Inference from Visible Information and Background Knowledge

MICHAEL BENEDIKT, University of Oxford
PIERRE BOURHIS, CNRS, CRIStAL
BALDER TEN CATE, Google
GABRIELED PUPPIS, University of Udine
MICHAEL VANDEN BOOM, d'Overbroeck's College

We provide a wide-ranging study of the scenario where a subset of the relations in a relational vocabulary is visible to a user—that is, their complete contents are known—while the remaining relations are invisible. We also have a background theory—invariants given by logical sentences—that may relate the visible relations to invisible ones, and also may constrain both the visible and invisible relations in isolation. We want to determine whether some other information, given as a positive existential formula, can be inferred using only the visible information and the background theory. This formula whose inference we are concerned with is denoted as the *query*. We consider whether positive information about the query can be inferred, and also whether negative information—the sentence does not hold—can be inferred. We further consider both the instance-level version of the problem, where both the query and the visible instance are given, and the schema-level version, where we want to know whether truth or falsity of the query can be inferred in *some* instance of the schema.

CCS Concepts: • **Information systems → Relational database query languages**; • **Theory of computation → Problems, reductions and completeness**

Additional Key Words and Phrases: Query answering, GNF

1 INTRODUCTION

This article concerns a scenario in which a user has access to only a subset of the relations from a data source. For example, for privacy reasons, a data owner may restrict access to a subset of the stored relations. Another use case arises in information integration, where the integrated schema exposed to users may contain "virtual" relations, whose content is defined through logical specifications involving the relations in the source schemas. More generally, we consider the situation where there is a *relational schema* consisting of visible and/or hidden relations, and a *background theory* (specified by constraints in some logical language). A basic computational problem is to determine what queries can be answered (in the positive or in the negative) with access to the visible relations and reasoning using the background theory. The following example (from logical analysis of information disclosure, in the spirit of prior works such as that of Guarnieri and Basin [36]) illustrates this.

Example 1.1. Consider a medical datasource with relation Appointment(p, a, d) containing patient names p, appointment ids a, and doctor names d. A data owner may provide access only to the projection of Appointment by creating a relation Patient(p), defined by the following logical sentences Σ:

$$\forall p \; \text{Patient}(p) \; \rightarrow \; \exists a \; d \; \text{Appointment}(p, a, d)$$
$$\forall p \; a \; d \; \text{Appointment}(p, a, d) \; \rightarrow \; \text{Patient}(p).$$

Consider the Boolean query $Q = \exists a \; \text{Appointment}(\text{"Smith"}, a, \text{"Jones"})$, which asks whether patient Smith has an appointment with Dr. Jones. A user with access only to the Patient relation can never be sure that the query is true, in any instance. We say that there is no **Positive Query Implication (PQI)** in any instance. However, there exist instances where a user with access to Patient can infer that Q is false. Indeed, if the Patient relation does not contain "Smith," then Q is false in the database instance in question. We say that there is a **Negative Query Implication (NQI)** in such cases.

A second motivating example relates to *incomplete databases*.

Example 1.2. In many situations, data sources are inherently incomplete but may nevertheless be known to be partially complete. For example, consider an enterprise setting where we have a relation sales(p, y, c, q), where p is a product id, y is a year, c is a country, and q is a quantity, and this relation is known to be complete only for data with $c = $ "US". This can be specified formally as follows:

$$\forall p \; y \; c \; q \; \text{sales}(p, y, c, q) \; \rightarrow \; \text{sales}_{rw}(p, y, c, q)$$
$$\forall p \; y \; c \; q \; \text{sales}_{rw}(p, y, c, q) \wedge (c = \text{"US"}) \; \rightarrow \; \text{sales}(p, y, c, q),$$

where sales$_{rw}$ is a hidden relation that represents the complete ("real-world") sales relation. Consider the Boolean query $Q = \exists p \; q \; \text{sales}_{rw}(p, 2019, \text{"US"}, q)$, expressing our interesting in knowing whether (in the real world) any product was sold in 2019 in the United States. It is easy to see informally that, under the above background theory and the assumption that sales is visible, the answer to Q can be obtained from the sales relation. In particular, there are database instances where we have a PQI: a user with access only to sales can be sure that Q is true. And there are also instances where we have an NQI. Indeed, every instance will have one or the other, depending on whether Q held in sales.

More generally, a wide variety of partial-completeness conditions can be expressed using pairs of constraints of the form $\forall \bar{x} \; R(\bar{x}) \; \rightarrow \; R_{rw}(\bar{x})$ and $\forall \bar{x} \; R_{rw}(\bar{x}) \wedge \alpha(\bar{x}) \; \rightarrow \; R(\bar{x})$, where $\alpha(\bar{x})$ is some side condition. One of the main constraint languages that we will be studying in this ar-

Background Theory Σ	PQI data complexity	PQI combined complexity	$\exists$PQI
IDs	ExpTime-cmp Thm. 4.4 / Cor 4.7	2ExpTime-cmp Thm. 4.1 / Thm 4.8 (See also [47])	PSpace-cmp without constants Thm. 4.20
Linear TGDs	ExpTime-cmp Thm. 4.4 / Cor 4.7	2ExpTime-cmp Thm. 4.1 / Thm 4.8	ExpTime-cmp without constants Thm. 4.21 [10]; undecidable with constants Thm. 4.19
Conn. FGTGDs & FGTGDs	ExpTime-cmp Thm. 4.4 / Cor 4.7	2ExpTime-cmp Thm. 4.1 / Thm 4.8	2ExpTime-cmp without constants Cor. 4.13 / Cor 4.15; undecidable with constants Thm. 4.19
Disj Linear TGDs & GNFO	ExpTime-cmp Thm. 4.4 / Cor 4.7	2ExpTime-cmp Thm. 4.1 / Thm 4.8	undecidable Thm. 4.16

All upper bounds hold for Boolean UCQs. All lower bounds hold for Boolean CQs, with the exception of the undecidability results (Thm. 4.19 and Thm. 4.16) which rely on UCQs. All results, unless stated otherwise, hold with and without constants.

Fig. 1. Summary of complexity results for PQI.

ticle, **Frontier-Guarded Tuple-Generating Dependencies (FGTGDs)**, subsumes many such partial-completeness conditions.

The classic setting of **Open-World Query Answering (OWQ)** can also be viewed as a special case of our framework, where we again have constraints of the form $\forall \bar{x} \, R(\bar{x}) \rightarrow R_{rw}(\bar{x})$ but no constraints in the opposite direction, whereas the background theory may include additional constraints over the hidden R_{rw} relations.

We study the computational problem of testing for PQI and NQI, both at the *instance level* (in a given instance, can a positive, respectively, a negative answer, to a query Q be inferred from the visible data using the background theory?) and at the *schema level* (does there exist an instance in which a positive, respectively, a negative, answer to Q can be inferred?). The schema-level problems are motivated from schema-design considerations: they can help understand (e.g., during the data-access API design phase) what the consequences are of providing users with the means to access different relations. Variants of the schema-level problems defined here occur in prior work [45], as well as in subsequent works [10, 11, 16]: see Section 2 for details.

We consider background theories specified in a variety of logical languages that are rich enough to capture complex relationships between relations, including relationships that arise in information integration and restrictions on a single source that have been studied in the database research community ("integrity constraints") (cf. Figure 3). At the query side, we consider **Conjunctive Queries (CQs)** and **Unions of Conjunctive Queries (UCQs)**.

