condition (1) and (2). \square

As, up to isomorphism, the sets M , H , and E are of size at most single exponential in \mathcal{O} and since the conditions of Lemma 9.5 can be checked in polynomial time, we obtain a NEXPTIME procedure for deciding materializability.

Theorem 9.1 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-completeness of query evaluation, 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 9.6. 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 tree instances 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 symbol. The presence of this chain is verified by propagating a marker upwards along the chain. To ensure that \mathcal{O} is materializable for tree instances of depth smaller than $2^n - 1$, we represent this marker by a universally quantified formula and also hide some other unary relation symbols in the same way. For each unary relation symbol 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:

$$\begin{aligned} & \overline{X}_1(x) \wedge \cdots \wedge \overline{X}_n(x) \rightarrow H_V(x) \\ & X_i(x) \wedge \exists R.(X_i(y) \wedge \overline{X}_j(y)) \rightarrow H_{\text{ok}_i}(x) \\ & \overline{X}_i(x) \wedge \exists R.(\overline{X}_i(y) \wedge \overline{X}_j(y)) \rightarrow H_{\text{ok}_i}(x) \\ & X_i(x) \wedge \exists R.(\overline{X}_i(y) \wedge X_1(y) \wedge \cdots \wedge X_{i-1}(y)) \rightarrow H_{\text{ok}_i}(x) \\ & \overline{X}_i(x) \wedge \exists R.(X_i(y) \wedge X_1(y) \wedge \cdots \wedge X_{i-1}(y)) \rightarrow H_{\text{ok}_i}(x) \\ & H_{\text{ok}_1}(x) \wedge \cdots \wedge H_{\text{ok}_n}(x) \wedge \exists R.H_V(y) \rightarrow H_V(x) \\ & \exists R.X_i(y) \wedge \exists R.\overline{X}_i(y) \rightarrow \perp \\ & X_1(x) \wedge \cdots \wedge X_n(x) \wedge H_V(x) \rightarrow B_1(x) \vee B_2(x) \end{aligned}$$

where x is universally quantified, $\exists R.\varphi(y)$ is an abbreviation for $\exists y(R(x, y) \wedge \varphi(y))$, i ranges over $1..n$, and j over $1..i - 1$. Note that X_1, \dots, X_n and $\overline{X}_1, \dots, \overline{X}_n$ 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.

10 CONCLUSION

Perhaps the most surprising result of our analysis is that it is possible to escape Ladner's Theorem and prove a strong dichotomy between DataLog^{\neq} -rewritability and CONP-completeness

of query evaluation for rather large subsets of the guarded fragment that cover many practically relevant DL ontologies. Ontology languages covered by this positive result further enjoy the property that Datalog[≠]-rewritability, materializability, unraveling tolerance, and PTIME query evaluation are all equivalent notions, and in several cases we even observe decidability of meta problems such as deciding whether a given ontology admits PTIME query evaluation. Our study also shows that increasing the expressive power in seemingly harmless ways often results in CSP-hardness and in Datalog[≠]-rewritability diverging from PTIME query evaluation, and in several cases even in a provable loss of the PTIME/CONP dichotomy. The proof of the latter comes with a variation of Ladner’s theorem we believe to be potentially useful also in other contexts where some form of precoloring of the input is unavoidable, such as in consistent query answering for which the topic of precoloring is discussed in [58].

There are a number of interesting future research questions. The main open question regarding dichotomies is whether the PTIME/CONP dichotomy can be generalized from uGF_2 to $uGF(=)$ and even to $GF(=)$. This appears to be very challenging as it is related to a difficult open question in the area of constraint satisfaction problems (CSPs): whether the logic $MMSNP_2$ [61], also known as $GMSNP$ [15], has a dichotomy between PTIME and NP. The equivalence of the two questions has been shown in [15] for the setup where queries are not quantified and we conjecture that it is also true in the case of universally quantified queries studied in this paper.

Another interesting question is whether the CSP-hardness results established in this paper that do not come with a dichotomy (third row of Figure 1) can be shown to admit a dichotomy or to provably not enjoy a dichotomy. Also of interest is the decidability and complexity of the problem to decide PTIME query evaluation for $uGF(1)$, $uGF_2^-(2)$, and for \mathcal{ALCHIF} ontologies of depth 2. It would further be interesting to study fragments of GF in which 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 symbol is transitive.

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.

REFERENCES

- [1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
- [2] Hajnal Andréka, István Németi, and Johan van Benthem. 1998. Modal Languages and Bounded Fragments of Predicate Logic. *J. Philosophical Logic* 27, 3 (1998), 217–274.
- [3] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press.
- [4] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyashev. 2009. The DL-Lite Family and Relations. *J. of Artificial Intelligence Research* 36 (2009), 1–69.
- [5] Albert Atserias. 2008. On digraph coloring problems and treewidth duality. *Eur. J. Comb.* 29, 4 (2008), 796–820.
- [6] Franz Baader, Sebastian Brandt, and Carsten Lutz. 2005. Pushing the \mathcal{EL} Envelope. In *Proc. of IJCAI*. 364–369.
- [7] Franz Baader, Deborah, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (Eds.). 2003. *The Description Logic Handbook*. Cambridge University Press.
- [8] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. 2017. *An Introduction to Description Logic*. Cambridge University Press.

- [9] Vince Bárány, Georg Gottlob, and Martin Otto. 2014. Querying the Guarded Fragment. *Logical Methods in Computer Science* 10, 2 (2014).
- [10] Andrew Bate, Boris Motik, Bernardo Cuenca Grau, Frantisek Simancik, and Ian Horrocks. 2016. Extending Consequence-Based Reasoning to SRIQ. In *Proc. of KR*. 187–196.
- [11] Catriel Beeri and Philip A. Bernstein. 1979. Computational Problems Related to the Design of Normal Form Relational Schemas. *ACM Trans. Database Syst.* 4, 1 (1979), 30–59.
- [12] Catriel Beeri and Moshe Y. Vardi. 1984. A Proof Procedure for Data Dependencies. *J. ACM* 31, 4 (1984), 718–741.
- [13] Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. 2016. First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics. In *Proceedings of IJCAI*. 965–971.
- [14] Meghyn Bienvenu and Magdalena Ortiz. 2015. Ontology-Mediated Query Answering with Data-Tractable Description Logics. In *Proc. of Reasoning Web*. 218–307.
- [15] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. 2014. Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Syst.* 39, 4 (2014), 33:1–33:44.
- [16] Manuel Bodirsky, Hubie Chen, and Tomás Feder. 2012. On the Complexity of MMSNP. *SIAM J. Discrete Math.* 26, 1 (2012), 404–414.
- [17] Pierre Bourhis, Marco Manna, Michael Morak, and Andreas Pieris. 2016. Guarded-Based Disjunctive Tuple-Generating Dependencies. *ACM Trans. Database Syst.* 41, 4 (2016), 27:1–27:45.
- [18] Andrei A. Bulatov. 2017. A Dichotomy Theorem for Nonuniform CSPs. In *Proc. of FOCS*. 319–330.
- [19] Andrea Cali, Georg Gottlob, and Michael Kifer. 2013. Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. *J. Artif. Intell. Res. (JAIR)* 48 (2013), 115–174.
- [20] Andrea Cali, Georg Gottlob, and Andreas Pieris. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193 (2012), 87–128.
- [21] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2013. Data complexity of query answering in description logics. *Artificial Intelligence* 195 (2013), 335–360.
- [22] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *J. Autom. Reasoning* 39, 3 (2007), 385–429.
- [23] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2000. View-Based Query Processing and Constraint Satisfaction. In *Proc. of LICS*. 361–371.
- [24] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2003. View-Based query containment. In *Proc. of PODS*. 56–67.
- [25] Stefano Ceri, Georg Gottlob, and Letizia Tanca. 1989. What you Always Wanted to Know About Datalog (And Never Dared to Ask). *IEEE Trans. Knowl. Data Eng.* 1, 1 (1989), 146–166.
- [26] C. C. Chang and H. Jerome Keisler. 1990. *Model Theory*. Studies in Logic and the Foundations of Mathematics, Vol. 73. Elsevier.
- [27] Hubie Chen and Benoit Larose. 2016. Asking the metaquestions in constraint tractability. *CoRR* abs/1604.00932 (2016).
- [28] David Cohen and Peter Jeavons. 2006. The complexity of constraint languages. In *Handbook of Constraint Programming*. Elsevier, Chapter 8.
- [29] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. 2008. The chase revisited. In *Proc. of PODS*. ACM, 149–158.
- [30] Thomas Eiter, Georg Gottlob, and Heikki Mannila. 1997. Disjunctive Datalog. *ACM Trans. Database Syst.* 22, 3 (1997), 364–418.
- [31] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. 2012. Query Rewriting for Horn-SHIQ Plus Rules. In *Proc. of AAAI*.
- [32] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336, 1 (2005), 89–124.
- [33] Tomás Feder and Moshe Y. Vardi. 1998. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM J. Comput.* 28, 1 (1998), 57–104.
- [34] Tomás Feder and Moshe Y. Vardi. 2003. Homomorphism Closed vs. Existential Positive. In *Proc. of LICS*. 311–320.
- [35] Cristina Feier, Antti Kuusisto, and Carsten Lutz. 2018. Rewritability in Monadic Disjunctive Datalog, MMSNP, and Expressive Description Logics. *ACM Transactions of Database Systems* (2018).

- [36] Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer. <https://doi.org/10.1007/3-540-29953-X>
- [37] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. 2008. Conjunctive Query Answering for the Description Logic *SHIQ*. *J. Artif. Intell. Res. (JAIR)* 31 (2008), 157–204.
- [38] Georg Gottlob, Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, Thomas Schwentick, and Michael Zakharyashev. 2014. The price of query rewriting in ontology-based data access. *Artif. Intell.* 213 (2014), 42–59.
- [39] Georg Gottlob, Marco Manna, and Andreas Pieris. 2015. Polynomial Rewritings for Linear Existential Rules. In *Proc. of IJCAI*. 2992–2998.
- [40] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. 2014. Query Rewriting and Optimization for Ontological Databases. *ACM Trans. Database Syst.* 39, 3 (2014), 25:1–25:46.
- [41] Erich Grädel. 1999. On The Restraining Power of Guards. *J. Symb. Log.* 64, 4 (1999), 1719–1742.
- [42] Erich Grädel and Martin Otto. 2014. The Freedoms of (Guarded) Bisimulation. In *Johan van Benthem on Logic and Information Dynamics*. 3–31.
- [43] André Hernich, Carsten Lutz, Fabio Papacchini, and Frank Wolter. 2017. Dichotomies in Ontology-Mediated Querying with the Guarded Fragment. In *Proc. of PODS*. 185–199.
- [44] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. 2005. Data Complexity of Reasoning in Very Expressive Description Logics. In *Proc. of IJCAI*. 466–471.
- [45] Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. 2016. Datalog rewritability of Disjunctive Datalog programs and non-Horn ontologies. *Artif. Intell.* 236 (2016), 90–118.
- [46] Yevgeny Kazakov. 2004. A Polynomial Translation from the Two-Variable Guarded Fragment with Number Restrictions to the Guarded Fragment. In *Proc. of JELIA*. 372–384.
- [47] Roman Kontchakov and Michael Zakharyashev. 2014. An Introduction to Description Logics and Query Rewriting. In *Proc. of Reasoning Web*. 195–244.
- [48] Adila Krisnadhi and Carsten Lutz. 2007. Data Complexity in the EL family of DLs. In *Proc. of DL*.
- [49] Andrei A. Krokhin and Stanislav Zivny (Eds.). 2017. *The Constraint Satisfaction Problem: Complexity and Approximability*. Dagstuhl Follow-Ups, Vol. 7. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.
- [50] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. 2013. Complexities of Horn Description Logics. *ACM Trans. Comput. Log.* 14, 1 (2013), 2:1–2:36.
- [51] Gábor Kun. 2007. Constraints, MMSNP, and Expander Structures. (2007). Available at <http://arxiv.org/abs/0706.1701v1>.
- [52] Richard E. Ladner. 1975. On the Structure of Polynomial Time Reducibility. *J. ACM* 22, 1 (1975), 155–171.
- [53] Benoit Larose and Pascal Tesson. 2009. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.* 410, 18 (2009), 1629–1647.
- [54] Hector J. Levesque and Gerhard Lakemeyer. 2000. *The logic of knowledge bases*. MIT Press. <https://mitpress.mit.edu/books/logic-knowledge-bases>
- [55] Carsten Lutz. 2008. The Complexity of Conjunctive Query Answering in Expressive Description Logics. In *Proc. of IJCAR*. 179–193.
- [56] Carsten Lutz, Robert Piro, and Frank Wolter. 2011. Description Logic TBoxes: Model-Theoretic Characterizations and Rewritability. In *Proc. of IJCAI*. IJCAI/AAAI, 983–988.
- [57] Carsten Lutz and Frank Wolter. 2012. Non-Uniform Data Complexity of Query Answering in Description Logics. In *Proc. of KR*.
- [58] Carsten Lutz and Frank Wolter. 2015. On the Relationship between Consistent Query Answering and Constraint Satisfaction Problems. In *Proc. of ICDT*. 363–379.
- [59] Carsten Lutz and Frank Wolter. 2017. The Data Complexity of Description Logic Ontologies. *Logical Methods in Computer Science* Volume 13, Issue 4 (Nov. 2017).
- [60] Florent R. Madelaine. 2009. Universal Structures and the Logic of Forbidden Patterns. *Logical Methods in Computer Science* 5, 2 (2009).
- [61] Florent R. Madelaine. 2009. Universal Structures and the logic of Forbidden Patterns. *Logical Methods in Computer Science* 5, 2 (2009). <http://arxiv.org/abs/0904.2521>
- [62] Florent R. Madelaine and Iain A. Stewart. 2007. Constraint Satisfaction, Logic and Forbidden Patterns. *SIAM J. Comput.* 37, 1 (2007), 132–163.
- [63] Jack Minker (Ed.). 1988. *Foundations of Deductive Databases and Logic Programming*. Elsevier.
- [64] Marie-Laure Mugnier and Michaël Thomazo. 2014. An Introduction to Ontology-Based Query Answering with Existential Rules. In *Proc. of Reasoning Web*. 245–278.

- [65] Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. 2008. Data Complexity of Query Answering in Expressive Description Logics via Tableaux. *Journal of Automated Reasoning* 41, 1 (2008), 61–98.
- [66] Martin Otto. 2012. Highly acyclic groups, hypergraph covers, and the guarded fragment. *J. ACM* 59, 1 (2012), 5:1–5:40.
- [67] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. *J. Data Semantics* 10 (2008), 133–173.
- [68] Ian Pratt-Hartmann. 2007. Complexity of the Guarded Two-variable Fragment with Counting Quantifiers. *J. Log. Comput.* 17, 1 (2007), 133–155.
- [69] Ian Pratt-Hartmann. 2009. Data-complexity of the two-variable fragment with counting quantifiers. *Inf. Comput.* 207, 8 (2009), 867–888.
- [70] Andrea Schaerf. 1993. On the Complexity of the Instance Checking Problem in Concept Languages with Existential Quantification. *J. of Intel. Inf. Systems* 2 (1993), 265–278.
- [71] Frantisek Simancik, Yevgeny Kazakov, and Ian Horrocks. 2011. Consequence-Based Reasoning beyond Horn Ontologies. In *Proc. of IJCAI*. 1093–1098.
- [72] P. van Emde Boas. 1997. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*. 331–363.
- [73] Patricia L Whetzel, Natalya F Noy, Nigam H Shah, Paul R Alexander, Csongor Nyulas, Tania Tudorache, and Mark A Musen. 2011. 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 (2011), W541–W545.
- [74] Dmitriy Zhuk. 2017. A Proof of CSP Dichotomy Conjecture. In *Proc. of FOCS*. 331–342.

A PROOFS FOR SECTION 2

Theorem 2.2 (restated) *A sentence in GF(=) (resp. GF) is invariant under disjoint unions iff it is equivalent to a sentence in uGF(=) (resp. uGF).*

PROOF. The direction from right to left is easy. We prove the converse direction for GF(=); the proof for GF is similar and omitted.

We first 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 replace any subformula of φ of the form $\exists x, y(x = y \wedge \psi(x, y))$ or $\forall x, y(x = y \rightarrow \psi(x, y))$ by $\neg \forall x(x = x \rightarrow \neg \psi(x, x))$ and $\forall x(x = x \rightarrow \psi(x, x))$, respectively. In the resulting sentence, replace any formula of the form $\forall y(x = y \rightarrow \psi(x, y))$ or $\exists y(x = y \wedge \psi(x, y))$ and with x, y distinct variables by $\psi(x, x)$. Denote the resulting GF(=) sentence by φ' . Call a sentence ψ *simple* if it contains no subsentence within the scope of a guarded quantifier. Observe that any simple subsentence of φ' is a Boolean combination of sentences in uGF(=). Now we apply the following equivalent rewriting exhaustively to $\chi := \varphi'$: if ψ is a simple subsentence of χ within the scope of a guarded quantifier, then rewrite χ into $(\chi[\psi/\text{true}] \wedge \psi) \vee (\chi[\psi/\text{false}] \wedge \neg \psi)$, where **true** and **false** are the standard propositional constants (which can be eliminated in the well-known way). It is straightforward to show that the resulting sentence is a Boolean combination of sentences in uGF(=).

Consider now a Boolean combination φ of uGF(=) sentences and assume that φ is 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 suffices 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 a Boolean combination of uGF(=) sentences. \square

B PROOFS FOR SECTION 3

Lemma 3.4 (restated) *There exists a materializable ontology \mathcal{O} in uGF(2) not admitting hom-universal models. Moreover, CQ-evaluation w.r.t. \mathcal{O} is in PTIME.*

PROOF. We construct an ontology \mathcal{O} in uGF(2) expressing that every constant in a unary relation $C(x)$ is the center of a ‘cartwheel’ represented by a ternary relation $W(x, y, z)$. The cartwheel can be generated using the third component of W (called ‘turning left’) or its second component (called ‘turning right’). There does not exist a hom-universal model of $\mathfrak{D}_0 = \{C(a)\}$ and \mathcal{O} as no model of \mathfrak{D}_0 and \mathcal{O} can be homomorphically mapped into the two resulting models but one can ensure that \mathcal{O} is materializable. As a first attempt to construct \mathcal{O} take unary relation symbols L (turn left) and R (turn right) and state that one can choose either L or R when generating the cartwheel with center C :

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

The following sentences then generate the wheel accordingly:

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

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

The instance \mathfrak{D}_0 shows that this ontology does not admit hom-universal models. It is, however, also not materializable since $\mathcal{O}, \mathfrak{D}_0 \models L(a) \vee R(a)$ but neither $\mathcal{O}, \mathfrak{D}_0 \models L(a)$ nor $\mathcal{O}, \mathfrak{D}_0 \models$

$R(a)$. 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 axioms 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 a with $C(a) \in \mathfrak{D}$. The resulting ontology is still not materializable: if $W(a, b, c) \in \mathfrak{D}$ then CQs can detect whether one introduces a constant c' with $W(a, c, c')$ or a constant b' with $W(a, b', b)$ when building a model of \mathcal{O} and \mathfrak{D} . To deal with this problem we ensure that a cartwheel has to be generated from atoms $W(a, b, c)$ *only if* $b, c \notin \mathfrak{D}$. In detail, the construction of \mathcal{O} is as follows. Let A , L , and R be unary relation symbols and aux and gen be binary relation symbols. First, \mathcal{O} states that every node has aux -successors in A , and gen -successors in L and R :

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

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

$$\forall x(C(x) \rightarrow (\exists y(\text{gen}(x, y) \wedge \neg L(x)) \vee \exists y(\text{gen}(x, y) \wedge \neg R(x))))$$

Now we use the following complex W' rather than W to represent the cartwheel:

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

Then for any instance \mathfrak{D} and $b \in \text{dom}(\mathfrak{D})$ one can construct a model \mathfrak{A} of \mathfrak{D} and our ontology such that $\mathfrak{A} \not\models W'(a, b, c)$ for any a, c by adding $\text{aux}(b, d)$ to \mathfrak{A} for a fresh constant d with $A(d) \notin \mathfrak{A}$, and similarly for the third component of W' . The following axiom starts the generation of the cartwheel.

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

Finally, we turn either left or right:

$$\forall x, y, z(W'(x, y, z) \rightarrow (\exists y(\text{gen}(x, y) \wedge \neg L(y)) \rightarrow \exists z' W'(x, z, z')))$$

$$\forall x, y, z(W'(x, y, z) \rightarrow (\exists y(\text{gen}(x, y) \wedge \neg R(y)) \rightarrow \exists y' W'(x, y', y)))$$

This finishes the definition of \mathcal{O} . \mathcal{O} is a uGF(2) ontology. One can now easily construct for any instance \mathfrak{D} a materialization of \mathfrak{D} and \mathcal{O} that shows that CQ evaluation w.r.t. \mathcal{O} is in PTIME. On the other hand, for $\mathfrak{D}_0 = \{C(a)\}$ there does not exist a hom-universal model of \mathcal{O} and \mathfrak{D}_0 . \square

Theorem 3.7 (restated) *For all uGF(=) 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[≠]-rewritable’ and with ‘CONP-hard’ (and with ‘Datalog-rewritable’ if \mathcal{O} is a uGF ontology).

PROOF. We first deal with PTIME membership. In this case, it suffices to prove the implication of (3) by (1). By Theorem 3.6, we may assume that \mathcal{O} is materializable. To prove that UCQ-evaluation w.r.t. \mathcal{O} is in PTIME, we exploit materializability to reduce UCQ-evaluation w.r.t. \mathcal{O} to rAQ-evaluation w.r.t. \mathcal{O} . The following claim formally states the properties of this reduction.

CLAIM 1. *Let $q(\vec{x})$ be a UCQ. Then there exists a finite set \mathcal{D} of pairs $(\phi(\vec{x}, \vec{y}), \mathcal{P})$, where*

- (1) $\phi(\vec{x}, \vec{y})$ is a conjunction of atomic formulas (possibly equality atoms) that contains all the variables from \vec{x} ;
- (2) \mathcal{P} is a finite set of rAQs with free variables in \vec{x} and \vec{y} ;

In addition, for each instance \mathfrak{D} and each tuple \vec{a} in \mathfrak{D} , we have $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$ iff there exists a pair $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ in \mathcal{D} and an assignment π of constants in $\text{dom}(\mathfrak{D})$ to the variables in ϕ such that

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

Using Claim 1 it is easy to complete the proof of (3). To show that evaluating a UCQ $q(\vec{x})$ w.r.t. \mathcal{O} is in PTIME, we first fix a set \mathcal{D} as provided by the claim. On input of an instance \mathfrak{D} and a tuple \vec{a} in \mathfrak{D} , we then check if there exists a pair $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ in \mathcal{D} and an assignment π of constants in $\text{dom}(\mathfrak{D})$ to the variables in ϕ such that Conditions 3–5 are true, where for Condition 5 we exploit that rAQ-evaluation is in PTIME.

We note that similar reductions of UCQs to certain forms of acyclic CQs have been used in [9, 19], but the construction of the set \mathcal{D} in the above claim is more subtle due to the requirement that the queries that occur in the set \mathcal{P} of a pair in \mathcal{D} have to be rAQs. In particular, each query that occurs in \mathcal{P} has at least one answer variable. Before we prove the claim, we introduce a tool that helps us to achieve the latter property.

One of the key steps in the proof of Claim 1 is to express a Boolean CQ $q() \leftarrow \psi$ whose body ψ is the body of a rAQ by another rAQ. The main issue is that a homomorphism that maps the canonical database \mathfrak{D}_q of such a CQ into a model \mathfrak{A} of \mathfrak{D} and \mathcal{O} is able to map the atomic formulas of ψ to facts in \mathfrak{A} that are arbitrarily ‘far away’ from the facts in \mathfrak{D} , making them inaccessible to any fixed rAQ. Here, the distance between facts in \mathfrak{A} is defined as the distance between their corresponding guarded sets. Given two guarded sets G, G' in \mathfrak{A} , the *distance* between G and G' in \mathfrak{A} , denoted by $\text{dist}_{\mathfrak{A}}(G, G')$, is the length of a shortest sequence G_1, \dots, G_d of guarded sets in \mathfrak{A} such that $G_1 = G$, $G_d = G'$, and $G_i \cap G_{i+1} \neq \emptyset$ for all $i \in \{1, \dots, d-1\}$. For a sub-interpretation \mathfrak{B} of \mathfrak{A} , we define $\text{dist}_{\mathfrak{A}}(\mathfrak{B}, G')$ as the minimum of $\text{dist}_{\mathfrak{A}}(G, G')$, where G ranges over all guarded sets in \mathfrak{B} . Now, the following Claim 2 resolves the above problem by showing that if $\mathfrak{A} \models q$, then there is a homomorphism from \mathfrak{D}_q to \mathfrak{A} that maps all atomic formulas in ψ to facts in \mathfrak{A} with bounded distance to the facts in \mathfrak{D} .

CLAIM 2. *For every rAQ $q(\vec{x})$ there exists an integer $d_0 \geq 0$ with the following properties. If \mathfrak{A} is a rAQ-materialization of \mathcal{O} and an instance \mathfrak{D} and if $\mathfrak{A} \models \exists \vec{x} q(\vec{x})$, then there exists a tuple \vec{a} in \mathfrak{A} with $\mathfrak{A} \models q(\vec{a})$ and $\text{dist}_{\mathfrak{A}}(\mathfrak{D}, [\vec{a}]) \leq d_0$.*

PROOF. The proof uses a pumping argument, which is based on the following notion of a type. To simplify the presentation, we view q as an openGF formula. This is possible, because there exists a cg-tree decomposition of \mathfrak{D}_q in which the root’s bag contains all the answer variables of q . We define the *closure* of \mathcal{O} and q as the smallest set $\text{cl}(\mathcal{O}, q)$ satisfying:

- $\mathcal{O} \cup \{q\} \subseteq \text{cl}(\mathcal{O}, q)$;
- for each relation symbol R that occurs in \mathcal{O} or q we have that $\text{cl}(\mathcal{O}, q)$ contains an atomic formula $R(\vec{x})$, where \vec{x} is a tuple of distinct variables;
- $\text{cl}(\mathcal{O}, q)$ contains an atomic formula $x = y$, where x and y are distinct variables;
- $\text{cl}(\mathcal{O}, q)$ is closed under subformulas and single negation.

The *type* of a tuple $\vec{a} = (a_1, \dots, a_n)$ in an interpretation \mathfrak{A} w.r.t. \mathcal{O} and q is the set $\Phi(\vec{x})$ of all formulas $\phi(\vec{x})$ such that $\mathfrak{A} \models \phi(\vec{a})$ and ϕ is obtained from a formula in $\text{cl}(\mathcal{O}, q)$ by substituting variables in \vec{x} . Here, \vec{x} is an arbitrary tuple of distinct variables, called the *free variables* of the type. A *type* w.r.t. \mathcal{O} and q is the type of a tuple in some interpretation w.r.t. \mathcal{O} and q . In the following, we will not explicitly mention \mathcal{O} and q if these are understood. Types $\Phi(\vec{x})$ and $\Psi(\vec{y})$ are *equivalent*, denoted by $\Phi(\vec{x}) \equiv \Psi(\vec{y})$, if there is a bijective mapping f on the variables in \vec{x} such that $\Psi(\vec{y})$ can be obtained from $\Phi(\vec{x})$ by consistently renaming each free variable x to $f(x)$. Since \mathcal{O} and q are fixed, the set of all non-equivalent types with a fixed number of free variables can be computed in constant time using any satisfiability procedure for the guarded fragment [41].

Let w be the maximum arity of a relation symbol that occurs in $\text{cl}(\mathcal{O}, q)$. Since $\text{cl}(\mathcal{O}, q)$ is finite, the number of non-equivalent types with at most w free variables is finite. Let τ be this number, and define $d_0 := \tau^2$.

We are now ready to prove the claim. Without loss of generality, we may assume that \mathfrak{A} is a forest model of \mathfrak{D} and \mathcal{O} obtained using uGF-unravelings, as described in the proof of Lemma 2.12. Let d be the smallest integer with $H_d(\mathfrak{A}) \neq \emptyset$, where for each integer $c \geq 0$ and each model \mathfrak{B} we let $H_c(\mathfrak{B})$ be the set of all homomorphisms h from \mathfrak{D}_q to \mathfrak{B} with $\text{dist}_{\mathfrak{B}}(\mathfrak{D}, [h(\vec{x})]) \leq c$. For a contradiction, suppose that $d > d_0$.

Consider any $h \in H_d(\mathfrak{A})$. Since \mathfrak{A} is a forest model of \mathfrak{D} and \mathcal{O} , it has the form $\mathfrak{D} \cup \bigcup_{G \in \mathcal{G}} \mathfrak{A}_G$, where \mathcal{G} is the set of all maximally guarded sets in \mathfrak{D} and the interpretations \mathfrak{A}_G are cg-tree decomposable. Since $d > d_0$, there exists a unique $G \in \mathcal{G}$ such that $[h(\vec{x})] \subseteq \text{dom}(\mathfrak{A}_G)$. Fix such a G , and let (T, E, bag) be a cg-tree decomposition of \mathfrak{A}_G with root r and $\text{bag}(r) = G$. Let t_1, t_2, \dots, t_m be the shortest path in (T, E) from $t_1 = r$ to a node t_m with $[h(\vec{x})] \subseteq \text{bag}(t_m)$. It is straightforward to verify that there is a rAQ of the form

$$\tilde{q}(\vec{y}_1) \leftarrow R_1(\vec{y}_1) \wedge R_2(\vec{y}_2) \wedge \dots \wedge R_m(\vec{y}_m) \wedge q(\vec{x})$$

such that $\mathfrak{A} \models \tilde{q}(\vec{a})$ for some tuple \vec{a} in \mathfrak{D} . Intuitively, $\tilde{q}(\vec{a})$ selects an atom $R_i(\vec{a}_i)$ with $[\vec{a}_i] = \text{bag}(t_i)$ from the bag of each node along the path t_1, t_2, \dots, t_m and then checks if $q(h(\vec{x}))$ is true. Note that, as a consequence of $\mathfrak{A} \models \tilde{q}(\vec{a})$, we have $\mathcal{O}, \mathfrak{D} \models \tilde{q}(\vec{a})$. We now use a pumping argument to construct a model of \mathfrak{D} and \mathcal{O} in which $\tilde{q}(\vec{a})$ is not true, which yields the desired contradiction.