Our results. As mentioned earlier, we consider both the instance-level problems (denoted PQI and NQI) and the schema-level problem (denoted $\exists$PQI and $\exists$NQI). For the instance-level problem, we study both *data complexity* and *combined complexity*. Our main results are summarized in Figures 1 and 2. They apply to Boolean UCQs that may contain constants. The extension to arbitrary k-ary UCQs ($k > 0$) is discussed in Section 6, where we also discuss some richer query languages and constraint languages (e.g., CQ view definitions).

Our results in Figures 1 and 2 show how different syntactic features of the constraint language and the query language (e.g., the use of constants, connectedness, and disjunction) affect the decidability and complexity of the decision problems. They also show that, even for simple background theories, the PQI problem is already ExpTime-hard in *data complexity*—that is, when everything except the visible instance is fixed. This is a big jump in complexity compared to the special cases of the problem studied in the description logic [31] and database community [1] in the past. For

background theory Σ	NQI data complexity	NQI combined complexity	∃NQI
IDs	In PTime Cor. 5.12	ExpTime-cmp Cor. 5.12 / Thm. 5.14	In PTime without constants Thm. 5.19
Linear TGDs	In PTime Cor. 5.12	ExpTime-cmp Cor. 5.12 / Thm. 5.14	PSpace-cmp without constants Cor. 5.18 / Thm. 5.21
Conn. FGTGDs & Disj. Conn. FGTGDs	ExpTime-cmp Thm. 5.1 / Cor. 5.3	2ExpTime-cmp Thm. 5.1 / Cor. 5.4	2ExpTime-cmp without constants Cor. 5.16 / Thm. 5.20; undecidable with constants Thm. 5.25
FGTGDs & GNFO	ExpTime-cmp Thm. 5.1 / Cor. 5.3	2ExpTime-cmp Thm. 5.1 / Cor. 5.4	undecidable Thm. 5.24

All upper bounds all holds for Boolean UCQs. All lower bounds hold for Boolean CQs, with the exception of Theorem 5.14. All results, unless stated otherwise, hold with and without constants.

Fig. 2. Summary of complexity results for NQI.

example, in the work of Abiteboul and Duschka [1], it is shown that the data complexity of the OWQ problem (which, we recall, can be seen as a particular instance of PQI with **Inclusion Dependencies (IDs)** that go from visible to hidden relations) is shown to be in co-NP, and Franconi et al. [31] proveco-NP-completeness of query answering in certain description logics with DBoxes.

Our techniques. We develop a number of tools for reasoning on mixtures of complete and incomplete information:

- ***Connection to guarded negation***: Our first technique involves showing that a large class of instance-level problems can be solved by translating them into satisfiability problems for a rich fragment of first-order logic, the guarded negation fragment (GNFO). In fact, we show that there is a natural connection between these inference problems and GNFO, in that the "visibility restriction" can be expressed in GNFO. This allows us to exploit powerful prior decidability results for GNFO "off-the-shelf." However, to get tight complexity bounds, we need a new analysis of the complexity of satisfiability for GNFO. This analysis is of interest outside of these inference problems, in that we give a self-contained reduction from GNFO satisfiability to tree automata, a reduction that allows us to give a finer-grained analysis of the sources of complexity in GNFO satisfiability.
- ***Decidability via canonical counterexamples***: The schema-level analysis asks if there is some instance on which information about the query can be derived. As mentioned earlier, we show that whenever there is some instance, this can be taken to be the "simplest possible instance." Although this idea has been used before to simplify analysis of undecidability (e.g., [32]), and for decidability of Datalog satisfiability [49], we provide a significant extension of the technique, and provide new applications of it for decidability.
- ***Tractability via greatest fixed-point***: We show that some of our instance-level implication problems can be reduced to evaluating a certain query of **Greatest Fixedpoint Datalog (GFP-Datalog)** on the given visible instance. Since GFP-Datalog queries can be evaluated in polynomial time, this shows tractability in the instance size. The reduction to GFP-Datalog requires a new analysis of when these inference problems are "active-domain controllable" (it suffices to see that the query value is invariant over all hidden instances that lie within the active domain of the visible instance).
- ***Relationships between problems***: In this article, we explain how the four inference problems we consider (combinations of positive/negative and instance level/schema level) differ from previously studied problems, such as the OWQ problem. However, we also pro-

vide reductions between open world querying and some of our schema-level problems. In addition to clarifying the relationship of the problems, we can use these reductions to derive complexity bounds.

Organization. After a review of related work in Section 2, we formally define the problem in Section 3. Section 4 presents our results on the PQI problems PQI and ∃PQI (cf. Figure 1), whereas Section 5 presents our results for the NQI problems NQI and ∃NQI. In these two sections, we restrict attention to Boolean queries. Section 6 shows how to extend the results to arbitrary, non-Boolean queries, and it also discusses some further extensions and variants of the framework, including background theories specified by CQ view definitions. We close in Section 7 with conclusions.

2 RELATED WORK

Two different communities have studied the problem of determining which information can be inferred from complete access to data in a subset of the relations, using background knowledge in the form of logical sentences relating the subset to the full vocabulary.

In the database community, the focus has been on views. The schema is divided into the "base tables" and "view tables," with the latter being defined by queries (typically CQs) in terms of the former. Given a query over the schema, the basic computational problem is determining which answers can be inferred using only the values of the views. Abiteboul and Duschka [1] isolate the complexity of this problem in the case where views are defined by CQs; in their terminology, it is "querying under the Closed World Assumption," emphasizing the fact that the possible worlds revealed by the views are those where the view tables have exactly their visible content. In our terminology, this corresponds exactly to the PQI problem in the case where the background theory consists entirely of CQ view definitions. Chirkova and Yu [26] extend to the case where CQ views are supplemented by weakly acyclic dependencies. Another subcase of PQI that has received considerable attention is the case where the background theory consists *only* of "completeness assertions" between the invisible and visible portions of the schema. A series of works by Fan and Geerts [29, 30] isolate the complexity for several variations of the problem, with particular attention to the case where the completeness assertions are via IDs from the invisible to the visible part.

The PQI problem we study in the first part of this work is also related to research on instance-based determinacy (in particular, see the results of Koutris et al. [38]), whereas the NQI problem in the second half of the article is examined in the view context by Zhang and Mendelzon [51], under the name of "conditional emptiness." As in the other work mentioned earlier, the emphasis has been on view definitions rather than more general background knowledge that may restrict both the visible and invisible instance. In contrast, in our work, we deal with logical languages for the background theory that can restrict the visible and invisible data in ways incomparable to view definitions (see also the comparison in Section 6).