For each $i \in \{1, 2, \dots, m\}$, we let \vec{a}_i be a guarded tuple in \mathfrak{A} with $[\vec{a}_i] = \text{bag}(t_i)$. We also define sub-interpretations \mathfrak{A}_i and \mathfrak{A}_{-i} of \mathfrak{A} , where \mathfrak{A}_i is the sub-interpretation of \mathfrak{A} induced by the bags of all nodes in the subtree rooted at t_i in (T, E) , and $\mathfrak{A}_{-i} := \mathfrak{A} \setminus \mathfrak{A}_i$. Let $\Phi_i^{\text{in}}(\vec{x}_i)$ and $\Phi_i^{\text{out}}(\vec{x}_i)$ be the type of \vec{a}_i in \mathfrak{A}_i and $\mathfrak{A}_{-i} \cup \text{Bag}(t_i)$, respectively. Since $m \geq d > d_0$, there are nodes t_i and t_j with $1 \leq i < j \leq m$ such that $\Phi_i^{\text{in}}(\vec{x}_i) \equiv \Phi_j^{\text{in}}(\vec{x}_j)$ and $\Phi_i^{\text{out}}(\vec{x}_i) \equiv \Phi_j^{\text{out}}(\vec{x}_j)$. We now construct a new interpretation \mathfrak{A}' from \mathfrak{A} by replacing the sub-interpretation \mathfrak{A}_j by an isomorphic copy of \mathfrak{A}_i . More precisely, let \mathfrak{A}'_i be an isomorphic copy of \mathfrak{A}_i obtained by replacing each occurrence of a constant in \vec{a}_i by the corresponding constant in \vec{a}_j and each remaining constant by a fresh constant not contained in $\text{dom}(\mathfrak{A}_{-j})$. Then, $\mathfrak{A}' = \mathfrak{A}_{-j} \cup \mathfrak{A}'_j$. Since $\Phi_i^{\text{in}}(\vec{x}_i) \equiv \Phi_j^{\text{in}}(\vec{x}_j)$ and $\Phi_i^{\text{out}}(\vec{x}_i) \equiv \Phi_j^{\text{out}}(\vec{x}_j)$, the new interpretation \mathfrak{A}' is a model of \mathfrak{D} and \mathcal{O} . By construction, we also have $|H_d(\mathfrak{A}')| < |H_d(\mathfrak{A})|$. Repeating this procedure for \mathfrak{A}' and all subsequent models yields a model \mathfrak{A}'' of \mathfrak{D} and \mathcal{O} with $H_d(\mathfrak{A}'') = \emptyset$. In particular, $\mathfrak{A}'' \not\models \tilde{q}(\vec{a})$, which is the desired contradiction. \dashv

We are now ready to prove Claim 1.

PROOF. Let n be the maximum number of atomic formulas in any disjunct of $q(\vec{x})$, and let d_0 be the integer from Claim 2. Define \mathcal{D} to be the set of all pairs $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ that satisfy Conditions 1–2 in Claim 1 and the following additional conditions:

- (6) if $\mathcal{O}, \mathcal{D} \models \hat{q}(\vec{a})$ for $\hat{q}(\vec{x}) = \exists \vec{y}(\phi(\vec{x}, \vec{y}) \wedge \bigwedge_{q'(\vec{z}) \in \mathcal{P}} q'(\vec{z}))$, then $\mathcal{O}, \mathcal{D} \models q(\vec{a})$;
- (7) the total number of atomic formulas in ϕ and the rAQs in \mathcal{P} is at most $(2 + d_0)n$.

We show that for each instance \mathcal{D} and each tuple \vec{a} in \mathcal{D} , we have $\mathcal{O}, \mathcal{D} \models q(\vec{a})$ iff there exists a pair $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ in \mathcal{D} and an assignment π of constants in $\text{dom}(\mathcal{D})$ to the variables in ϕ such that Conditions 3–5 in Claim 1 are satisfied. The ‘if’ direction is trivial due to Condition 6 above. We now prove the ‘only if’ direction.

Recall that \mathcal{O} is materializable. Consider any rAQ-materialization \mathfrak{A} of \mathcal{D} and \mathcal{O} . By Lemma 2.12, there exists a forest model \mathfrak{B} of \mathcal{D} and \mathcal{O} and a homomorphism h from \mathfrak{B} to \mathfrak{A} that preserves $\text{dom}(\mathcal{D})$. Again, we may assume that \mathfrak{B} is obtained using uGF-unravelings, as described in the proof of Lemma 2.12. Say, $\mathfrak{B} = \mathcal{D} \cup \bigcup_{G \in \mathcal{G}} \mathfrak{B}_G$ is a forest model of \mathcal{D} defined using \mathcal{G} , where \mathcal{G} is the set of all maximal guarded subsets of \mathcal{D} . Since $\mathcal{O}, \mathcal{D} \models q(\vec{a})$, we have $\mathfrak{B} \models q(\vec{a})$, so there exists a disjunct $q'(\vec{x})$ of $q(\vec{x})$ and a homomorphism g from $\mathcal{D}_{q'}$ to \mathfrak{B} with $g(\vec{x}) = \vec{a}$. We use $q'(\vec{x})$ and the homomorphisms h and g to define the desired pair $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ in \mathcal{D} and assignment π . The idea is that atomic formulas in $q'(\vec{x})$ that are mapped by g to the sub-interpretation \mathcal{D} of \mathfrak{B} define the first component $\phi(\vec{x}, \vec{y})$ of a pair $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ in \mathcal{D} , while the remaining atomic formulas that are mapped by g to the cg-tree decomposable components \mathfrak{B}_G of \mathfrak{B} define the rAQs in \mathcal{P} .

A set $C \subseteq \mathfrak{B}$ is *connected* if for every two guarded sets G, G' in C there is a sequence G_0, G_1, \dots, G_m of guarded sets in C such that $G_0 = G$, $G_m = G'$, and $G_i \cap G_{i+1} \neq \emptyset$ for every $i < m$. Let us partition the image of $\mathcal{D}_{q'}$ under g into a minimal collection of sets $\Phi, \Psi_1, \dots, \Psi_k$ such that $\Phi \subseteq \mathcal{D}$ and for each $i \in \{1, \dots, k\}$ there exists a $G_i \in \mathcal{G}$ such that Ψ_i is a connected subset of $\mathfrak{B}_{G_i} \setminus \mathcal{D}$. Assume for the moment that each Ψ_i has a cg-tree decomposition with root r_i and $\emptyset \neq \text{dom}(\mathcal{D}) \cap \text{dom}(\Psi_i) = \text{bag}(r_i)$. Then we obtain the desired pair $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ in \mathcal{D} as follows. First, we rename each constant a in $\text{dom}(\Phi \cup \Psi_1 \cup \dots \cup \Psi_k)$ to a variable x_a with $g(x_a) = a$. Then, we define

$$\phi(\vec{x}, \vec{y}) = \bigwedge \Phi \wedge \bigwedge \{x = x' \mid x \text{ and } x' \text{ occur in } \vec{x} \text{ and } g(x) = g(x')\}$$

and

$$q_i(\vec{z}_i) \leftarrow \bigwedge \Psi_i \quad \text{for } 1 \leq i \leq k$$

where \vec{y} consists of the variables in Φ that do not occur in \vec{x} , and \vec{z}_i consists of all variables that correspond to the constants in $\text{bag}(r_i)$. The pair $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ with $\mathcal{P} = \{q_i(\vec{z}_i) \mid 1 \leq i \leq k\}$ satisfies Conditions 1 and 2 in Claim 1 and both Condition 6 and Condition 7 at the beginning of the proof. For Condition 7, note that $|\Phi| + |\Psi_1| + \dots + |\Psi_k| \leq n$. Moreover, if π is the composition of the homomorphisms g and h , then it is straightforward to check that Conditions 3–5 in Claim 1 hold (for Condition 5, note that \mathfrak{B} is a rAQ-materialization of \mathcal{D} and \mathcal{O} , which it inherits from \mathfrak{A}). Altogether, this would complete the proof.

In general, the assumption that each of the sets Ψ_i has a cg-tree decomposition whose root r_i satisfies $\emptyset \neq \text{dom}(\mathcal{D}) \cap \text{dom}(\Psi_i) = \text{bag}(r_i)$ does not hold. To fix this, we augment Φ and the sets Ψ_i as follows. First, it is easy to see that by adding to Ψ_i at most $|\Psi_i| - 1$ facts from \mathfrak{B}_{G_i} we obtain a superset $\Psi'_i \subseteq \mathfrak{B}_{G_i}$ of Ψ_i that has a cg-tree decomposition with root r'_i and $\text{dom}(\mathcal{D}) \cap \text{dom}(\Psi'_i) \subseteq \text{bag}(r'_i)$. If $\emptyset \neq \text{dom}(\mathcal{D}) \cap \text{dom}(\Psi'_i) = \text{bag}(r'_i)$, then we let $\Psi''_i := \Psi'_i$ and $r''_i := r'_i$. Otherwise, we proceed as follows to construct a superset $\Psi''_i \subseteq \mathfrak{B}_{G_i}$ of Ψ'_i that has a cg-tree decomposition whose root r''_i satisfies $\emptyset \neq \text{dom}(\mathcal{D}) \cap \text{dom}(\Psi''_i) = \text{bag}(r''_i)$:

- Case 1: $\emptyset \neq \text{dom}(\mathfrak{D}) \cap \text{dom}(\Psi'_i) \subsetneq \text{bag}(r'_i)$. In this case, Ψ''_i is obtained from Ψ'_i by adding an arbitrary fact $R(\vec{a})$ from \mathfrak{B}_{G_i} with $[\vec{a}] = G_i$. Note that in this case, G_i is the bag of the root of a cg-tree decomposition for Ψ''_i .
- Case 2: $\text{dom}(\mathfrak{D}) \cap \text{dom}(\Psi'_i) = \emptyset$. We use Ψ'_i to define a rAQ $q'_i(\vec{z}'_i)$ in the same way as we used Ψ_i to define the rAQ $q_i(\vec{z}_i)$. In particular, \vec{z}'_i corresponds to the constants in $\text{bag}(r'_i)$. By construction of Ψ'_i we have $\mathfrak{B} \models \exists \vec{z}'_i q'_i(\vec{z}'_i)$, so by Claim 2 there exists a tuple \vec{a} in \mathfrak{B} with $\mathfrak{B} \models q'_i(\vec{a})$ and $\text{dist}_{\mathfrak{B}}(\mathfrak{D}, [\vec{a}]) \leq d_0$. We may therefore assume without loss of generality that $\text{dist}_{\mathfrak{B}}(\mathfrak{D}, \text{bag}(r'_i)) \leq d_0$. Let $R_1(\vec{a}_1), \dots, R_m(\vec{a}_m)$ be a shortest sequence of facts in \mathfrak{B} such that $[\vec{a}_1] \subseteq \text{dom}(\mathfrak{D})$, $[\vec{a}_m] = \text{bag}(r'_i)$, and $[\vec{a}_j] \cap [\vec{a}_{j+1}] \neq \emptyset$ for $1 \leq j < m$. Then, $m \leq d_0$ and we define $\Psi''_i := \Psi'_i \cup \{R_j(\vec{a}_j) \mid 1 \leq j \leq m\}$. Note that $[\vec{a}_1]$ is the bag of the root of a cg-tree decomposition of Ψ''_i .

Finally, let $\Phi'' := \Phi \cup \{R_i(\vec{a}_i) \mid 1 \leq i \leq k\}$, where $R_i(\vec{a}_i)$ is a fact in \mathfrak{D} such that $[\vec{a}_i] = G_i$. Note that

$$|\Phi''| + \sum_{i=1}^k |\Psi''_k| \leq |\Phi| + k + \sum_{i=1}^k (2|\Psi_i| - 1 + d_0) \leq 2(|\Phi| + \sum_{i=1}^k |\Psi_i|) + d_0 k \leq (2 + d_0)n.$$

We now construct the queries $\phi(\vec{x}, \vec{y})$ and $q_i(\vec{z}_i)$ as above, except that we substitute $\Phi'', \Psi''_1, \dots, \Psi''_k$ for $\Phi, \Psi_1, \dots, \Psi_k$. Then, the pair $(\phi(\vec{x}, \vec{y}), \mathcal{P})$ with $\mathcal{P} = \{q_i(\vec{z}_i) \mid 1 \leq i \leq k\}$ satisfies Conditions 1 and 2 in Claim 1 and Condition 6 and 7 at the beginning of the proof, and it is easy to see that there is an extension of the mapping π such that Conditions 3–5 in Claim 1 hold. \dashv

Datalog $^{\neq}$ -rewritability can be handled similarly. Again, it suffices to prove the implication of (3) by (1) and we may assume that \mathcal{O} is materializable. We construct a Datalog $^{\neq}$ -program for evaluating $q(\vec{x})$ w.r.t. \mathcal{O} as follows. Fix a set \mathcal{D} as provided by Claim 1, and let \mathcal{Q} be the set of all rAQs that occur in a pair in \mathcal{D} . 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 relational symbols used in different programs $\Pi_{q'}$ and $\Pi_{q''}$ are disjoint, and that the goal predicate of $\Pi_{q'}$ is $\text{goal}_{q'}$. 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 $(\phi(\vec{x}, \vec{y}), \mathcal{P}) \in \mathcal{D}$:

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

Note that if each $\Pi_{q'}$ is a Datalog program, then Π is a Datalog program. Then, for all instances \mathfrak{D} and all tuples \vec{a} in \mathfrak{D} , we have $\mathfrak{D} \models \Pi(\vec{a})$ iff $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$.

Finally, we deal with CONP-hardness. In this case, it suffices to prove the implication of (1) by (3). We prove the stronger statement that if UCQ-evaluation w.r.t. \mathcal{O} is CONP-hard, then unary rAQ-evaluation w.r.t. \mathcal{O} is CONP-hard. If \mathcal{O} is not materializable, then by Theorem 3.6 we have that unary rAQ-evaluation w.r.t. \mathcal{O} is CONP-hard and we are done. In the following, we may therefore assume that \mathcal{O} is materializable. Let $q(\vec{x})$ be a UCQ that witnesses CONP-hardness of UCQ-evaluation w.r.t. \mathcal{O} , and fix a set \mathcal{D} as in Claim 1. Let \mathcal{Q} be the set of all rAQs that occur in some pair in \mathcal{D} . Using \mathcal{Q} we construct a unary rAQ $\tilde{q}(x)$ such that evaluating $q(\vec{x})$ w.r.t. \mathcal{O} is polynomially reducible to evaluating $\tilde{q}(x)$ w.r.t. \mathcal{O} .

Let $q'_1(\vec{z}_1), \dots, q'_m(\vec{z}_m)$ be an enumeration of the rAQs in \mathcal{Q} , and let k_i be the length of \vec{z}_i , for each $i \in \{1, \dots, m\}$. 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. Now, given an instance \mathfrak{D} and a tuple \vec{a} in \mathfrak{D} , we

construct a new instance $\tilde{\mathcal{D}}$ as follows. We start with the disjoint union of $\mathcal{D}, \mathcal{D}_{q'_1}, \dots, \mathcal{D}_{q'_m}$. Let \vec{c}_i be the tuple of elements in the copy of $\mathcal{D}_{q'_i}$ that represents the tuple \vec{z}_i . Next, we add the following facts for each pair $\delta = (\phi(\vec{x}, \vec{y}), \mathcal{P})$ in \mathcal{D} and each assignment π of elements in $\text{dom}(\mathcal{D})$ to the variables in ϕ that satisfies $\pi(\vec{x}) = \vec{a}$ and $\mathcal{D} \models \phi(\pi(\vec{x}, \vec{y}))$:

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

Here, the a_0, a_δ , and $a_{\delta, \pi}$ are constants that do not occur in $\text{dom}(\mathcal{D})$. Since \mathcal{D} is of constant size, we can compute $\tilde{\mathcal{D}}$ in time polynomial in the size of \mathcal{D} . It is now straightforward to verify that $\mathcal{O}, \mathcal{D} \models q(\vec{a})$ holds iff $\mathcal{O}, \tilde{\mathcal{D}} \models \tilde{q}(a_0)$, where $\tilde{q}(x)$ is the rAQ

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

Note that the mapping $\mathcal{D} \mapsto \tilde{\mathcal{D}}$ is a polynomial-time reduction from evaluating $q(\vec{x})$ w.r.t. \mathcal{O} to evaluating $\tilde{q}(x)$ w.r.t. \mathcal{O} . Since the former problem is CONP-hard, we conclude that evaluating $\tilde{q}(x)$ w.r.t. \mathcal{O} is CONP-hard. \square

Theorem 3.8 (restated) *For all $uGC_2(=)$ ontologies \mathcal{O} , the statements (1) to (3) of Theorem 3.7 are equivalent and also equivalent to*

- (4) *unary rAQ-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’.

PROOF. The proof is almost the same as the proof of Theorem 3.8. Instead of using forest models constructed via uGF-unravelings one uses forest models constructed via uGC_2 -unravelings from the proof of Lemma 2.12. To deal with Condition (4), in Claim 1 one has to add the condition that every rAQ that occurs in \mathcal{P} is unary which is easily achieved using the assumption that in $uGC_2(=)$ all relation symbols are at most binary. \square

C PROOFS FOR SECTION 4

Lemma 4.3 (restated) *The implication (2) \Rightarrow (1) in Definition 4.2 holds for every $uGF(=)$ and $uGC_2(=)$ ontology and every rAQ.*

PROOF. Observe that the mapping $h : a \mapsto a^\uparrow$ is a homomorphism from \mathcal{D}^u to \mathcal{D} (Lemmas 2.10 and 2.11). Thus, if \mathcal{O} does not use equality or counting quantifiers, then the implication (2) \Rightarrow (1) follows from the fact that certain answers are preserved under homomorphisms between instances, for any ontology given in FO without equality [15] (Proposition 5.9). In general, this is not the case, and a different argument is required. Assume \mathfrak{A} is a model of \mathcal{D} and \mathcal{O} such that $\mathfrak{A} \not\models q(\vec{a})$, where q is an rAQ. We may assume that \mathfrak{A} is a forest model (Lemma 2.12). We construct from \mathfrak{A} a model \mathfrak{A}^u of \mathcal{D}^u and \mathcal{O} such that $(\mathfrak{A}^u, \vec{b})$ and (\mathfrak{A}, \vec{a}) are connected guarded bisimilar. It then follows that $\mathfrak{A}^u \not\models q(\vec{b})$ since q can be regarded as an openGF formula. To construct \mathfrak{A}^u in the $uGF(=)$ case, hook to \mathcal{D}^u at every maximally guarded set G a copy of the cg-tree decomposable interpretation $\mathfrak{A}_{G^\uparrow}$ hooked to \mathcal{D} at $G^\uparrow = \{a^\uparrow \mid a \in G\}$ in the construction of the forest model \mathfrak{A} by identifying every $a \in G$ with $a^\uparrow \in G^\uparrow$. It is straightforward to construct the required connected guarded bisimulation between $(\mathfrak{A}^u, \vec{b})$ and (\mathfrak{A}, \vec{a}) by taking the union of the guarded bisimulation between \mathcal{D}^u and \mathcal{D} and the obvious isomorphisms between the copies of $\mathfrak{A}_{G^\uparrow}$ hooked to G

and the original $\mathfrak{A}_{G^\uparrow}$. This connected guarded bisimulation also shows that \mathfrak{A}^u is a model of \mathcal{O} (it is a model of \mathfrak{D}^u by definition), see Lemma 2.6. The argument for uGC_2 is similar and left to the reader. \square

Theorem 4.5 (restated) *If \mathcal{O} is an unraveling tolerant $\text{uGF}(=)$ or $\text{uGC}_2(=)$ ontology, then rAQ -evaluation w.r.t. \mathcal{O} is Datalog^\neq -rewritable (resp., Datalog -rewritable if \mathcal{O} is formulated in uGF).*

PROOF. Assume first that \mathcal{O} is an unraveling tolerant $\text{uGF}(=)$ ontology, and that $q(\vec{x})$ is a rAQ . We discuss how we can decide $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$ for a given instance \mathfrak{D} and tuple \vec{a} in \mathfrak{D} , and then construct the desired Datalog^\neq -rewriting. Without loss of generality, we may assume that $[\vec{a}] = G$ is maximally guarded in \mathfrak{D} , so by unraveling tolerance it suffices to decide $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$, where \mathfrak{D}^u is the global uGF -unraveling of \mathfrak{D} and \vec{b} is the copy of \vec{a} in $\text{bag}(G)$. We use the notion of a *type* from the proof of Theorem 3.7 (see the beginning of the proof of Claim 2 on p. 53). The key is to determine if we can label each maximally guarded tuple \vec{c} in \mathfrak{D}^u with the type $\Phi_{\vec{c}}(\vec{x}_{\vec{c}})$ of \vec{c} in a model $\mathfrak{A}_{\vec{c}}$ of $\mathfrak{D}^u|_{[\vec{c}]}$ and \mathcal{O} such that $q(\vec{x}_{\vec{b}}) \notin \Phi_{\vec{b}}(\vec{x}_{\vec{b}})$. Clearly, this is possible if there exists a model \mathfrak{A} of \mathfrak{D}^u and \mathcal{O} that falsifies $q(\vec{b})$, since we can use \mathfrak{A} as the label of each maximally guarded tuple in \mathfrak{D}^u . Conversely, by imposing a certain consistency condition on the types of intersecting maximally guarded tuples, we can ensure that the interpretation obtained from \mathfrak{D}^u by hooking each model $\mathfrak{A}_{\vec{c}}$ to \mathfrak{D}^u at $[\vec{c}]$ is a model of \mathfrak{D}^u and \mathcal{O} that falsifies $q(\vec{b})$.

To decide the existence of a labeling with the above properties, we use the notion of a *type assignment*, which is a mapping T that assigns to each maximally guarded tuple $\vec{c} = (c_1, \dots, c_n)$ in \mathfrak{D} a non-empty set $T(\vec{c})$ of types $\Phi(x_1, \dots, x_n)$ such that:

- (1) each type in $T(\vec{c})$ is the type of \vec{c} in some model of $\mathfrak{D}|_{[\vec{c}]}$ and \mathcal{O} ;
- (2) for each maximally guarded tuple \vec{c} in \mathfrak{D} with $[\vec{c}] \cap [\vec{c}'] \neq \emptyset$ there exists a type $\Phi'(\vec{x}')$ in $T(\vec{c}')$ such that $\Phi(\vec{c})$ and $\Phi'(\vec{c}')$ are consistent.

Here, given a type $\Phi(x_1, \dots, x_n)$ and constants c_1, \dots, c_n , we denote by $\Phi(c_1, \dots, c_n)$ the set of all formulas that are obtained from a formula in $\Phi(x_1, \dots, x_n)$ by replacing each free occurrence of x_i by c_i , for each $i \in \{1, \dots, n\}$, where we regard c_1, \dots, c_n as the free variables of $\Phi(c_1, \dots, c_n)$. Moreover, types $\Phi(\vec{x})$ and $\Psi(\vec{y})$ are *consistent* if they agree on all formulas that contain only the variables in $[\vec{x}] \cap [\vec{y}]$. It turns out that a type assignment T with the property that $q(\vec{x}) \notin \Phi(\vec{x})$ for some type $\Phi(\vec{x}) \in T(\vec{a})$ yields the desired labeling of the maximally guarded tuples in \mathfrak{D}^u , and thus leads to a decision procedure for $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$.

CLAIM 1. $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$ iff there exists a type assignment T and a type $\Phi(\vec{x}) \in T(\vec{a})$ with $q(\vec{x}) \notin \Phi(\vec{x})$.

PROOF. For the ‘only if’ direction, assume $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$. It will be more convenient to work with the equivalent assumption that $\mathcal{O}, \mathfrak{D} \not\models q(\vec{a})$, so let \mathfrak{A} be a model of \mathfrak{D} and \mathcal{O} with $\mathfrak{A} \not\models q(\vec{a})$. For each maximally guarded tuple $\vec{c} = (c_1, \dots, c_n)$ of \mathfrak{D} , let $\Phi_{\vec{c}}(x_1, \dots, x_n)$ be the type of \vec{c} in \mathfrak{A} , and set $T(\vec{c}) = \{\Phi_{\vec{c}}(x_1, \dots, x_n)\}$. Then, T is a type assignment. Furthermore, $q(x_1, \dots, x_{|\vec{a}|})$ does not occur in the type $\Phi_{\vec{a}}(x_1, \dots, x_{|\vec{a}|})$ assigned to \vec{a} .

For the ‘if’ direction, let T be a type assignment such that $q(\vec{x}) \notin \Phi(\vec{x})$ for some type $\Phi(\vec{x}) \in T(\vec{a})$. We construct a model \mathfrak{A} of \mathfrak{D}^u and \mathcal{O} with $\mathfrak{A} \not\models q(\vec{b})$.

We first assign to each maximally guarded tuple \vec{c} in \mathfrak{D}^u a type in $T(\vec{c}^\uparrow)$. It suffices to do this for the maximally guarded tuples corresponding to the bags in the cg -tree decomposition $(T(\mathfrak{D}, G'), E, \text{bag})$, for each maximally guarded set G' in \mathfrak{D} . Fix a maximally guarded set G'

in \mathfrak{D} . For each node t in $T(\mathfrak{D}, G')$, let \vec{c}_t be a tuple consisting of all constants in $\text{bag}(t)$ such that $\vec{c}_t = \vec{b}$ if t is the root $G = [\vec{a}]$ of $(T(\mathfrak{D}, G), E, \text{bag})$. We inductively assign to each node t in $T(\mathfrak{D}, G')$ a type $\Phi_t(\vec{x}_t) \in T(\vec{c}_t^\uparrow)$. If $t = G'$, then we let $\Phi_t(\vec{x}_t)$ be any type in $T(\vec{c}_t^\uparrow)$. If in addition we have $G' = G$, then $\vec{c}_t^\uparrow = \vec{a}$ and we select $\Phi_t(\vec{x}_t)$ so that $q(\vec{x}_t) \notin \Phi_t(\vec{x}_t)$. For the induction step, consider a node in $T(\mathfrak{D}, G')$ of the form $t' = tG''$. Since $\Phi_t(\vec{x}_t) \in T(\vec{c}_t^\uparrow)$ and $[\vec{c}_t] \cap [\vec{c}_{t'}] \neq \emptyset$, there exists a type $\Phi_{t'}(\vec{x}_{t'}) \in T(\vec{c}_{t'}^\uparrow)$ such that $\Phi_t(\vec{c}_t)$ and $\Phi_{t'}(\vec{c}_{t'})$ are compatible. We assign this type to t' .

To obtain the desired model of \mathfrak{D}^u and \mathcal{O} , we proceed as follows. For each $t \in T(\mathfrak{D})$, we pick a model \mathfrak{A}_t of $\mathfrak{D}^u_{[\vec{c}_t]}$ and \mathcal{O} such that $\Phi_t(\vec{x}_t)$ is the type of \vec{c}_t in \mathfrak{A}_t . Without loss of generality, we may assume that \mathfrak{A}_t has a cg-tree decomposition such that the root's bag is exactly $[\vec{c}_t]$. Furthermore, we may assume that $\text{dom}(\mathfrak{A}_t) \cap \text{dom}(\mathfrak{D}^u) = [\vec{c}_t]$ for each node t and $\text{dom}(\mathfrak{A}_t) \cap \text{dom}(\mathfrak{A}_{t'}) = [\vec{c}_t] \cap [\vec{c}_{t'}]$ for every two distinct nodes t, t' . Let \mathfrak{A} be the interpretation obtained from \mathfrak{D}^u by hooking \mathfrak{A}_t to \mathfrak{D}^u for each node t :

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

It can be shown that \mathfrak{A} is a model of \mathfrak{D}^u and \mathcal{O} with $\mathfrak{A} \not\models q(\vec{b})$. To prove that \mathfrak{A} is a model of \mathcal{O} and that $\mathfrak{A} \not\models q(\vec{b})$, we can use that for all openGF formulas $\phi(\vec{x})$, for all guarded tuples \vec{c} of \mathfrak{A} , and for all $t \in T(\mathfrak{D})$ with $[\vec{c}] \subseteq \text{dom}(\mathfrak{A}_t)$ we have $\mathfrak{A} \models \phi(\vec{c})$ iff $\mathfrak{A}_t \models \phi(\vec{c})$. The proof is straightforward by induction on the structure of ϕ . \square

To conclude the proof for the case of uGF(=) ontologies, let us show how the condition in Claim 1 can be verified by a Datalog[≠] program $\Pi_{\mathcal{O}, q}$. The idea is to derive the desired type assignment inductively, starting with a set of all possible types for each guarded tuple in \mathfrak{D} , and removing a type $\Phi(\vec{x})$ from the set of a guarded tuple \vec{c} whenever there exists a guarded tuple \vec{c}' with $[\vec{c}] \cap [\vec{c}'] \neq \emptyset$ such that $\Phi(\vec{c})$ is not consistent with $\Phi'(\vec{c}')$ for any type $\Phi'(\vec{x}')$ in the set for \vec{c}' . In the Datalog[≠] program, we use relation symbols P_T to assign sets T of types to each guarded tuple in \mathfrak{D} . The program will assign many such sets to each guarded tuple \vec{c} , but there will be an inclusion-minimal one, which we pick as the set assigned to \vec{c} in a type assignment.

For the formal description of the program, let w be the maximum arity of a relation symbol in \mathcal{O} or q , and let k be the number of answer variables of $q(\vec{x})$. Fix $2w$ variables z_1, \dots, z_{2w} . In the description below, \vec{u}, \vec{v} range over tuples consisting of at most w of these variables, and \vec{w} ranges over k -tuples of variables in $\{z_1, \dots, z_{2w}\}$. The rules of $\Pi_{\mathcal{O}, q}$ are as follows:

- (1) $P_T(z_i) \leftarrow \alpha$, where α is an atomic formula that involves only the variable z_i , $i \in \{1, \dots, 2w\}$, and T consists of all types with free variable z_i that contain α ;
- (2) $P_T(\vec{u}) \leftarrow R(\vec{u}) \wedge \alpha$, where $R \in \text{sig}(\mathcal{O} \cup \{q\})$, α is an atomic formula (possibly an equality) that involves only variables from \vec{u} , and T consists of all types with free variables \vec{u} that contain both $R(\vec{u})$ and α ;
- (3) $P_{T \rightsquigarrow U}(\vec{u}) \leftarrow P_T(\vec{u}) \wedge P_U(\vec{v})$, where T and U are sets of types with free variables \vec{u} and \vec{v} , respectively, \vec{u} and \vec{v} share at least one variable, and $T \rightsquigarrow U$ denotes the set of all types in T that are consistent with some type in U ;
- (4) $P_{T \cap T'}(\vec{u}) \leftarrow P_T(\vec{u}) \wedge P_{T'}(\vec{u})$, where T, T' are sets of types with free variables \vec{u} ;
- (5) $\text{goal}(\vec{w}) \leftarrow P_T(\vec{u})$, where T is a set of types with free variables \vec{u} such that $q(\vec{w})$ is contained in all types in T ;
- (6) $\text{goal}(\vec{w}) \leftarrow P_\emptyset(\vec{u})$, where \vec{u} and \vec{w} do not share any variables.

If equality occurs at a non-guarded position in \mathcal{O} (i.e., if \mathcal{O} is not formulated in uGF), then the program also contains the following rule:

- (7) $P_T(\vec{u}) \leftarrow z_i \neq z_j$, where z_i and z_j are distinct variables in \vec{u} , and T consists of all types with free variables \vec{u} that contain $\neg(z_i = z_j)$.

Note that the above program is technically not a Datalog[≠] program, since the bodies of some rules may contain equality atoms. However, it is not difficult to see that equality atoms can be eliminated by introducing additional Datalog rules that define the equality predicate.

Using Claim 1, we can now show:

CLAIM 2. $\mathfrak{D} \models \Pi_{\mathcal{O},q}(\vec{a})$ iff $\mathcal{O}, \mathfrak{D}^u \models q(\vec{a})$.

PROOF. By Claim 1, it suffices to show that $\mathfrak{D} \not\models \Pi_{\mathcal{O},q}(\vec{a})$ iff there exists a type assignment T and a type $\Phi(\vec{x}) \in T(\vec{a})$ with $q(\vec{x}) \notin \Phi(\vec{x})$. First assume that $\mathfrak{D} \not\models \Pi_{\mathcal{O},q}(\vec{a})$. Let \mathfrak{A} be the unique minimal model of \mathfrak{D} and $\Pi_{\mathcal{O},q}$. For each guarded tuple \vec{c} in \mathfrak{D} , let $T(\vec{c})$ be the unique inclusion-minimal set of types with free variables $z_1, \dots, z_{|\vec{c}|}$ such that $P_{T(\vec{c})}(\vec{c}) \in \mathfrak{A}$. Then, $T(\vec{c})$ is non-empty. Since $\text{goal}(\vec{a}) \notin \mathfrak{A}$, there exists a type in $T(\vec{a})$ that does not contain $q(z_1, \dots, z_k)$. It follows that T leads to the desired type assignment.