In the description logic community, the emphasis has not been on views but on querying incomplete information in the presence of a logical theory. Our PQI problems relate to work in the description logic community on *hybrid closed- and open-world query answering* or *DBoxes*, in which the schema is divided into closed-world and open-world relations. Given a Boolean CQ, we want to find out if it holds in all instances that can add facts to the open-world relations but do not change the closed-world relations. In the non-Boolean case, the generalization is to consider which tuples from the initial instance are in the query answer on all such instances. Thus, closed-world and open-world relations match our notion of visible and invisible, and the hybrid closed and OWQ problem matches our notion of PQI, except we restrict to the case where the open-world

relations of the instance are empty. It is easy to see that this restriction is actually without loss of generality: one can reduce the general case to the case we study with a simple linear time reduction, making a closed-world copy R' of each open-world relation R, and adding an ID from R' to R. As with the database community, the main distinction between our study of the PQI problem and the prior work in the DL community concerns the classes of background theories considered. Lutz et al. [39–41] study the complexity of this problem for background knowledge for several description logics. For example, for the description logics $\mathcal{EL}$ and DL-LITE, they provide a dichotomy between CO-NP-hard and first-order rewritable theories. They also show that in all of the tractable cases, the problem coincides with the classical OWQ problem. Franconi et al. [31] show CO-NP-completeness for a disjunction-free description logic. Our results on the data complexity of PQI consider the same problem, although for background theories that are more expressive and, in particular, can handle relations of arbitrary arity rather than arity at most 2 as in other works [31, 40, 41].

In summary, both the database and DL communities considered the PQI questions addressed in this article, although for background theories that are different from those we consider. However, in some cases, we can infer relevant lower bounds for our problems from those in the DL or DB community: in particular, see Theorem 4.8, a variant of a result from Ngo et al. [47]. The NQI problems are not well studied in the prior literature. However, a special case of NQI is the problem of "consistency" in the setting of hybrid open- and closed-world querying. This is what we call *realizability*.[1] Bounds on the combined complexity of this problem can be found in the work of Ngo et al. [47], but they are for description logics that are orthogonal in expressiveness to the logics we study.

We know of no work dealing with the schema-level questions (asking for the existence of an instance with a query implication) prior to the publication of the conference version of this article. Schema-level problems are mentioned in a work by Nash and Deutsch [45], who appear to define a variant of our ∃PQI problem ("source-independent guarantees"). But the only results given in the work (Section 5.4 of Nash and Deutsch [45]) refer to the opposite problem, a variant of determinacy. However, in this article, we show (see Section 5.2) that there is a close relation between these schema-level questions and the works of Lutz et al. that concern conservativity and modularity of ontologies [37, 42].

Note that our schema-level analysis considers the existence of *some* instance where the query result can be inferred. The converse problem is to determine whether the query result can be inferred on *all* instances. This is exactly the problem of *determinacy* [46], which is closely related to the notion of *implicit definability* in classical logic [21]. Determinacy has been extensively studied for both views [33, 46] and for background theories and visible relations [18, 19].

Another contrast is to the work of Miklau and Suciu [44], which considers whether such an inference is valid probabilistically, looking asymptotically at the uniform distribution over models of increasing size.

Subsequent to the conference version of this article, there have been a number of follow-ups. Two works [15, 16] analyzed the complexity of query implication in the presence of information disclosure methods based on query answering interfaces—where an external user can query under the certain answer semantics—rather than the model of disclosure based on exporting a subset of the data, as in our setting. The analysis in these works build on the techniques presented in this submission. For example, another work [16] proves an undecidability result for our static disclosure problem in the case of background theories corresponding to UCQ views; see the comments prior

---

[1]More precisely, realizability can be viewed as a trivial case of NQI, where the query is the tautology ⊤. Note, however, that all of our lower bounds results for NQI in this article hold even when the input is required to be a realizable instance.

to Theorem 6.2 for details. Yet another work [10] refines the complexity analysis presented in this work in some special cases of interest in data integration: the background theory can only consist of certain restricted sentences on the invisible relations and some simple "mappings" between invisible and visible. In the process, that work [10] corrects an error from the conference version of this work, as discussed in Theorem 4.21. The idea of using the critical instance as a technique for optimizing static analysis, which we highlight in this submission, has been picked up by later work in a different context (simplifying Datalog rules) in the work [17]. Another work [11] applies the results here in analyzing trade-offs between privacy and utility.

3 DEFINITIONS

Schemas and instances. We consider partitioned schemas (or simply, schemas) $S = S_h \cup S_v$, where the partition elements S_h and S_v are finite sets of relation names (or simply, relations), where each relation R has an associated arity, denoted $\text{arity}(R)$. These are the *hidden* and *visible* relations, respectively. An *instance* of a schema maps each relation to a set of tuples of the associated arity. Throughout this article, instances are assumed to be finite unless explicitly specified otherwise. The *active domain* of an instance is the set of values occurring within the interpretation of some relation in the instance. A *fact* over an instance $\mathcal{I}$ is a ground atom $R(c_1 \dots c_n)$, where R has arity n and tuple $c_1 \dots c_n$ are in the set associated by $\mathcal{I}$ to R.

As a suggestive notation, we write $\mathcal{V}$ (Visible) for instances over S_v and $\mathcal{F}$ (Full) for instances over S, and we will use the term *full instance* to refer to an instance over the full schema S. Given such an instance $\mathcal{F}$ for S, its restriction to the S_v relations will be referred to as its *visible part*, denoted $\text{Visible}(\mathcal{F})$.

CQs; unique name assumption. In this work, we will consider queries specified as CQs—first-order formulas built up from relational atoms via conjunction and existential quantification (equivalently, relational algebra queries built via selection, projection, join, and rename operations)—and also UCQs, which are disjunctions (relational algebra unions) of CQs. *Boolean* (U)CQs are simply (U)CQs with no free variables.

Unless specified otherwise, we allow constants in queries (as well as in background theories, defined below). More precisely, we assume that we have associated with each value a corresponding constant, and we will identify the constant with its value. Thus, distinct constants will always be forced to denote distinct domain elements—this is often called the **Unique Name Assumption (UNA)** [2].

Every CQ Q is associated with a *canonical instance* $\text{CanonInst}(Q)$, where the domain consists of variables and constants of Q and the facts are the atoms of Q.

Background theories. We will look at background theories defined in a number of constraint languages (cf. Figure 3). One class of constraints that we will focus on are **Tuple-Generating Dependencies (TGDs)**, which are first-order logic sentences of the form

$$\forall \bar{x} \; \phi(\bar{x}) \;\rightarrow\; \exists \bar{y} \; \rho(\bar{x}, \bar{y}),$$

where ϕ and ρ are conjunctions of atoms, which may contain variables and/or constants, and where all universally quantified variables $\bar{x}$ appear in $\phi(\bar{x})$. For all of the problems considered in this work, one can take without loss of generality the right-hand side ρ to consist of a single atom, and we will assume this henceforth. For brevity, we will often omit the universal quantifiers in TGDs: writing just $\phi(\bar{x}) \;\rightarrow\; \exists \bar{y} \; \rho(\bar{x}, \bar{y})$. TGDs form an important and well-studied class of database constraints. However, most inference problems involving TGDs are undecidable [2], including those that we study here. To obtain decidability and complexity results, we will look at classes of TGDs that are computationally better behaved:
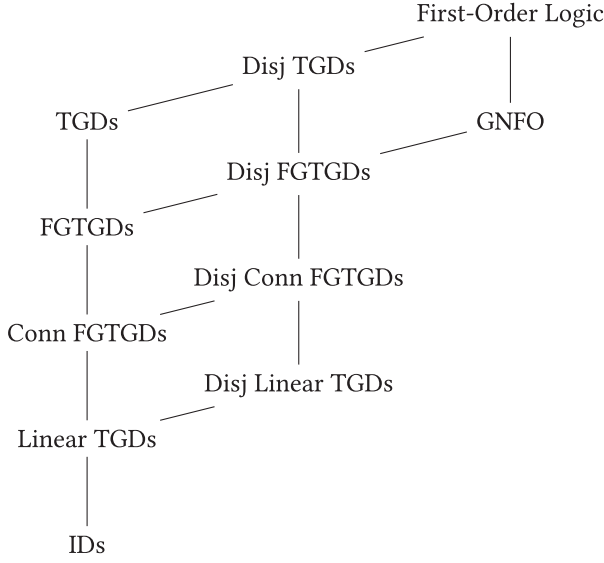
Fig. 3. Constraint languages for specifying background theories.