For the ‘if’ direction, let T be a type assignment such that $q(\vec{x}) \notin \Phi(\vec{x})$ for some type $\Phi(\vec{x}) \in T(\vec{a})$. Without loss of generality, we may assume that T is maximal in the sense that for every type assignment T' we have $T'(\vec{c}) \subseteq T(\vec{c})$. We rename each variable x_i that occurs free in a type assigned by T to z_i , and extend T so that it assigns to each guarded tuple \vec{c} that is properly contained in a maximally guarded tuple \vec{c}' the set of all types in $T(\vec{c}')$ restricted to all formulas whose free variables correspond to the constants in \vec{c} . Now, rules (1)–(4) and (7) of $\Pi_{\mathcal{O},q}$ ensure that for each guarded tuple \vec{c} in \mathfrak{D} the unique minimal model \mathfrak{A} of \mathfrak{D} and $\Pi_{\mathcal{O},q}$ contains the fact $P_{T(\vec{c})}(\vec{c})$. By the choice of T , it follows that $\text{goal}(\vec{a}) \notin \mathfrak{A}$, and consequently $\mathfrak{D} \not\models \Pi_{\mathcal{O},q}(\vec{a})$. \square

Altogether, this concludes the proof for the case that \mathcal{O} is a uGF(=) ontology.

The case of ontologies \mathcal{O} formulated in uGC₂(=) is similar, but requires a bit more care. In this case, we assume that all relation symbols are at most binary, and \mathfrak{D}^u is the global uGC₂-unraveling of \mathfrak{D} . As in the case of uGF(=) ontologies, the key is to determine if we can label the maximally guarded tuples \vec{c} of \mathfrak{D}^u with the type $\Phi_{\vec{c}}(\vec{x})$ of \vec{c} in a model of $\mathfrak{D}^u_{|\vec{c}|}$ and \mathcal{O} such that $q(\vec{x}_{\vec{c}}) \notin \Phi_{\vec{c}}(\vec{x}_{\vec{c}})$. However, to ensure that we can hook appropriate interpretations to \mathfrak{D}^u in order to obtain a model of \mathfrak{D}^u and \mathcal{O} that falsifies $q(\vec{b})$ we need a stronger consistency condition. The reason is that uGC₂(=) allows us to count. In particular, in uGC₂(=) we can count the number of guarded tuples that intersect with a given guarded tuple \vec{c} and satisfy a certain property. It is therefore no longer sufficient to ensure consistency between pairwise intersecting maximally guarded tuples, but for a given maximally guarded tuple \vec{c} we need to take into account all maximally guarded tuples that intersect with \vec{c} .

We generalize the definition of a type assignment as follows. A *type assignment* is a mapping T that assigns to each maximally guarded tuple (c_1, c_2) in \mathfrak{D} a non-empty set $T(c_1, c_2)$ of types $\Phi(x_1, x_2)$ for which there exists a model \mathfrak{A} of \mathcal{O} with the following properties:

- $\mathfrak{D}_{\{c_1, c_2\}} \subseteq \mathfrak{A}$ and $\Phi(x_1, x_2)$ is the type of (c_1, c_2) in \mathfrak{A} ;
- for each maximally guarded tuple (c'_1, c'_2) in \mathfrak{D} with $\{c_1, c_2\} \cap \{c'_1, c'_2\} \neq \emptyset$ we have $\mathfrak{D}_{\{c'_1, c'_2\}} \subseteq \mathfrak{A}$ and there exists a type $\Phi'(x_1, x_2)$ in $T(c'_1, c'_2)$ such that $\Phi'(x_1, x_2)$ is the type of (c'_1, c'_2) in \mathfrak{A} .

As in the case of uGF(=) ontologies, we can now show the following:

CLAIM 3. $\mathcal{O}, \mathfrak{D}^u \not\models q(\vec{b})$ iff there exists a type assignment T and a type $\Phi(\vec{x}) \in T(\vec{a})$ with $q(\vec{x}) \notin \Phi(\vec{x})$.

PROOF. The ‘only if’ direction is exactly as in Claim 1. For the ‘if’ direction, let T be a type assignment such that $q(\vec{x}) \notin \Phi(\vec{x})$ for some type $\Phi(\vec{x}) \in T(\vec{a})$. We are going to construct a model \mathfrak{A} of \mathfrak{D}^u and \mathcal{O} with $\mathfrak{A} \not\models q(\vec{b})$.

We start as in Claim 1 and assign to each maximally guarded tuple \vec{c} in \mathfrak{D}^u a type in $T(\vec{c}^\dagger)$. This is slightly different from how it was done in Claim 1 due to the different consistency criterion for types used in the definition of a type assignment. If G' is a maximally guarded set in \mathfrak{D} and $t \in T(\mathfrak{D}, G')$, let \vec{c}_t is a tuple consisting of all constants in $\text{bag}(t)$ such that $\vec{c}_t = \vec{b}$ if t is the root $G = [\vec{a}]$ of $(T(\mathfrak{D}, G), E, \text{bag})$. Fix a maximally guarded set G' in \mathfrak{D} . We inductively assign to each node t in $T(\mathfrak{D}, G')$ a type $\Phi_t(\vec{x}_t) \in T(\vec{c}_t^\dagger)$. If $t = G'$, then we let $\Phi_t(\vec{x}_t)$ be any type in $T(\vec{c}_t^\dagger)$. If in addition we have $G' = G$, then $\vec{c}_t^\dagger = \vec{a}$ and we select $\Phi_t(\vec{x}_t)$ so that $q(\vec{x}_t) \notin \Phi_t(\vec{x}_t)$. For the induction step, consider a node $t \in T(\mathfrak{D}, G')$ and all successor nodes $t_i = tG_i$, $i \in \{1, \dots, k\}$, of t in $T(\mathfrak{D}, G')$. Since $\Phi_t(\vec{x}_t) \in T(\vec{c}_t^\dagger)$ and $[\vec{c}_t] \cap [\vec{c}_{t_i}] \neq \emptyset$ for each $i \in \{1, \dots, k\}$, there exists a model \mathfrak{A} of \mathcal{O} with the following properties:

- $\mathfrak{D}^u_{|[\vec{c}_t]} \subseteq \mathfrak{A}$ and $\Phi(\vec{x}_t)$ is the type of \vec{c}_t in \mathfrak{A} ;
- for each $i \in \{1, \dots, k\}$ we have $\mathfrak{D}^u_{|[\vec{c}_{t_i}]} \subseteq \mathfrak{A}$ and there is a type $\Phi_i(\vec{x}_{t_i})$ in $T(\vec{c}_{t_i}^\dagger)$ such that $\Phi_i(\vec{x}_{t_i})$ is the type of \vec{c}_{t_i} in \mathfrak{A} .

We assign to each of the nodes t_i , $i \in \{1, \dots, k\}$ the type $\Phi_i(\vec{x}_i)$. We also assign to t the model $\mathfrak{A}_t := \mathfrak{A}$. In the following, we will assume that $\text{dom}(\mathfrak{D}^u) \cap \text{dom}(\mathfrak{A}_t)$ consists of exactly the constants in $[\vec{c}_t] \cup [\vec{c}_{t_1}] \cup \dots \cup [\vec{c}_{t_k}]$. This concludes the induction step.

We are now ready to construct a model \mathfrak{B} of \mathfrak{D}^u and \mathcal{O} with $\mathfrak{B} \not\models q(\vec{b})$. Let \mathfrak{B}_0 be the union of all interpretations $\mathfrak{A}_t|_{\text{dom}(\mathfrak{D}^u)}$, for $t \in T(\mathfrak{D})$. By the choice of the types $\Phi_t(\vec{x}_t)$ and the interpretations \mathfrak{A}_t , the interpretation \mathfrak{B}_0 contains \mathfrak{D}^u and agrees with \mathfrak{A}_t on all facts that involve only constants in $\text{dom}(\mathfrak{A}_t) \cap \text{dom}(\mathfrak{D}^u)$. We now obtain \mathfrak{B} from \mathfrak{B}_0 by hooking appropriate interpretations \mathfrak{B}_c to each constant $c \in \text{dom}(\mathfrak{D}^u)$. To define \mathfrak{B}_c , fix any node $t \in T(\mathfrak{D})$ with $c \in \text{bag}(t)$. We will extract \mathfrak{B}_c from \mathfrak{A}_t as follows. Take a maximally guarded set G' of \mathfrak{A}_t with $\text{dom}(G') \cap \text{dom}(\mathfrak{D}^u) = \{c\}$, and construct an interpretation $\mathfrak{B}_{G'}$ along the lines of the proof of Lemma 2.12. To this end, start with the uGC₂-unraveling of \mathfrak{A}_t at G' , but then keep only the portion of this unraveling that is defined by the nodes $G_0G_1 \dots G_n$ in $T(\mathfrak{A}_t, G')$ with $\text{dom}(G_1) \cap \text{dom}(\mathfrak{D}^u) = \emptyset$. The interpretation \mathfrak{B}_c is obtained from the disjoint union of all interpretations $\mathfrak{B}_{G'}$, where G' is a maximally guarded set of \mathfrak{A}_t with $\text{dom}(G') \cap \text{dom}(\mathfrak{D}^u) = \{c\}$, by identifying the copy of c in the root bag of each $\mathfrak{B}_{G'}$ with c . It can now be shown that

$$\mathfrak{B} := \mathfrak{B}_0 \cup \bigcup_{c \in \text{dom}(\mathfrak{D}^u)} \mathfrak{B}_c$$

is a model of \mathfrak{D}^u and \mathcal{O} with $\mathfrak{B} \not\models q(\vec{b})$. ┘

It remains to construct the Datalog[≠] program $\Pi_{\mathcal{O}, q}$. We follow a similar strategy as in the case of uGF(=) ontologies, namely that we derive the desired type assignment inductively, starting with a set of all possible types for each guarded tuple in \mathfrak{D} , and removing a type from the set of a guarded tuple \vec{c} whenever it is not consistent with types from the sets of the guarded tuples in \mathfrak{D} that intersect with \vec{c} . Since the number of guarded tuples in \mathfrak{D} that intersect a given guarded tuple in \mathfrak{D} may be unbounded, a Datalog[≠] program cannot implement this strategy directly. We will exploit the fact that in order to establish consistency of a type for \vec{c} it suffices to inspect a bounded number of guarded tuples that

intersect with \vec{c} , where the bound depends only on \mathcal{O} and q . More precisely, let τ be one plus the number of non-equivalent types with at most two free variables, and let N be the largest integer such that a formula of the form $\exists \geq^N x \phi$ occurs in $\text{cl}(\mathcal{O}, q)$, or 1 if there is no such formula in $\text{cl}(\mathcal{O}, q)$. Then the number of intersecting guarded tuples that have to be considered in order to establish consistency is at most $N\tau 2^\tau$. In what follows, we first exploit this fact to construct the Datalog[≠] program $\Pi_{\mathcal{O},q}$, and then prove that $\Pi_{\mathcal{O},q}$ has the intended effect.

Fix variables $z_1, z_2, z_3, \dots, z_m$, where $m := N\tau 2^\tau + 2$. In the description below, $\vec{u}_0, \vec{u}_1, \dots, \vec{u}_{N\tau 2^\tau}$ range over tuples consisting of at most two of these variables, and \vec{w} ranges over k -tuples of variables in $\{z_1, \dots, z_m\}$, where $k \leq 2$ is the number of answer variables of q . Given a type $\Phi_0(\vec{x}_0)$ and sets T_1, \dots, T_ℓ of types with free variables $\vec{x}_1, \dots, \vec{x}_\ell$, we write $\Phi_0(\vec{x}_0) \rightsquigarrow T_1, \dots, T_\ell$ if there is a model \mathfrak{A} of \mathcal{O} and an assignment $\pi: [\vec{x}_0] \cup \dots \cup [\vec{x}_\ell] \rightarrow \text{dom}(\mathfrak{A})$ with the following properties:

- $\Phi_0(\vec{x}_0)$ is the type of $\pi(\vec{x}_0)$ in \mathfrak{A} ;
- for each $i \in \{1, \dots, \ell\}$ there is a type $\Phi_i(\vec{x}_i)$ in T_i such that $\Phi_i(\vec{x}_i)$ is the type of $\pi(\vec{x}_i)$ in \mathfrak{A} .

The rules of $\Pi_{\mathcal{O},q}$ are as follows:

- (1) $P_T(z_i) \leftarrow \alpha$, where α is an atomic formula that involves only the variable z_i , $i \in \{1, \dots, m\}$, and T consists of all types with free variable z_i that contain α ;
- (2) $P_T(\vec{u}_0) \leftarrow R(\vec{u}_0) \wedge \alpha$, where $R \in \text{sig}(\mathcal{O} \cup \{q\})$, α is an atomic formula (possibly an equality) that involves only variables from \vec{u}_0 , and T consists of all types with free variables \vec{u}_0 that contain both $R(\vec{u}_0)$ and α ;
- (3) $P_T(\vec{u}_0) \leftarrow z_i \neq z_j$, where z_i and z_j are distinct variables in \vec{u}_0 , and T consists of all types with free variables \vec{u}_0 that contain $\neg(z_i = z_j)$;
- (4) $P_V(\vec{u}_0) \leftarrow \bigwedge_{i=0}^\ell P_{T_i}(\vec{u}_i)$, where $\ell \leq N\tau 2^\tau$, T_i is a set of types with free variables \vec{u}_i , the tuples \vec{u}_0 and \vec{u}_i share a variable, for each $i \leq \ell$, and V is the set of all types $\Phi_0(\vec{u}_0)$ in T_0 such that $\Phi_0(\vec{u}_0) \rightsquigarrow T_1, \dots, T_\ell$;
- (5) $P_{T \cap T'}(\vec{u}_0) \leftarrow P_T(\vec{u}_0) \wedge P_{T'}(\vec{u}_0)$, where T, T' are sets of types with free variables \vec{u}_0 ;
- (6) $\text{goal}(\vec{w}) \leftarrow P_T(\vec{u}_0)$, where T is a set of types with free variables \vec{u}_0 such that $q(\vec{w})$ is contained in all types in T ;
- (7) $\text{goal}(\vec{w}) \leftarrow P_0(\vec{u}_0)$, where \vec{u}_0 and \vec{w} do not share any variables.

We can now show:

CLAIM 4. $\mathfrak{D} \models \Pi_{\mathcal{O},q}(\vec{a})$ iff $\mathcal{O}, \mathfrak{D} \models q(\vec{a})$.

PROOF. By Claim 3, it suffices to show that $\mathfrak{D} \not\models \Pi_{\mathcal{O},q}(\vec{a})$ iff there exists a type assignment T and a type $\Phi(\vec{x}) \in T(\vec{a})$ with $q(\vec{x}) \notin \Phi(\vec{x})$. The ‘if’ direction is exactly as in Claim 2, so we focus on the ‘only if’ direction.

Assume that $\mathfrak{D} \not\models \Pi_{\mathcal{O},q}(\vec{a})$. Let \mathfrak{A} be the unique minimal model of \mathfrak{D} and $\Pi_{\mathcal{O},q}$. For each guarded tuple \vec{c} in \mathfrak{D} , let $T(\vec{c})$ be the unique inclusion-minimal set of types with free variables z_1 or (z_1, z_2) such that $P_{T(\vec{c})}(\vec{c}) \in \mathfrak{A}$. Then, $T(\vec{c})$ is non-empty. We now show that T is a type assignment.