- *Linear TGDs*, those where ϕ consists of a single atom.
- *IDs*, linear TGDs where each of ϕ and ρ have no constants and no repeated variables. These correspond to traditional referential integrity constraints in databases.
- *FGTGDs* [4] are TGDs where one of the conjuncts of ϕ is an atom that includes every universally quantified variable x_i occurring in ρ. Such variables are the *exported variables* of the TGD.
- *Connected TGDs* require only that the *co-occurrence graph* of ϕ is connected. The nodes of this graph are the variables $\bar{x}$, and variables are connected by an edge if they co-occur in an atom of ϕ.

Note that every ID is a linear TGD, and every linear TGD is frontier guarded. We will also consider two logical languages that are generalizations of the above:

- We allow disjunction, by considering *disjunctive TGDs*, which are of the form

$$\forall \bar{x} \, \phi(\bar{x}) \; \rightarrow \; \exists \bar{y} \bigvee_i \rho_i(\bar{x}, \bar{y}),$$

where each ρ_i is a conjunction of atoms. The notion of a disjunctive TGD being *connected* is the same as for TGDs above. A *disjunctive FGTGD* additionally requires that there is an atom in ϕ that includes all variables x_j occurring in any of the ρ_i's. Some of our negative results will hold for a special case of disjunctive FGTGDs in which ϕ is a single atom; these are *disjunctive linear TGDs*, which clearly subsume linear TGDs.

Note that in each of the above cases, with the exception of IDs, constants are allowed by default (and, as we already pointed out earlier, constants are assumed to satisfy the UNA).

A key role will be played by an even richer logic, containing disjunctive FGTGDs, the *guarded negation fragment*. GNFO is built up inductively according to the grammar:

$$\phi \; ::= \; R(\bar{t}) \mid t_1 = t_2 \mid \exists x \, \phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \alpha(\bar{y}) \wedge \neg\phi(\bar{y}),$$

where R is a relation symbol, each t_i is a variable or constant, and $\alpha(\bar{y})$ is an atomic formula (which we will refer to as a guard), in which all variables in $\bar{y}$ occur. The name *guarded negation fragment*

reflects the fact that any use of negation must occur in conjunction with a guard. Note, however, that if $\bar{y}$ consists of a single variable y, then the trivial equality $y = y$ is a possible such guard (and therefore unguarded negations of subformulas with at most one free variable can be expressed).

In database terms, GNFO is equivalent to relational algebra where the difference operator can only be used to subtract query results from a relation. The VLDB article [5] gives both relational algebra and SQL-based syntax for GNFO, and argues that it covers useful queries and database integrity constraints in practice.

Note that the classical semantics of a first-order logic formula requires an interpretation of each relation in the formula and also a domain, a set of elements that includes the active domain of the instance. Whenever we will speak about GNFO sentences, we will always assume that they are *domain independent*—that is, their truth value does not depend on which domain we choose (as long as it includes the active domain). This allows us to meaningfully interpret these sentences in an instance without explicitly specifying a domain. The relational algebraic presentation of GNFO mentioned earlier provides an explicit syntax for the domain-independent fragment of GNFO.

For many of the results in the article, the reader only needs to know a few facts about GNFO. The first is that it is quite expressive, so in proving things about GNFO sentences, we immediately get the results for many classes of theories that we have mentioned previously. GNFO contains every positive existential formula, is closed under Boolean combinations of sentences, and it subsumes disjunctive FGTGDs up to equivalence. In other words, by simply writing out a disjunctive FGTGD using $\exists, \neg, \wedge$, one sees that these are expressible in GNFO.

Second, we will use that GNFO is "tame," encapsulated in the following result from Bárány et al. [6].

THEOREM 3.1 ([6]). *Satisfiability for GNFO sentences can be tested effectively and is* 2ExpTime-*complete. Furthermore, every satisfiable sentence has a finite satisfying model.*

Instance-level PQI and NQI. A fundamental definition for our work is presented next.

Definition 3.2. Let Q be a Boolean UCQ over schema S, Σ a background theory over S (specified in any of the constraint languages introduced earlier), and $\mathcal{V}$ an instance over a visible schema $S_v \subseteq S$:
- PQI$(Q, \Sigma, S, \mathcal{V})$ = true if for every finite instance $\mathcal{F}$ satisfying Σ, if $\mathcal{V} = \text{Visible}(\mathcal{F})$ then $Q(\mathcal{F})$ = true.
- NQI$(Q, \Sigma, S, \mathcal{V})$ = true if for every finite instance $\mathcal{F}$ satisfying Σ, if $\mathcal{V} = \text{Visible}(\mathcal{F})$ then $Q(\mathcal{F})$ = false.

The above definition gives rise to decision problems: given Q, Σ, S, and $\mathcal{V}$, decide if PQI$(Q, \Sigma, S, \mathcal{V})$ = true (respectively, if NQI$(Q, \Sigma, S, \mathcal{V})$ = true).

We call an S_v-instance $\mathcal{V}$ *realizable* with respect to Σ if there is an S-instance $\mathcal{F}$ satisfying Σ such that $\mathcal{V} = \text{Visible}(\mathcal{F})$. If an instance $\mathcal{V}$ is not realizable with respect to Σ, then, trivially, PQI$(Q, \Sigma, S, \mathcal{V})$ = NQI$(Q, \Sigma, S, \mathcal{V})$ = true. In practice, realizable instances are the only S_v-instances we should ever encounter. When studying the PQI and NQI decision problems, for simplicity we assume that the input is an arbitrary, not-necessarily-realizable instance of S_v. However, our lower bound arguments will only involve realizable instances, and therefore, an alternative definition that assumes realizable inputs yields the same complexity bounds.

The definition of PQI$(Q, \Sigma, S, \mathcal{V})$ involves a quantification over finite instance, in line with our default assumption that instances are finite. We can also talk about an "unrestricted version" where the quantification is over every (finite or infinite) instance. We denote this variant by PQI$_\infty(Q, \Sigma, S, \mathcal{V})$. As we will see, for most of the background theories we consider, PQI and

PQI$_\infty$ turn out to coincide. When this holds, we say PQI is *finitely controllable* for the background theories in question. The same applies to NQI.

We need a definition of the size of the input. In our case, an input consists of a query Q, a set of sentences Σ, a relational schema $\mathbf{S}$, and an instance $\mathcal{V}$, and the size is defined by taking the length of the binary encoding of such objects. Other intuitive notions of size (e.g., number of symbols) would also suffice for our results, since they differ from the bit-encoding notion only up to a polynomial factor.