Consider a maximally guarded tuple \vec{c}_0 in \mathfrak{D} and a type $\Phi_{\vec{c}_0}(\vec{x}_{\vec{c}_0})$ in $T(\vec{c}_0)$. Let C be the set of all maximally guarded tuples $\vec{c} \neq \vec{c}_0$ in \mathfrak{D} that have a non-empty intersection with \vec{c}_0 . For each pair \vec{c}, \vec{c}' of tuples in C define $\vec{c} \sim \vec{c}'$ iff $T(\vec{c}) = T(\vec{c}')$, and let C_1, \dots, C_s be the equivalence classes w.r.t. \sim . Then, $s \leq 2^\tau$. For each $i \in \{1, \dots, s\}$, pick a subset C'_i of C_i of size $\min\{|C_i|, N\tau\}$. Let $\vec{c}_1, \dots, \vec{c}_\ell$ be an enumeration of the tuples in $C'_1 \cup \dots \cup C'_s$. Then,

$\ell \leq N\tau 2^{\tau}$. By construction of $\Pi_{\mathcal{O},q}$, we have $\Phi_{\vec{c}_0}(\vec{x}_{\vec{c}_0}) \rightsquigarrow T_1(\vec{c}_1), \dots, T_\ell(\vec{c}_\ell)$, which implies that there exists a model \mathfrak{B} of \mathcal{O} such that:

- $\Phi_{\vec{c}_0}(\vec{x}_{\vec{c}_0})$ is the type of \vec{c}_0 in \mathfrak{B} ;
- for each $i \in \{1, \dots, \ell\}$ there is a type $\Phi_{\vec{c}_i}(\vec{x}_{\vec{c}_i})$ in $T(\vec{c}_i)$ such that $\Phi_{\vec{c}_i}(\vec{x}_{\vec{c}_i})$ is the type of \vec{c}_i in \mathfrak{B} .

The first three rules of the program ensure that each $\Phi_{\vec{c}_i}(\vec{x}_{\vec{c}_i})$ contains information about all atomic formulas and inequalities that are true about \vec{c}_i in $\mathfrak{D}_{\llbracket \vec{c}_i \rrbracket}$, so the fact that $\Phi_{\vec{c}_i}(\vec{x}_{\vec{c}_i})$ is the type of \vec{c}_i in \mathfrak{B} also implies $\mathfrak{D}_{\llbracket \vec{c}_i \rrbracket} \subseteq \mathfrak{B}$.

Finally, consider any tuple $\vec{c} \in C_i \setminus C'_i$. Note that $|C'_i| = N\tau$, so there is a type $\Phi_{\vec{c}}(\vec{x}_{\vec{c}}) \in T(\vec{c})$ that is assigned to at least N of the tuples in C'_i . Fix such a type for each tuple $\vec{c} \in C_i \setminus C'_i$ and each $i \in \{1, \dots, s\}$. By the choice of N , we can transform \mathfrak{B} into a model \mathfrak{B}' such that:

- $\mathfrak{D}_{\llbracket \vec{c}_0 \rrbracket} \subseteq \mathfrak{B}'$ and $\Phi_{\vec{c}_0}(\vec{x}_{\vec{c}_0})$ is the type of \vec{c}_0 in \mathfrak{B}' ;
- for each $\vec{c} \in C$ we have $\mathfrak{D}_{\llbracket \vec{c} \rrbracket} \subseteq \mathfrak{B}'$ and there is a type $\Phi_{\vec{c}}(\vec{x}_{\vec{c}})$ in $T(\vec{c})$ such that $\Phi_{\vec{c}}(\vec{x}_{\vec{c}})$ is the type of \vec{c} in \mathfrak{B}' .

This implies that the restriction of T to the maximally guarded tuples in \mathfrak{D} is a type assignment. Moreover, since $\text{goal}(\vec{a}) \notin \mathfrak{A}$, there exists a type in $T(\vec{a})$ that does not contain $q(\vec{z})$. \dashv

Altogether, this concludes the proof of the theorem. \square

D PROOFS FOR SECTION 5

Theorem 5.2 (restated) *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 $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$, then \mathcal{O} is unraveling tolerant.*

PROOF. Recall that we proved already that if \mathcal{O} is materializable for the class of cg-tree decomposable instances \mathfrak{D} with $\text{sig}(\mathfrak{D}) \subseteq \text{sig}(\mathcal{O})$, then it is materializable for the class of all cg-tree decomposable instances without any signature restrictions.

Also recall that we proved the result already for ontologies in $uGC_2^-(1,=)$. We next consider $uGF^-(1,=)$. Assume that \mathcal{O} is an ontology in $uGF^-(1,=)$. The proof is similar to the proof for $uGF^-(1,=)$ and we only give a sketch. Let \mathfrak{D} be an instance and \mathfrak{D}^u its global uGF -unraveling. Let \mathfrak{B}^u be a materialization of \mathcal{O} and \mathfrak{D}^u . We may assume that \mathfrak{B}^u is a forest model. Define a model \mathfrak{B} of \mathfrak{D} by

- hooking to \mathfrak{D} at every $c^\dagger \in \text{dom}(\mathfrak{D})$ a copy $\mathfrak{B}_{c^*}^{c^\dagger}$ of the interpretation \mathfrak{B}^u by identifying c^\dagger and c^* (we assume $\text{dom}(\mathfrak{D}) \cap \text{dom}(\mathfrak{B}_{c^*}^{c^\dagger}) = \{c^\dagger\}$) and
- adding the atoms $\{R(\vec{c}^\dagger) \mid R(\vec{c}) \in \mathfrak{B}_{\text{dom}(\mathfrak{D}^u)}^u\}$.

As in the previous case, it suffices to show that \mathfrak{B} is a model of \mathcal{O} and \mathfrak{D} which is connected guarded bisimilar (for the appropriate tuples) to a materialization \mathfrak{B}^* of \mathcal{O} and \mathfrak{D}^u . Define \mathfrak{B}^* by hooking to \mathfrak{D}^u at every $c \in \mathfrak{D}^u$ a copy $\mathfrak{B}_{c^*}^c$ of \mathfrak{B}^u by identifying c and c^* and adding $\mathfrak{B}_{\text{dom}(\mathfrak{D}^u)}^u$. We assume $\text{dom}(\mathfrak{D}^u) \cap \text{dom}(\mathfrak{B}_{c^*}^c) = \{c\}$. We show that \mathfrak{B}^* is a materialization of \mathcal{O} and \mathfrak{D}^u . First, by Lemma 5.1, $\mathfrak{B}^* \models q(\vec{a})$ iff $\mathfrak{B}^u \models q(\vec{a})$ for all $\vec{a} \in \text{dom}(\mathfrak{D}^u)$ and rAQs $q(\vec{x})$. It remains to show that \mathfrak{B}^* is a model of \mathcal{O} . Again it is crucial that \mathcal{O} is an ontology in $uGF^-(1,=)$. Let $\varphi \in \mathcal{O}$. Then φ is of the form $\forall x(x = x \rightarrow \psi(x))$, where $\psi(x)$ is a formula of depth 1 in openGF. Consider $a \in \text{dom}(\mathfrak{B}^*)$. We have to show that $\mathfrak{B}^* \models \psi(a)$. We distinguish two cases:

Case 1. $a \in \text{dom}(\mathfrak{B}_{c^*}^c) \setminus \{c\}$ for some $c \in \text{dom}(\mathfrak{D}^u)$. This case is considered in exactly the same way as Case 1 for ontologies in $\text{uGC}_2^-(1, =)$.

Case 2. $a \in \text{dom}(\mathfrak{D}^u)$. Denote for $c \in \text{dom}(\mathfrak{D}^u)$ by $N(c)$ the set of all $d \in \text{dom}(\mathfrak{D}^u)$ such that there exists a guarded set G in \mathfrak{D}^u with $c, d \in G$. By Lemma 5.1, the interpretations $\mathfrak{B}_{|N(c)}^u$ and $\mathfrak{B}_{|N(c^*)}^u$ are isomorphic for every $c \in \text{dom}(\mathfrak{D}^u)$. Now the argument is exactly the same as in Case 2 for ontologies in $\text{uGC}_2^-(1, =)$.

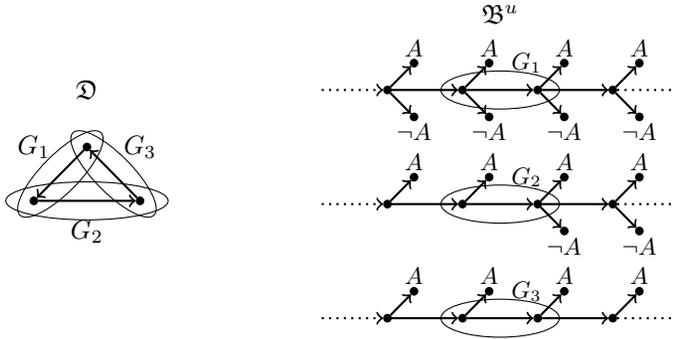
We have shown that \mathfrak{B}^* is a materialization of \mathcal{O} and \mathfrak{D}^u . The required connected guarded bisimulation between \mathfrak{B}^* and \mathfrak{B} is obtained by taking the set of partial isomorphisms between \mathfrak{D}^u and \mathfrak{D} from Lemma 2.10 and adding the induced partial isomorphisms between guarded sets from the obvious isomorphisms between $\mathfrak{B}_{c^*}^c$, $c \in \mathfrak{D}^u$, and the $\mathfrak{B}_{c^*}^{c^\dagger}$ hooked to c^\dagger in \mathfrak{B} .

The proofs given above do not work for the remaining fragments. The reason is that the model \mathfrak{B}^* defined above by hooking to \mathfrak{D}^u at c certain cg-tree decomposable \mathfrak{B}_{c^*} obtained from \mathfrak{B}^u is not guaranteed to be a model of \mathcal{O} , and so \mathfrak{B} is not guaranteed to be a model of \mathcal{O} either. The following examples illustrates the situation.

Example D.1. Let \mathcal{O} contain

$$\forall x \exists y (S(x, y) \wedge A(y)), \quad \forall x, y (R(x, y) \rightarrow (\varphi(x) \rightarrow \varphi(y))), \quad \text{where } \varphi(x) = \exists z (S(x, z) \wedge \neg A(z)).$$

Thus, in every model \mathfrak{A} of \mathcal{O} each node has an S -successor in A and having an S -successor b with $A(b) \notin \mathfrak{A}$ is propagated along R . \mathcal{O} is unraveling tolerant. Consider the instance \mathfrak{D} from Example 2.9 (1) depicted here again with the maximally guarded sets G_1, G_2, G_3 .



We have seen that the uGC_2 -unraveling $\mathfrak{D}_{G_i}^u$ of \mathfrak{D} at G_i consists of a single chain. The global unraveling \mathfrak{D}^u of \mathfrak{D} thus consists of three chains. An example of a forest model \mathfrak{B}^u of \mathcal{O} and \mathfrak{D}^u is given on the right hand side of the figure. When we now choose $c^* \in \{b \mid c \sim^u b\}$ arbitrarily when constructing \mathfrak{B}^* , then it is not guaranteed that we obtain a model in which the propagation condition for the existence of S -successors b with $A(b) \notin \mathfrak{B}^*$ holds.

For the remaining fragments we therefore

- expand \mathfrak{D}^u to a new cg-tree decomposable instance \mathfrak{D}^{u+} by adding entailed rAQs to \mathfrak{D}^u ; in the example above *every* $a \in \mathfrak{D}^u$ now has an S -successor b with $A(b) \notin \mathfrak{D}^{u+}$;
- take a materialization \mathfrak{B}^{u+} of \mathcal{O} and \mathfrak{D}^{u+} and define the model \mathfrak{B} of \mathcal{O} and \mathfrak{D} by hooking to \mathfrak{D} appropriate cg-tree decomposable interpretations hooked to \mathfrak{D}^u in \mathfrak{B}^{u+} ;
- prove that \mathfrak{B} is a model of \mathcal{O} with $\mathfrak{B} \not\models q_0(\bar{a})$ by constructing an appropriate guarded bisimulation from a uniformization \mathfrak{B}^{u*} of \mathfrak{B}^{u+} .

Thus, the main difference to the proof above is that we first expand \mathfrak{D}^u to a new instance \mathfrak{D}^{u+} and then work with a materialization of \mathfrak{D}^{u+} instead of \mathfrak{D}^u . For this to work it is crucial that any materialization of \mathfrak{D}^{u+} is a materialization of \mathfrak{D}^u as well, and therefore $\mathcal{O}, \mathfrak{D}^u \models q(\vec{a})$ iff $\mathcal{O}, \mathfrak{D}^{u+} \models q(\vec{a})$ for all tuples \vec{a} in \mathfrak{D}^u and all rAQs $q(\vec{x})$. For uGF(1) and uGF₂⁻(2) this will follow directly from the observation that they are fragments of FO without equality and thus answers to CQs are preserved under homomorphisms of instances [15]. For *ALCHIF* this preservation result does not hold and a more careful construction of \mathfrak{D}^{u+} is needed to ensure this property.

Let \mathfrak{D} be an instance and \mathfrak{D}^u its global uGF unraveling. Let \vec{a} be a tuple with $[\vec{a}] = G_0$ maximally guarded in \mathfrak{D} and let \vec{b} be the copy in $\text{bag}(G_0)$ of \vec{a} . Further let q_0 be an rAQ such that $\mathcal{O}, \mathfrak{D}^u \not\models q_0(\vec{b})$. We show that $\mathcal{O}, \mathfrak{D} \not\models q_0(\vec{a})$. We first hook to \mathfrak{D}^u at any $\text{bag}(t)$ with $t \in T(\mathfrak{D})$ a copy of any rAQ entailed by \mathcal{O} and \mathfrak{D}^u at $\text{bag}(t)$. In detail, let \mathfrak{D}_t be the union of all canonical instances $\mathfrak{D}_q(\vec{a})$, where q is an rAQ, $[\vec{a}] \subseteq \text{bag}(t)$, $\mathcal{O}, \mathfrak{D}^u \models q(\vec{a})$, and where we assume that $\text{dom}(\mathfrak{D}_q(\vec{a})) \cap \text{dom}(\mathfrak{D}^u) = [\vec{a}]$ and $\text{dom}(\mathfrak{D}_q(\vec{a})) \cap \text{dom}(\mathfrak{D}_{q'}(\vec{a}')) = [\vec{a}] \cap [\vec{a}']$ for $\mathfrak{D}_q(\vec{a}) \neq \mathfrak{D}_{q'}(\vec{a}')$. Then let $\mathfrak{D}^{u+} = \mathfrak{D}^u \cup \bigcup_{t \in T(\mathfrak{D})} \mathfrak{D}_t$. The following properties of \mathfrak{D}^{u+} follow directly from the definition and Lemma 4.6.

- (a) 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 $h_{t,t'}$ from $\text{bag}(t)$ to $\text{bag}(t')$, and an automorphism $h_{t,t'}^+$ of \mathfrak{D}^{u+} that extends the canonical automorphism $\hat{h}_{t,t'}$ of \mathfrak{D}^u .
- (b) there is a homomorphism from \mathfrak{D}^{u+} to any materialization of \mathcal{O} and \mathfrak{D}^u preserving $\text{dom}(\mathfrak{D}^u)$. Thus, if \mathcal{O} is given in FO without equality then by, preservation of certain answers under homomorphisms between instances ([15], Proposition 5.9), any materialization of \mathfrak{D}^{u+} is a materialization of \mathfrak{D}^u and for every rAQ $q(\vec{x})$ and \vec{a} in \mathfrak{D}^u :

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

As \mathfrak{D}^{u+} is cg-tree decomposable, there is a materialization \mathfrak{B}^{u+} of \mathcal{O} and \mathfrak{D}^{u+} which is a forest model of \mathcal{O} and \mathfrak{D}^{u+} . Thus, \mathfrak{B}^{u+} is obtained from \mathfrak{D}^u by hooking to \mathfrak{D}^u at every $\text{bag}(t)$, $t \in T(\mathfrak{D})$, a cg-tree decomposable model \mathfrak{B}_t^{u+} of \mathfrak{D}_t . Observe that we obtain from Point (a):

- (c) For any $t, t' \in T(\mathfrak{D})$ with $t \sim t'$, the mapping $h_{t,t'}^+$ is an automorphism of $\mathfrak{B}_{\mathfrak{D}^{u+}}^{u+}$ and its restriction to $\text{dom}(\mathfrak{D}_t)$ is an isomorphism from $\mathfrak{B}_{|\text{dom}(\mathfrak{D}_t)}^{u+}$ onto $\mathfrak{B}_{|\text{dom}(\mathfrak{D}_{t'})}^{u+}$.