Often we will be interested in studying the behavior of the PQI and NQI problems when Q, Σ, and $\mathbf{S}$ are fixed (e.g., looking at how the computation time varies in the size of $\mathcal{V}$ only). We refer to this as the *data complexity* of the PQI (respectively, NQI) problem.

The PQI problem contrasts with the usual OWQ problem (a.k.a., the *Certain Answer* problem), denoted here OWQ$(Q, \Sigma, \mathcal{F})$, which is studied extensively in databases and description logics. The latter problem takes as input a Boolean query Q, an instance I, and a set of sentences Σ, and returns true if and only if the query holds in any finite instance I' containing all facts of I. In PQI (and NQI), we further constrain the instance to be fixed on the visible part while requiring the invisible part of the input instance to be empty. This is the mix of closed world and open world, and we will see that this closed-world restriction can make the complexity significantly higher.

Example 3.3. Consider a scenario where the background theory consists of IDs $F_1(x) \rightarrow \exists y\, U(x, y)$ and $U(x, y) \rightarrow F_2(y)$. In the schema, the relations F_1 and F_2 are visible but U is not. Consider the query $Q = \exists x\, U(x, x)$ and instance consisting only of facts $F_1(a)$, $F_2(a)$.

There is a PQI on this instance, since $F_1(a)$ implies that $U(a, c)$ holds for some c, but the other constraint and the fact that F_2 must hold only on a means that $c = a$, and hence Q holds.

In contrast, one can easily see that Q is not certain in the usual sense, where F_1 and F_2 can be freely extended with additional facts.

Since the PQI generalizes OWQ, by allowing us to constrain part of the schema, we clearly have a reduction from OWQ to PQI. As we will see later (Theorem 5.2), there is a further reduction from PQI to NQI. Since OWQ is undecidable for Boolean CQs under TGDs (e.g., [9]) this means the following.

PROPOSITION 3.4. *Both* PQI$(Q, \Sigma, \mathbf{S}, \mathcal{V})$ *and* NQI$(Q, \Sigma, \mathbf{S}, \mathcal{V})$ *are undecidable when Q ranges over Boolean CQs and Σ ranges over TGDs.*

For this reason, in most of this article, we focus on more restricted classes of (disjunctive) TGDs, as well as on GNFO (cf. Figure 3).

Schema-level PQI and NQI. Our schema-level problems ask if there is a realizable instance that admits a query implication.

Definition 3.5. Let Q be a Boolean CQ over schema $\mathbf{S}$ and Σ a background theory over $\mathbf{S}$:
- $\exists$PQI$(Q, \Sigma, \mathbf{S})$ = true if there is a realizable finite $\mathbf{S}_v$-instance $\mathcal{V}$ such that PQI$(Q, \Sigma, \mathbf{S}, \mathcal{V})$ = true.
- $\exists$NQI$(Q, \Sigma, \mathbf{S})$ = true if there is a realizable finite $\mathbf{S}_v$-instance $\mathcal{V}$ such that NQI$(Q, \Sigma, \mathbf{S}, \mathcal{V})$ = true.

Note that these problems now quantify over instances twice, and hence there are alternatives depending on whether the instance $\mathcal{V}$ is restricted to be finite, and whether the hidden instances $\mathcal{F}$ are restricted to be finite. We denote by $\exists$PQI$_\infty$ and $\exists$NQI$_\infty$ the variants of $\exists$PQI and $\exists$NQI where all quantification is over possibly infinite instances. In addition, for a class of input Q, Σ, $\mathbf{S}$, we say that "$\exists$PQI$(Q, \Sigma, \mathbf{S})$ is *finitely controllable*" if each of the two quantifications can be freely replaced with quantification over arbitrary instances without changing the truth value of the statement.

4 POSITIVE QUERY IMPLICATION

4.1 The Instance-Level Problem PQI

Here we study the (instance-level) problem $\text{PQI}(Q, \Sigma, S, \mathcal{V})$. Recall that this asks whether $Q(\mathcal{F}) =$ true for every full instance $\mathcal{F}$ satisfying Σ that agrees with $\mathcal{V}$ in the visible part. The section is organized in two parts. In the first part, we prove upper bounds for the (instance-level) PQI problem, establishing a connection to the guarded negation fragment. In the second part, we present matching lower bounds.

Upper bounds and the connection to guarded negation. We begin by showing that PQI is decidable when background theories are in the logic GNFO, the guarded negation fragment of first-order logic. This is interesting first of all because GNFO is a very expressive logic. It subsumes the other decidable logics that we consider here, such as guarded TGDs, disjunctive guarded TGDs, and Boolean combinations of Boolean CQs. Further, it highlights the fact that GNFO can express that an instance has a particular restriction to the visible relations. This is exploited in the following reduction to the satisfiability problem for GNFO.

THEOREM 4.1. *The problem* $\text{PQI}(Q, \Sigma, S, \mathcal{V})$, *as Q ranges over Boolean UCQs and Σ ranges over GNFO sentences, is in* 2EXPTIME. *Furthermore, for such Q and Σ, the problem is finitely controllable—that is,* $\text{PQI}(Q, \Sigma, S, \mathcal{V}) = \text{PQI}_\infty(Q, \Sigma, S, \mathcal{V})$.

PROOF. One easily sees that $\text{PQI}(Q, \Sigma, S, \mathcal{V})$ translates to unsatisfiability of the following formula:

$$\phi_{Q,\Sigma,S,\mathcal{V}}^{\text{PQItoGNF}} = \neg Q \;\wedge\; \Sigma \;\wedge\; \bigwedge_{R \in S_v} \left(\bigwedge_{R(\bar{a}) \in \mathcal{V}} R(\bar{a}) \;\wedge\; \forall \bar{x} \left(R(\bar{x}) \rightarrow \bigvee_{R(\bar{a}) \in \mathcal{V}} \bar{x} = \bar{a} \right) \right).$$

Intuitively, the formula requires that the instance on which it is evaluated (which includes visible and hidden relations) satisfies the background theory, but not the query, and in addition, the visible part of the instance agrees with $\mathcal{V}$. Note that the formula has size linear in the inputs to PQI, and thus this gives a polynomial time reduction.

Note that the third conjunct of $\phi_{Q,\Sigma,S,\mathcal{V}}^{\text{PQItoGNF}}$ is a conjunction of disjunctive **Equality-Generating Dependencies (EGDs)**. Although this is not a well-studied type of constraints, they have arisen in the past as a technical tool in the study of OWQ for CQs with inequalities [28].

If the background theory consists of GNFO sentences, then the formula above is also in GNFO. Indeed, the only places where negation is used, either explicitly or implicitly, are $\neg Q$, which is guarded since Q has no free variables, and the universal quantification $\forall \bar{x} (R(\bar{x}) \rightarrow \ldots)$, which translates to $\neg \exists \bar{x} (R(\bar{x}) \wedge \neg \ldots)$, with the inner negation guarded by $R(\bar{x})$ and the outer negation involving no free variables.

The finite controllability of $\text{PQI}(Q, \Sigma, S, \mathcal{V})$ comes from the finite model property of GNFO (Theorem 3.1). □

Above, we are using results on satisfiability of GNFO as a "black box." Satisfiability tests for GNFO work by translating a satisfiability problem for a formula into a tree automaton, which is then tested for non-emptiness. By a finer analysis of this translation of GNFO formulas to automata, we can show that the *data complexity* of the problem is only singly exponential.