The following result states that every finite subinterpretation of \mathfrak{B}_t^{u+} exists already in $\mathfrak{B}_{|\text{dom}(\mathfrak{D}_t)}^{u+}$ (up to renaming).

- (d) Let $t \in T(\mathfrak{D})$. For any finite subinterpretation \mathfrak{A} of \mathfrak{B}_t^{u+} there exists an isomorphic embedding of \mathfrak{A} into $\mathfrak{B}_{|\text{dom}(\mathfrak{D}_t)}^{u+}$ preserving $\text{bag}(t)$.

To prove (d), we may assume that \mathfrak{A} is connected and $\text{dom}(\mathfrak{A}) \cap \text{dom}(\mathfrak{B}_t^{u+}) = \text{bag}(t)$. By Point (b), there is an isomorphism between \mathfrak{A} and some $\mathfrak{D}_q(\vec{a})$ from the construction of \mathfrak{D}_t preserving $\text{bag}(t)$. Fix $\mathfrak{D}_q(\vec{a})$. It remains to be proved that there does not exist any $R(\vec{b})$ with $[\vec{b}] \subseteq \text{dom}(\mathfrak{D}_q(\vec{a}))$ such that $R(\vec{b}) \in \mathfrak{B}^{u+} \setminus \mathfrak{D}_q(\vec{a})$. But using the fact that \mathfrak{A} is a subinterpretation of the model \mathfrak{B}^{u+} of \mathcal{O} and \mathfrak{D}^{u+} isomorphic to $\mathfrak{D}_q(\vec{a})$ one can easily construct a model of \mathfrak{D}^{u+} and \mathcal{O} that contains no $R(\vec{b}) \notin \mathfrak{D}_q(\vec{a})$ with $[\vec{b}] \subseteq \text{dom}(\mathfrak{D}_q(\vec{a}))$. Thus, as \mathfrak{B}^{u+} is a materialization of \mathcal{O} and \mathfrak{D}^{u+} , \mathfrak{B}^{u+} contains no such $R(\vec{b})$. This finishes the proof of (d). In what follows we require the following consequence of Points (c) and (d):

- (e) For all $t, t' \in T(\mathfrak{D})$ with $t \sim t'$ and all guarded tuples \vec{a} in \mathfrak{B}_t^{u+} there exists a guarded tuple \vec{b} in $\mathfrak{B}_{|\text{dom}(\mathfrak{D}_{t'})}^{u+}$ and mapping $p : \vec{a} \mapsto \vec{b}$ which coincides with $h_{t,t'}$ on $\text{bag}(t)$ such that p is an isomorphism from $\mathfrak{B}_{|\vec{a}}^{u+}$ to $\mathfrak{B}_{|\vec{b}}^{u+}$.

The next steps depend on whether we consider ontologies in $\text{uGF}(1)$ or $\text{uGF}_2^-(2)$. We start with the former case. Fix for every equivalence class $\{t' \mid t \sim t'\}$ a $t^* \sim t$. Define a model \mathfrak{B} of \mathfrak{D} by hooking to \mathfrak{D} at every $\text{bag}(t)^\dagger$ in \mathfrak{D} a copy of the interpretation \mathfrak{B}_t^{u+} by identifying every $a \in \text{bag}(t^*)$ with a^\dagger . To show that \mathfrak{B} is a model of \mathcal{O} and $\mathfrak{B} \not\models q_0(\vec{a})$ we uniformize \mathfrak{B}^{u+} as before and then take an appropriate connected guarded bisimulation between the uniformization and \mathfrak{B} . Define \mathfrak{B}^{u*} by hooking to \mathfrak{D}^u at every $\text{bag}(t)$, $t \in T(\mathfrak{D})$, a copy \mathfrak{B}_t^{u*} of the interpretation \mathfrak{B}_t^{u+} by identifying every $a \in \text{bag}(t)$ with the unique $a' \in \text{bag}(t^*)$ with $a \sim^u a'$. We show that \mathfrak{B}^{u*} is a materialization of \mathfrak{D} and \mathcal{O} . Denote for $a \in \text{dom}(\mathfrak{B}_t^{u*})$ by a' the corresponding element of \mathfrak{B}_t^{u+} (thus, for $a \in \text{bag}(t)$ we have $a \sim^u a'$). 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 the construction of \mathfrak{B}^{u*} from \mathfrak{B}^{u+} preserves answers to rAQs. Consider a sentence $\varphi \in \mathcal{O}$. Then $\varphi = \forall \vec{y}(R(\vec{y}) \rightarrow \psi(\vec{y}))$, where $\psi(\vec{y})$ 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 tuple $\vec{a} = (a_1, \dots, a_k)$ in $\text{dom}(\mathfrak{B}^{u*})$. Then $[\vec{a}] \subseteq \text{dom}(\mathfrak{B}_t^{u*})$ for some $t \in T(\mathfrak{D})$. $\mathfrak{B}^{u*} \models \varphi$ follows from $\mathfrak{B}^{u+} \models \varphi$ if we can show that $\mathfrak{B}^{u*} \models \psi(\vec{a})$ iff $\mathfrak{B}^{u+} \models \psi(\vec{a}')$ where $\vec{a}' = (a'_1, \dots, a'_k)$.

To show that $\mathfrak{B}^{u*} \models \psi(\vec{a})$ iff $\mathfrak{B}^{u+} \models \psi(\vec{a}')$ it suffices to construct a connected depth 1 guarded bisimulation between $(\mathfrak{B}^{u*}, \vec{a})$ and $(\mathfrak{B}^{u+}, \vec{a}')$; i.e., to prove that for any guarded \vec{b} in \mathfrak{B}^{u*} with $[\vec{b}] \cap [\vec{a}] \neq \emptyset$ there exists a guarded \vec{c} in \mathfrak{B}^{u+} with $[\vec{c}] \cap [\vec{a}'] \neq \emptyset$ such that there is a partial isomorphism $p : \vec{b} \mapsto \vec{c}$ with $p(a) = a'$ for all $a \in [\vec{b}] \cap [\vec{a}]$ and vice versa. We prove the first direction, the converse is similar. Consider a guarded \vec{b} in \mathfrak{B}^{u*} with $[\vec{b}] \cap [\vec{a}] \neq \emptyset$. We distinguish two cases.

Case 1. $[\vec{b}] \cap \text{dom}(\mathfrak{B}_t^{u*}) \setminus \text{bag}(t) \neq \emptyset$. Then $[\vec{b}] \subseteq \text{dom}(\mathfrak{B}_t^{u*})$ and the claim follows directly from the fact that \mathfrak{B}_t^{u*} is a copy of \mathfrak{B}_t^{u+} ($\vec{c} = \vec{b}'$ is as required).

Case 2. $[\vec{b}] \cap \text{dom}(\mathfrak{B}_t^{u*}) \setminus \text{bag}(t) = \emptyset$. There exists $t' \in T(\mathfrak{D})$ (possibly $t' \neq t$) such that $[\vec{b}] \subseteq \text{dom}(\mathfrak{B}_{t'}^{u*})$. By (e), there is a guarded \vec{e} in $\mathfrak{B}_{|\text{dom}(\mathfrak{D}_{t'})}^{u+}$ and a mapping $p : \vec{b} \mapsto \vec{e}$ which is the identity on $\text{bag}(t') \cap [\vec{b}]$ such that p is an isomorphism from $\mathfrak{B}_{|\vec{b}}^{u*}$ to $\mathfrak{B}_{|\vec{e}}^{u+}$. Then $\vec{c} = h_{t,t'}^+(\vec{e})$ is as required.

We have shown that \mathfrak{B}^{u*} is a materialization of \mathcal{O} and \mathfrak{D}^u . One can now construct in the same way as before a connected guarded bisimulation between \mathfrak{B}^* and \mathfrak{B} showing that \mathfrak{B} is a model of \mathcal{O} and $\mathfrak{B} \not\models q_0(\vec{a})$.

Now let \mathcal{O} be in $\text{uGF}_2^-(2)$ and let \mathfrak{D}^u be the global GC_2 -unraveling of \mathfrak{D} . Assume that \mathfrak{D}^{u+} and \mathfrak{B}^{u+} are defined as above. The construction of the model \mathfrak{B} of \mathcal{O} and \mathfrak{D} is very similar to the $\text{uGC}_2^-(1, =)$ case except that we now use the model \mathfrak{B}^{u+} instead of \mathfrak{B}^u to construct \mathfrak{B} . Thus, we define \mathfrak{B} by

- hooking to \mathfrak{D} at every $c^\dagger \in \text{dom}(\mathfrak{D})$ a copy $\mathfrak{B}_{c^*}^{\dagger}$ of the interpretation \mathfrak{B}_{c^*} hooked to \mathfrak{D}^u at c^* in \mathfrak{B}^{u+} and
- adding the atoms $\{R(c_1^\dagger, c_2^\dagger) \mid R(c_1, c_2) \in \mathfrak{B}_{|\text{dom}(\mathfrak{D}^u)}^{u+}\}$.

To prove that \mathfrak{B} is a model of \mathcal{O} and \mathfrak{D} such that $\mathfrak{B} \not\models q_0(\vec{a})$, we uniformize \mathfrak{B}^{u+} , as before: define \mathfrak{B}^{u*} by hooking to \mathfrak{D}^u at c a copy $\mathfrak{B}_{c^*}^{\dagger}$ of the model \mathfrak{B}_{c^*} hooked to \mathfrak{D}^u at c^* in \mathfrak{B}^{u+} , for every $c \in \text{dom}(\mathfrak{D}^u)$ and adding $\mathfrak{B}_{|\text{dom}(\mathfrak{D}^u)}$. It can be shown as before that \mathfrak{B}^{u*} is a

materialization of \mathcal{O} and \mathfrak{D}^u and can be used show that \mathfrak{B} is a model of \mathfrak{D} with $\mathfrak{B} \not\models q_0(\vec{a})$ if we can show that \mathfrak{B}^{u*} is a model of \mathcal{O} . So we focus on showing that \mathfrak{B}^* is a model of \mathcal{O} . Let $\varphi \in \mathcal{O}$. Then $\varphi = \forall x(x = x \rightarrow \psi(x))$ for a formula $\psi(x)$ of depth 2 in openGF_2 . Let $a \in \text{dom}(\mathfrak{B}^{u*})$. We have to show that $\mathfrak{B}^{u*} \models \psi(a)$. To prove this let for any $a \in \text{dom}(\mathfrak{D}^u)$, $N(a) = \{a\} \cup \{b \mid R(a, b) \in \mathfrak{B}^{u+} \text{ or } R(b, a) \in \mathfrak{B}^{u+}\}$. Then (e) implies for all $a \in \text{dom}(\mathfrak{D}^u)$:

- (e') there is an isomorphism p from $\mathfrak{B}_{|N(a)}^{u+}$ to $\mathfrak{B}_{|N(a^*)}^{u+}$ mapping any $b \in N(a) \cap \text{dom}(\mathfrak{D}^u)$ to $p(b) \in \text{dom}(\mathfrak{D}^u)$ such that $p(b) \sim^u b$.

We now distinguish two cases.

Case 1. $a \in \text{dom}(\mathfrak{B}_{c^*}^c) \setminus \{c\}$ for some $c \in \text{dom}(\mathfrak{D}^u)$. Let a' be the element corresponding to a in $\mathfrak{B}_{c^*}^c$. As ψ has depth 2 and by (e'), $\mathfrak{B}^{u*} \models \psi(a)$ iff $\mathfrak{B}^{u+} \models \psi(a')$ and the claim follows from the assumption that \mathfrak{B}^{u+} is a model of \mathcal{O} .

Case 2. $a \in \text{dom}(\mathfrak{D}^u)$. As ψ has depth 2 and by (e'), $\mathfrak{B}^{u*} \models \psi(a)$ iff $\mathfrak{B}^{u+} \models \psi(a^*)$ and the claim follows from the assumption that \mathfrak{B}^{u+} is a model of \mathcal{O} .

Finally, assume that \mathcal{O} is a *ALCHIF* ontology of depth 2. The proof that follows is similar to the construction for $\text{uGF}_2^-(2)$, but one cannot hook to every $\text{bag}(t)$ all entailed rAQs as this can obviously lead to violations of the functionality axioms in \mathcal{O} .

Let \mathfrak{D} be an instance with $\mathcal{O}, \mathfrak{D} \not\models q_0(\vec{a})$ and let \mathfrak{D}^u be its global GC_2 -unraveling. Let \mathfrak{A} be a materialization of \mathcal{O} and \mathfrak{D}^u . We define for every $c \in \text{dom}(\mathfrak{D}^u)$ an instance \mathfrak{D}_c . 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{A} mapping x to c and such that $R(h(x), h(y)) \notin \mathfrak{A}$ for any R functional in \mathcal{O} and $R(h(y), h(x)) \notin \mathfrak{A}$ for any R^- functional in \mathcal{O} . Then let the instance \mathfrak{D}_c contain a copy of every such \mathfrak{D}_q obtained by identifying the variable x with c . Define \mathfrak{D}^{u+} by hooking to \mathfrak{D}^u at every c the instance \mathfrak{D}_c and adding $\mathfrak{A}_{|\text{dom}(\mathfrak{D}^u)}$. The following properties of \mathfrak{D}^{u+} follow directly from the definition.

- (a) for all $a, b \in \text{dom}(\mathfrak{D}^u)$ with $a \sim^u b$ there is an isomorphism from \mathfrak{D}_a onto \mathfrak{D}_b mapping a to b ;
- (b) for every rAQ $q(\vec{x})$ and tuple \vec{a} in $\text{dom}(\mathfrak{D}^u)$:

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

Take a materialization \mathfrak{B}^{u+} of \mathcal{O} and \mathfrak{D}^{u+} obtained from \mathfrak{D}^{u+} by hooking to every $c \in \text{dom}(\mathfrak{D}^u)$ a cg-tree decomposable model \mathfrak{B}_c of \mathfrak{D}_c . Define a model \mathfrak{B} of \mathfrak{D} by

- hooking to \mathfrak{D} at every $c^\uparrow \in \text{dom}(\mathfrak{D})$ a copy $\mathfrak{B}_{c^*}^{\uparrow}$ of the interpretation \mathfrak{B}_{c^*} hooked to \mathfrak{D}^u at c^* in \mathfrak{B}^{u+} and
- adding the atoms $\{R(c_1^\uparrow, c_2^\uparrow) \mid R(c_1, c_2) \in \mathfrak{B}_{|\text{dom}(\mathfrak{D}^u)}^{u+}\}$.

One can now show in the same way as in the proof for $\text{GF}_2^-(2)$ that \mathfrak{B} is a model of \mathcal{O} such that $\mathfrak{B} \not\models q_0(\vec{a})$. □

E PROOFS FOR SECTION 6

Lemma 6.7 (restated) *For all Σ -instances \mathfrak{D} and $a \in \text{dom}(\mathfrak{D})$, $\mathcal{O}, \mathfrak{D} \models q(a)$ iff $\mathfrak{D} \models \Pi(a)$.*

PROOF. First assume that $\mathcal{O}, \mathfrak{D} \not\models q(a)$. Then there is a model \mathfrak{A} of \mathcal{O} and \mathfrak{D} such that $\mathfrak{A} \not\models q(a)$. Let \mathfrak{B} be the extension of \mathfrak{D} obtained by adding, for all $b \in \text{dom}(\mathfrak{D})$, $P_{\mathfrak{A}(b)}(b)$; moreover if $\mathfrak{A} \models q(b)$, then also add $\text{goal}(b)$. It can be verified that \mathfrak{B} satisfies all rules

in Π . In particular, realizability of all relevant typed links is witnessed by \mathfrak{A} . Since clearly $\text{goal}(a) \notin \mathfrak{B}$, we obtain $\mathfrak{D} \not\models \Pi(a)$, as required.