We start by introducing a *normal form* for GNFO formulas, similar to the one introduced by Bárány et al. [7]. Formulas in such a normal form are generated using the following grammar:

$$\phi ::= \bigvee_i \exists \bar{x}_i \; \bigwedge_j \psi_{ij}$$
$$\psi ::= \alpha \;\mid\; \alpha \wedge \neg \phi,$$

where α is an atomic formula and free variables of ϕ are contained in free variables of α. As with GNFO, in the second production rule, we also allow α to be omitted if ϕ has at most one free variable x (thus allowing free negation of such formulas—note that such formulas can always be trivially guarded by the equality atom $x = x$). The ϕ's are referred to as *UCQ-shaped formulas*, with each of the disjuncts being a *CQ-shaped formula*. UCQ-shaped formulas are only used to define the normal form and the related notion of CQ-rank below.

As the name implies, every GNFO formula can be converted to an equivalent one in normal form.

PROPOSITION 4.2. *There is an* EXPTIME *procedure taking as input a GNFO formula and outputting a ϕ' in normal form that is equivalent to ϕ.*

This is a variant of a simple normalization procedure from Bárány et al. [7]; details are given in the appendix.

The *CQ-rank* of a GNFO formula ϕ in normal form, denoted rank(ϕ), is the maximum number of conjuncts ψ_i in any CQ-shaped subformula $\exists \bar{x}_i \bigwedge_i \psi_i$ of ϕ, for non-empty $\bar{x}_i$. For the purposes of CQ-rank, $\alpha \wedge \neg\phi$ is treated as a CQ-shaped subformula with one conjunct. The *width* of ϕ, denoted width(ϕ), is the maximum number of free variables of any subformula of ϕ.

What follows is the more detailed result of GNFO satisfiability that we will rely on.

THEOREM 4.3. *For every fixed numbers r and w, there is an* EXPTIME *algorithm that determines whether a given GNFO formula ϕ in normal form, with* rank(ϕ) $\leq r$ *and* width(ϕ) $\leq w$, *is satisfiable.*

The proof, which is spelled out in the appendix, is based on a fine-grained analysis of the translation of a GNFO formula ϕ in normal form to a suitable automaton, extending the translation found in earlier work [14]. Here we only summarize at a high level. The result is proven by creating an alternating two-way parity automaton whose states are formulas derived from ϕ. The automaton runs on a tree where nodes encode width(ϕ)-sized collections of elements in a tree-like model. The automaton at a state corresponding to some formula ϕ has transitions that verify ϕ, and there will be transitions for each rule in the normal form. For Boolean connectives, the automaton makes use of alternation. For example, when checking whether a formula $\psi_1 \wedge \psi_2$ holds at the current node of the input tree, the automaton spawns two subcomputations that check, respectively, that ψ_1 and ψ_2 hold at the current node. When evaluating a formula that starts with a quantifier, the automaton searches for nodes witnessing a "specialization" of the formula, and does so by inspecting both the current node and its neighborhood. This crucially relies on the capability of the automaton of navigating the tree in any direction—that is, from a node to its parent and/or to one of its children.

The difficulty in correctly evaluating a formula by means of an automaton as above is reflected in the definition of "specialization." If the formula ϕ to be evaluated were in the *guarded fragment*, namely, the fragment of first-order logic with guarded quantification, rather than in GNFO, it would suffice to use as states the subformulas of ϕ annotated with interpretations of the free variables. In particular, a possible specialization of a quantified formula like $\exists x \, \varphi(x)$ from the guarded fragment would be the subformula φ annotated with any interpretation of the free variable x by an element of a guarded set.

For formulas of GNFO, instead, one has to throw in new subformulas. In particular, specializations of CQ-shaped formulas need to represent the possible guesses as to which of the conjuncts were true of the elements associated to a given node of a tree-like structure. The bound on rank(ϕ) guarantees that this need to throw in new subformulas does not blow up the number of states. An additional issue is that, in this setting, tree nodes will not be associated with a guarded set but with a set whose size is controlled by width(ϕ). Thus, by bounding width(ϕ), we keep the number of annotations low. See the appendix for further details.

It is also important for our application that the result applies to GNFO formulas that have equality and constants, which are treated by adding additional cases for equality atoms in the automata, and conjoining with an additional automata that enforces that the facts involving constants are consistent across the tree. The details of this, as well as other subtleties in the proof of Theorem 4.3, are given in the appendix.

We are now ready to state our data-complexity result.

THEOREM 4.4. *If Q is a Boolean UCQ and Σ is a conjunction of GNFO sentences over a schema* S, *then the data complexity of* PQI$(Q, \Sigma, S, \mathcal{V})$ *(i.e., as $\mathcal{V}$ varies over instances) is in* EXPTIME.

PROOF. Fix a Boolean UCQ Q and a conjunction Σ of GNFO sentences over a schema S. Without loss of generality, by Proposition 4.2, we can assume that the sentences in Σ are already in normal form. Consider the formula $\phi_{Q,\Sigma,S,\mathcal{V}}^{\text{PQItoGNF}}$ in the proof of Theorem 4.1:

$$\neg Q \ \wedge \ \Sigma \ \wedge \ \bigwedge_{R \in S_v} \Big(\bigwedge_{R(\bar{a}) \in \mathcal{V}} R(\bar{a}) \ \wedge \ \forall \bar{x} \ \big(R(\bar{x}) \rightarrow \bigvee_{R(\bar{a}) \in \mathcal{V}} \bar{x} = \bar{a} \big) \Big).$$

This formula can be rewritten to eliminate the universally quantified implication, replacing this subformula with the negation of the sentence

$$\exists \bar{x} \ R(\bar{x}) \wedge \bigwedge_{R(\bar{a}) \in \mathcal{V}} \bigvee_i x_i \neq a_i.$$

We can rewrite this to be in normal form, either by adding the relational atom to each negated equality or by just observing that the negated equalities are unary. With these changes, which do not impact the size of the formula, the conditions of the normal form are satisfied. This shows that the formula $\phi_{Q,\Sigma,S,\mathcal{V}}^{\text{PQItoGNF}}$ can be normalized in polynomial time.

Moreover, the rank and the width of the normal form of $\phi_{Q,\Sigma,S,\mathcal{V}}^{\text{PQItoGNF}}$ are bounded when Q, Σ, and S are fixed. Applying Theorem 4.3, the bound claimed in Theorem 4.4 now follows. □

Lower bounds. In the following, we show that the data complexity bound in Theorem 4.4 is tight even for IDs. The proof proceeds by showing that a "universal machine" for alternating PSPACE can be constructed by fixing appropriate Q, Σ, S in a PQI problem. We first prove the hardness result using a UCQ Q; subsequently, we show how to strengthen this to apply to a CQ.

THEOREM 4.5. *There are a Boolean UCQ Q without constants and a set Σ of IDs over a schema* S *for which the problem* PQI$(Q, \Sigma, S, \mathcal{V})$ *is* EXPTIME-*hard in data complexity.*

PROOF. We reduce the acceptance problem for an alternating PSPACE Turing machine M to the negation of PQI$(Q, \Sigma, S, \mathcal{V})$.

A configuration of M is defined, as usual, by a control state, a position of the head on the tape, and a finite string representing the content on the tape. The input of the machine is assumed to be a string of blanks $\sqcup \cdots \sqcup$ (thus, only its length matters). Moreover, special symbols $\vdash, \dashv$ are added at the extremities of the input to mark the endpoints of the working tape. Accordingly, the initial configuration of M has tape content of the form $\vdash \sqcup \cdots \sqcup \dashv$ and the head on the first position.

The transition function of M describes a set of target configurations on the basis of the current configuration. We distinguish between existential and universal control states of M, and we assume that there is a strict alternation between existential and universal states along every sequence of transitions. Without loss of generality, we also assume that there are exactly two target configurations for each transition that departs from a universal state. A computation of M is thus represented by a tree of configurations, where the root represents the initial configuration and every node with an existential (respectively, universal) control state has exactly one (respectively, two) successor

configuration(s). Furthermore, to make the coding simpler, we adopt a non-standard acceptance condition. Specifically, we assume that the Turing machine M never halts, namely, its transition function is defined on every configuration, and we distinguish two special control states, q_{acc} and q_{rej}. We further assume that every infinite path in a computation tree of M eventually reaches a configuration with either q_{acc} or q_{rej} as the control state, and from there onward there is no change of configuration. Accordingly, we say that M accepts (its input) if it admits a computation tree where the state q_{acc} appears on all paths; symmetrically, we say that M rejects if every computation tree has a path leading to q_{rej}.

The general idea of the reduction is to create a schema, background theory, and query that together represent a "universal machine" for alternating PSpace. Then, given an alternating PSpace machine M encoded in the visible instance, an accepting computation tree of M will be encoded by an arbitrary full instance that satisfies the background theory and violates the query—that is, a witness of the failure of PQI. We first devise the schema with hidden relations that will store the computation tree of a generic alternating PSpace machine. The background theory and (the negation of) the query will be used to restrict the hidden relations so as to guarantee that the encoding of the computation tree is correct. By "generic," we mean that the hidden relations and corresponding background theory will be independent of the tape size, number of control states, and transition function of the machine. The visible instance will store the "representation" of an alternating PSpace machine M—that is, an encoding of M that can be calculated efficiently once M is known. This will include the tape size and an encoding of the transition function. We will then give the reduction that takes an alternating polynomial space machine M and instantiates all visible relations with the encoding. The space bound on M will allow us to create the tape components in the visible instance efficiently. In contrast, the hidden relations will store aspects of a computation that *cannot* be computed easily from M. In summary, in the following, we will be describe each part of the schema S for computation trees of a machine, along with the polynomial mapping that transforms a machine M into data filling up the visible parts of the schema.

To begin with, we explain how to encode the tape (devoid of its content) into a binary relation T. The relation T will be visible and can be filled efficiently once the length of the tape of M is known. Given M, it will be filled in the following natural way: it contains all of the facts $T(y, y')$, where y is the identifier of a cell and y' is the identifier of the successor of this cell in the tape. Recall that the Turing machine M works on a tape of polynomial length, and hence the visible instance for the relation T has also size polynomial in M. We also add unary visible relations First and Last, which are intended to distinguish the first and last cells of the tape. Given M, we will instantiate First (respectively, Last) with the singleton consisting of the identifier of the first (respectively, last) cell. Moreover, despite the fact that the tape length is finite, it is convenient to assume that every cell has a successor—this assumption will be exploited later to ease the instantiation of new tape contents for each configuration. We will thus add to the visible relation T also the "dummy" pair (y, y), where y is the identifier of the rightmost cell of the tape.

As for the configurations of the machine, these are described by specifying, for each configuration and each tape cell, a suitable value that represents the content of that cell, together with the information on whether the Turing machine has its head on the cell, to the right, or to the left, and what is the corresponding control state. Formally, the configurations of the machine are encoded by a hidden ternary relation C, where each fact $C(x, y, z)$ indicates that, in the configuration identified by x, the cell y has value z. We will enforce that the cell values range over an appropriate domain, defined by a visible unary relation V. In our reduction from M, we will fill this relation V with $\Sigma_Q \uplus \Sigma_\triangleleft \uplus \Sigma_\triangleright$, where Σ is the tape alphabet of M (which includes the markers $\vdash$ and $\dashv$), $\Sigma_Q = \Sigma \times Q$, $\Sigma_\triangleleft = \Sigma \times \{\triangleleft\}$, $\Sigma_\triangleright = \Sigma \times \{\triangleright\}$, Q is the set of its control states, and $\triangleleft, \triangleright$ are fresh symbols.

When a cell has value (a, q), this means that its content is a, the Turing machine stores the control state q, and the head is precisely on this cell. Similarly, when a cell has value $(a, \lhd)$ (respectively, $(a, \rhd)$), this means that its content is a and the cell is to the immediate left (respectively, immediate right) with respect to the position of the head of the Turing machine.

Because we need to associate the same tape structure with several different configurations, the content of the relations T and First will end up being replicated within new hidden relations T^C and First^C, where it will be paired with the identifier of a configuration. For example, a fact $T^C(x, y, y')$ will indicate that, in the configuration identified by x, the cell y precedes the cell y'. Similarly, a fact $\text{First}^C(x, y)$ will indicate that y is the first cell of the tape of configuration x. Of course, we will enforce the condition that the relations T^C and First^C, devoid of the first attribute, are contained in T and First, respectively.

We now turn to the encoding of the computation tree. For this, we introduce a visible unary relation I that contains the identifier of the initial configuration. We also introduce the hidden binary relations $S^\exists, S_1^\forall$, and $S_2^\forall$. We recall that every configuration x with an existential control state has exactly one successor x' in the computation tree, so we represent this with the fact $S^\exists(x, x')$. Symmetrically, every configuration x with a universal control state has exactly two successors x_1 and x_2 in the computation tree, and we represent this with the facts $S_1^\forall(x, x_1)$ and $S_2^\forall(x, x_2)$.

So far, we have introduced the visible relations T, First, Last, V, I, and the hidden relations C, T^C, First^C, $S^\exists$, $S_1^\forall$, $S_2^\forall$. These are sufficient to store an encoding of the computation tree of the machine. However, the background theories are only allowed to contain IDs, which are not powerful enough to guarantee that these relations indeed represent a correct encoding. To overcome this problem, we will later introduce a few additional relations and exploit a union of CQs to detect those violations of the background theory that are not captured by IDs.

We now list some IDs in Σ that enforce basic restrictions on the relations:

- We begin with some sentences that guarantee that the relations T and T^C induce the same "successor" relation on the cells of the tape:

$$\begin{array}{llll}
T^C(x, y, y') & \rightarrow & T(y, y') & \text{First}^C(x, y) \rightarrow \exists y'\, T^C(x, y, y') \\
\text{First}^C(x, y) & \rightarrow & \text{First}(y) & T^C(x, y, y') \rightarrow \exists y''\, T^C(x, y', y'').
\end{array}$$

  Note that we can easily enforce that T contains the projection of T^C onto the last two attributes, and similar for First and First^C. But it is more difficult to enforce that T^C contains copies of T annotated with each configuration identifier. This will be done indirectly by requiring that every tuple (x, y) in First^C is the source of an infinite chain of successors inside T^C, all annotated with the same configuration identifier. Paired with the previous sentences, this will guarantee that T^C contains the annotated copy $\{x\} \times T$. Further note that, for this to work, it is crucial to have assumed that there is a "dummy" successor $T(y, y)$ on the last tape cell y. The existence of facts of the form $\text{First}^C(x, y)$ for each configuration x will be enforced later.