Assume conversely that $\mathfrak{D} \not\models \Pi(a)$. Then there is an extension \mathfrak{B} of \mathfrak{D} to the intensional relation symbols of Π such that all rules in Π are satisfied and $\text{goal}(a) \notin \mathfrak{B}$. Assume w.l.o.g. that \mathfrak{B} is a minimal such extension of \mathfrak{D} (w.r.t. set inclusion). Then clearly there is a unique fact $P_t(b)$ in \mathfrak{B} , for every $b \in \text{dom}(\mathfrak{D})$. We use t_b to denote t . Note that there must be a model \mathfrak{A}_b of \mathcal{O} and t_b in the sense that $\mathfrak{A}_b \models \mathcal{O}$, $b \in \text{dom}(\mathfrak{A}_b)$, and $t^{\mathfrak{A}_b}(b) = t_b$. If there is no such model, then the link t_b, \emptyset, t_b would not be realizable, in contrast to the third type of rule being satisfied in \mathfrak{B} . Also note that for all distinct $b_1, b_2 \in \text{dom}(\mathfrak{D})$, there is a model \mathfrak{A}_{b_1, b_2} of \mathcal{O} and the typed link $t_{b_1}, \mathcal{R}^{\mathfrak{D}}(b_1, b_2), t_{b_2}$ in the sense that $\mathfrak{A}_{b_1, b_2} \models \mathcal{O}$, $b_1, b_2 \in \text{dom}(\mathfrak{A}_{b_1, b_2})$, $t^{\mathfrak{A}_{b_1, b_2}}(b_i) = t_{b_i}$ for $i \in \{1, 2\}$, and $\mathcal{R}^{\mathfrak{A}_{b_1, b_2}}(b_1, b_2) \supseteq \mathcal{R}^{\mathfrak{D}}(b_1, b_2)$. We may assume w.l.o.g. that $\mathfrak{A}_{b_1, b_2} = \mathfrak{A}_{b_2, b_1}$. All models \mathfrak{A}_b and \mathfrak{A}_{b_1, b_2} must satisfy the same sentences since, due to the rules in Π of the third kind, when P_{t_1} and P_{t_2} are non-empty in \mathfrak{B} , then t_1 and t_2 must contain the same sentences. We use Γ to denote the set of these sentences. Clearly, $\mathcal{O} \subseteq \Gamma$.

We assemble an interpretation \mathfrak{A} as follows:

- (1) Start with \mathfrak{A} being the result of hooking \mathfrak{A}_b to \mathfrak{D} for each $b \in \text{dom}(\mathfrak{D})$.
- (2) For all distinct $b_1, b_2 \in \text{dom}(\mathfrak{D})$, extend \mathfrak{A} constructed with all facts of the form $R(b_1, b_2)$ or $R(b_2, b_1)$ from \mathfrak{A}_{b_1, b_2} .

By construction, \mathfrak{A} is a model of \mathfrak{D} . We next observe the following.

CLAIM 1.

- (1) $\mathfrak{A} \models \varphi()$ iff $\varphi \in \Gamma()$ for all sentences $\varphi() \in \text{cl}(\mathcal{O}, q)$
- (2) $\mathfrak{A} \models \varphi(b)$ iff $\varphi(x) \in t_b$ for all $\varphi(x) \in \text{cl}(\mathcal{O}, q)$ and $b \in \text{dom}(\mathfrak{D})$;
- (3) $\mathfrak{A} \models \varphi(b')$ iff $\mathfrak{A}_b \models \varphi(b')$ for all $\varphi(x) \in \text{cl}(\mathcal{O}, q)$ and $b' \in \text{dom}(\mathfrak{A}_b)$, $b \in \text{dom}(\mathfrak{D})$.
- (4) $\mathfrak{A} \models \varphi(b_1, b_2)$ iff $\mathfrak{A}_{b_1, b_2} \models \varphi(b_1, b_2)$ for all subformulas $\varphi(x, y)$ of \mathcal{O} and distinct $b_1, b_2 \in \text{dom}(\mathfrak{D})$;
- (5) $\mathfrak{A} \models \varphi(b_1, b_2)$ iff $\mathfrak{A}_b \models \varphi(b_1, b_2)$ for all subformulas $\varphi(x, y)$ of \mathcal{O} and distinct $b_1, b_2 \in \text{dom}(\mathfrak{A}_b)$, $b \in \text{dom}(\mathfrak{D})$.

All five points can be proved by a mutual induction on the structure of the sentences $\varphi()$ and formulas $\varphi(x)$. Details are rather straightforward and omitted.

It remains to remark that $\mathcal{O} \subseteq \Gamma$ implies $\mathfrak{A} \models \mathcal{O}$ by Point (1) and $\text{goal}(a) \notin \mathfrak{B}$ together with the rules in Π of the second kind implies that $q(x) \notin t_a$ and thus $\mathfrak{A} \not\models q(a)$ by Point (4). \square

F PROOFS FOR SECTION 7

Lemma 7.2 (restated) *The ontology $\mathcal{O}_{\text{cell}}$ has the following properties for all instances \mathfrak{D} :*

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

PROOF. We first derive a necessary and sufficient condition for consistency of instances \mathfrak{D} w.r.t. $\mathcal{O}_{\text{cell}}$. Lemma 7.2 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. $\mathcal{O}_{\text{cell}}$:

- (c1) all X, Y, X^-, Y^- are functional in \mathfrak{D} ;

- (c2) \mathfrak{D} is consistent w.r.t. the sentences $\exists^W(= 1R_i)$ in $\mathcal{O}_{\text{cell}}$;
- (c3) 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 $^\neq$. By (c2) we may 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 $e_0 \leq \dots \leq e_n$ with $e_i \neq e_j$ for all $i \neq j$, or
- E is a cycle $e_0 \leq \dots \leq e_n$ with $e_i = e_j$ iff $\{i, j\} = \{0, n\}$ for all $i \neq j$.

We say that sets $E_1, E_2 \subseteq E$ partition E if $E_1 \cup E_2 = E$ and $E_1 \cap E_2 = \emptyset$. If E is not a singleton $\{e\}$ with $e \leq e$, then clearly there is a partition E_1, E_2 of E such that

(\dagger) if $e \leq e' \leq e''$, then $\{e, e', e''\} \not\subseteq E_i$, for $i = 1, 2$.

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

$$E_j^{-1} = \{d \in E \mid \mathfrak{D} \models (\geq 2R_i)(d)\}$$

CLAIM 1. Assume \mathfrak{D} satisfies Conditions (c1), (c2), and (c3). Then \mathfrak{D} is consistent w.r.t. $\mathcal{O}_{\text{cell}}$ iff the following conditions hold for all equivalence classes E :

(cell $^+$) if $E = \{e\}$ with $e \leq e$, then $e \notin E_1^{-1} \cup E_2^{-1}$;

(cell $^-$) if $|E| \geq 2$, then there exists a partition E_1, E_2 of E with $E_i \supseteq E_i^{-1}$ satisfying (\dagger).

Moreover, if (cell $^+$) and (cell $^-$) hold, then a materialization \mathfrak{B} satisfying the conditions of Lemma 7.2 (1) exists.

PROOF. (\Rightarrow) Let \mathfrak{D} be consistent w.r.t. $\mathcal{O}_{\text{cell}}$. First assume that Condition (cell $^+$) 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 $e \in E_j^{-1}$ ($i \neq j$). Now assume that (cell $^-$) does not hold. So there exists E with $|E| \geq 2$ such that there exists no partition E_1, E_2 of E with $E_i \supseteq E_i^{-1}$ satisfying (\dagger). Then the axioms under (2) and (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 $E_1^{-1} = E_2^{-1} = \emptyset$. But this contradicts our assumption that there is no partition E_1, E_2 of E with $E_i \supseteq E_i^{-1}$ satisfying (\dagger).

(\Leftarrow) Assume (cell $^+$) and (cell $^-$) 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 such that the $d \in \text{dom}(\mathfrak{D})$ for which there exist $X(d, d_1), Y(d_1, e), Y(d, d_2), X(d_2, e) \in \mathfrak{D}$ has exactly one P -successor. For any equivalence class E with $|E| \geq 2$ we can construct the relevant part of \mathfrak{B} such that each $e \in E_i$ has exactly one R_i -successor and each $e \in E \setminus E_i$ has at least two R_i -successors. As E_1 and E_2 are mutually disjoint, the axioms under (5) are satisfied. As (\dagger) is satisfied, the axioms under (4) are satisfied. As $E_1 \cup E_2$ contains E , the axiom (2) is satisfied. Thus, we can satisfy $(\geq 2P)$ in every $d \in \text{dom}(\mathfrak{D})$ such that $\mathfrak{D} \not\models \text{cell}(d)$ without violating axiom (3). \square

The construction under (\Leftarrow) shows that we can construct a materialization \mathfrak{B} satisfying Point (1) of Lemma 7.2. The proof of Point (2) of Lemma 7.2 is straightforward. For Point (3), observe that Conditions (cell $^+$) and (cell $^-$) of Claim 1 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 the equivalence (3) holds for any CQ q , instance \mathfrak{D} consistent w.r.t. $\mathcal{O}_{\text{cell}}$, and any \vec{d} in \mathfrak{D} . \square

G PROOFS FOR SECTION 8

This section presents a detailed proof of Theorem 8.4. We use the terminology and notation from the main body of the paper. In particular, see Section 8 for definitions and assumptions relevant to non-deterministic Turing machines (TMs) and to the run fitting problem.

Given a TM M , we denote by $L(M)$ the set of strings accepted by M , and by $\text{RF}(M)$ the run fitting problem for M . As observed in Section 8, $\text{RF}(M)$ is in NP for every TM M . We will now prove the following theorem by a careful adaptation of the proof of Ladner's theorem given in [3].

Theorem 8.4 (restated) *There is a TM whose whose run fitting problem is neither in PTIME nor NP-hard, unless PTIME = NP.*

The proof is a modification of the construction used in Impagliazzo's version of the proof of Ladner's Theorem [52], as presented in [3, Theorem 3.3].

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

- (1) Check if there exists 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) Start M_{SAT} in configuration 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 $H: \mathbb{N} \rightarrow \mathbb{N}$. 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 indices i such that $L(M_i) = A$.⁴ Then, $H(n)$ is defined as

$$H(n) := \min \{i < \log \log n \mid \text{for all } z \text{ of length } \leq \log n, M_i \text{ accepts } z \text{ iff } z \in \text{RF}(M_H)\} \cup \{\log \log n\}.$$

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 [3].

This finishes the construction of M_H . Lemma G.2 below shows that $\text{RF}(M_H)$ has the desired properties, namely that $\text{RF}(M_H)$ is neither in PTIME nor NP-complete, unless PTIME = NP. It uses the following auxiliary lemma.

LEMMA G.1.

- If $\text{RF}(M_H)$ is in PTIME, then $H(n) = O(1)$.
- If $\text{RF}(M_H)$ is not in PTIME, then $\lim_{n \rightarrow \infty} H(n) = \infty$.

⁴It 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).

PROOF. The proof is as in [3], but we here provide a proof for the sake of completeness. Assume first that $\text{RF}(M_H)$ is in PTIME. Then, there is an index i such that $L(M_i) = \text{RF}(M_H)$. Now, for all $n > 2^{2^i}$, we have $i < \log \log n$, which implies $H(n) \leq i$ 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 $\text{RF}(M_H)$ is not in PTIME. For a contradiction, suppose that $\lim_{n \rightarrow \infty} H(n) \neq \infty$. 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 $\text{RF}(M_H)$ on all strings of length at most $\log n$. Since this holds for all $n \geq n_0$, we conclude that $L(M_i) = \text{RF}(M_H)$. But then, $\text{RF}(M_H)$ is in PTIME, which contradicts our initial assumption that $\text{RF}(M_H)$ is not in PTIME. \square

We now prove the main lemma, which concludes the proof of Theorem 8.4.

LEMMA G.2. *If $\text{PTIME} \neq \text{NP}$, then $\text{RF}(M_H)$ is neither in PTIME nor NP-complete.*

PROOF. ‘ $\text{RF}(M_H)$ is not in PTIME’: For a contradiction, suppose that $\text{RF}(M_H)$ is in PTIME. By Lemma G.1, 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 $\text{RF}(M_H)$, which implies $\text{PTIME} = \text{NP}$ and leads to the desired contradiction.

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 $\tilde{\gamma}_0, \dots, \tilde{\gamma}_{i+p(n)}$ of M_H such that:
 - $\tilde{\gamma}_0, \dots, \tilde{\gamma}_i$ corresponds to the initialization phase of M_H on input w that generates the start configuration of M_{SAT} on input x . In particular, $\tilde{\gamma}_0, \dots, \tilde{\gamma}_i$ 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;
 - $\tilde{\gamma}_{i+1}, \dots, \tilde{\gamma}_{i+p(n)}$ are completely unspecified (i.e., they consist of wildcards only).

Note that the partial run $\tilde{\gamma}_0, \dots, \tilde{\gamma}_{i+p(n)}$ can be 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 $p(n)$ wildcard symbols.

‘ $\text{RF}(M_H)$ is not NP-complete’: Suppose, to the contrary, that $\text{RF}(M_H)$ is NP-complete. Then there is a polynomial-time (many-one) reduction f from SAT to $\text{RF}(M_H)$. 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 $\text{PTIME} \neq \text{NP}$.

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

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

for some partial run

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

of M_H . Moreover, $x \in \text{SAT}$ 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) $x \in \text{SAT}$;
- (2) $\tilde{\gamma}_0 \# \tilde{\gamma}_1 \# \cdots \# \tilde{\gamma}_m \in \text{RF}(M_H)$;
- (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 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

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

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 $\tilde{\gamma}$.

Let us take a closer look at $\tilde{\gamma}$. Let $i \geq 0$ be such that $\tilde{\gamma}_0, \dots, \tilde{\gamma}_i$ corresponds to the first step of the initialization phase of M_H on input $1^{n^{H(n)}}$. In particular, for each $j \in \{0, 1, \dots, i\}$, $\tilde{\gamma}_j$ 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) $|\tilde{\gamma}_j| \leq n$ for all $j \in \{i+1, \dots, m\}$;
- (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 $\tilde{\gamma}_{i+1} \# \cdots \# \tilde{\gamma}_m$, 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 $\tilde{\gamma}_{i+1}$, and that matches $\tilde{\gamma}_{i+1}, \dots, \tilde{\gamma}_m$. Let

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

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

$$|\phi_n| \leq |h(\tilde{\gamma}_{i+1}\#\dots\#\tilde{\gamma}_m)| \leq p(|\tilde{\gamma}_{i+1}\#\dots\#\tilde{\gamma}_m|) \leq p((m - i) \cdot (n + 1)) \leq p((p(n) + n) \cdot (n + 1)).$$

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.⁵

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

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

Now, since $\lim_{\ell \rightarrow \infty} H(\ell) = \infty$ (by Lemma G.1), we have $\lim_{\ell \rightarrow \infty} cH(\ell^c) = \infty$. 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)). \square

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) $(r(|x|) - 2 - k)^{H((r(|x|) - 2 - k)^{\frac{1}{k}})/k} > p(|x|)$; this is possible since $\lim_{\ell \rightarrow \infty} H(\ell) = \infty$ by Lemma G.1.

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

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

⁵Set $r(\ell) := \ell^{1-c}$ 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 $|x| \geq \lambda_c$.

This implies

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

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

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

which is the desired contradiction. \square