- We proceed by enforcing the existence of values associated with each cell in each configuration:

$$T^C(x, y, y') \rightarrow \exists z\, C(x, y, z) \qquad C(x, y, z) \rightarrow V(z).$$

  Note that the sentences in the background theory defined so far may allow a cell to be associated with multiple values. We will show later how to detect this case using a suitable query.

- We finally enforce a graph structure representing the evolution of the configurations, assuming that the machine starts with the existential configuration contained in the visible

relation I:

$$\begin{aligned}
I(x) &\rightarrow \exists x'\ S^{\exists}(x,x') & S^{\exists}(x,x') &\rightarrow \exists y\ \mathrm{First}^{C}(x,y) \\
S^{\exists}(x,x') &\rightarrow \exists x_1\ S_1^{\forall}(x',x_1) & & \\
S^{\exists}(x,x') &\rightarrow \exists x_2\ S_2^{\forall}(x',x_2) & S_1^{\forall}(x,x_1) &\rightarrow \exists y\ \mathrm{First}^{C}(x,y) \\
S_1^{\forall}(x,x_1) &\rightarrow \exists x'\ S^{\exists}(x_1,x') & & \\
S_2^{\forall}(x,x_2) &\rightarrow \exists x'\ S^{\exists}(x_2,x') & S_2^{\forall}(x,x_2) &\rightarrow \exists y\ \mathrm{First}^{C}(x,y).
\end{aligned}$$

Note that the rules on the right side above trigger the creation of a first tape cell for each configuration, which in turn spawns copies of the entire tape.

Next, we explain how to detect badly formed encodings of the computation tree. For this, we use additional visible relations Err_C, $\mathrm{Err}_{I,\mathrm{first}}$, $\mathrm{Err}_{I,\mathrm{last}}$, $\mathrm{Err}_{I,\mathrm{adj}}$, $\mathrm{Err}_{C,\mathrm{adj}}$, $\mathrm{Err}_{S^{\exists}}$, $\mathrm{Err}_{S_1^{\forall}}$, and $\mathrm{Err}_{S_2^{\forall}}$, instantiated as follows:

- The relation Err_C is binary and contains all pairs of *distinct* cell values from $V \times V$. This is used to check that every cell, in every configuration, is associated with at most one value. The CQ below holds precisely when this latter property is violated:

$$Q_C = \exists x\ y\ z\ z'\ C(x,y,z) \wedge C(x,y,z') \wedge \mathrm{Err}_C(z,z').$$

- The relation $\mathrm{Err}_{I,\mathrm{first}}$ is also binary and contains all pairs of values that cannot be associated with the first two cells in the initial configuration (recall that the first two cells carry the symbols $\vdash$ and $\sqcup$, and M starts with state q_0 on the first cell). Formally, $\mathrm{Err}_{I,\mathrm{first}}$ contains all pairs in $V \times V$ except (z_0, z_1), where $z_0 = (\vdash, q_0)$ and $z_1 = (\sqcup, \rhd)$. Accordingly, we can detect whether the values of the first two cells in the initial configuration are badly formed using the following CQ:

$$\begin{aligned}
Q_{I,\mathrm{first}} = \exists x\ y\ y'\ z\ z'\ & \\
& I(x) \wedge \mathrm{First}(y) \wedge T(y,y') \wedge C(x,y,z) \wedge C(x,y',z') \wedge \mathrm{Err}_{I,\mathrm{first}}(z,z').
\end{aligned}$$

- Similarly, the relation $\mathrm{Err}_{I,\mathrm{last}}$ contains pairs of values that cannot be associated with the last two cells in the initial configuration—that is, $\mathrm{Err}_{I,\mathrm{last}} = (V \times V) \setminus (z_1, z_{-1})$, where $z_1 = (\sqcup, \rhd)$ is defined as before and $z_{-1} = (\dashv, \rhd)$. We can detect whether the last two values in the initial configuration are inconsistent using the CQ

$$\begin{aligned}
Q_{I,\mathrm{last}} = \exists x\ y\ y'\ z\ z'\ & \\
& I(x) \wedge T(y,y') \wedge \mathrm{Last}(y') \wedge C(x,y,z) \wedge C(x,y',z') \wedge \mathrm{Err}_{I,\mathrm{last}}(z,z').
\end{aligned}$$

- The relation $\mathrm{Err}_{I,\mathrm{adj}}$ contains pairs of values that cannot appear on any two consecutive cells of the initial configuration, namely, $\mathrm{Err}_{I,\mathrm{adj}}$ contains all pairs in $V \times V$ but the following ones: $(z_0, z_1), (z_1, z_1), (z_1, z_{-1})$. This type of violation is checked with the CQ

$$Q_{I,\mathrm{adj}} = \exists x\ y\ y'\ z\ z'\ I(x) \wedge T(y,y') \wedge C(x,y,z) \wedge C(x,y',z') \wedge \mathrm{Err}_{I,\mathrm{adj}}(z,z').$$

- In a similar way, we can check violations of labelings of consecutive cells in every configuration. This is done with the binary visible relation $\mathrm{Err}_{C,\mathrm{adj}}$, instantiated with all pairs from $V \times V$ that cannot be adjacent in an arbitrary configuration (e.g., the pair $((a, \lhd), (b, \rhd))$), and the CQ

$$Q_{C,\mathrm{adj}} = \exists x\ y\ y'\ z\ z'\ T(y,y') \wedge C(x,y,z) \wedge C(x,y',z') \wedge \mathrm{Err}_{C,\mathrm{adj}}(z,z').$$

- The relation $\mathrm{Err}_{S^{\exists}}$ is used to check consistency along a transition that departs from an existential configuration. It contains a quadruple of cell values $(z, z', z'', z''') \in V \times V \times V \times V$ whenever it is *not* possible to have an existential configuration where the labels z, z', z'' appear on three consecutive positions y, y', y'', together with a successor configuration that

carries value z''' at position y'. Of course, the content of this relation depends on the transition function of the Turing machine. A violation of the corresponding constraint is exposed by the following CQ:

$$
\begin{aligned}
Q_{S^\exists} \ = \ \exists\, x\, x'\, y\, y'\, y''\, z\, z'\, z''\, z''' \\
S^\exists(x, x') \ \wedge\ T(y, y') \ \wedge\ T(y', y'') \ \wedge \\
C(x, y, z) \ \wedge\ C(x, y', z') \ \wedge\ C(x, y'', z'') \ \wedge\ C(x', y', z''') \ \wedge \\
\mathsf{Err}_{S^\exists}(z, z', z'', z''').
\end{aligned}
$$

- Similarly, the relation $\mathsf{Err}_{S_1^\forall}$ (respectively, $\mathsf{Err}_{S_2^\forall}$) contains quadruples of values that cannot appear on positions $y - 1, y, y + 1$ of some universal configuration x, and at position y of the first (respectively, second) successor configuration. The corresponding CQs $Q_{S_1^\forall}, Q_{S_2^\forall}$ are defined by

$$
\begin{aligned}
Q_{S_i^\forall} \ = \ \exists\, x\, x'\, y\, y'\, y''\, z\, z'\, z''\, z''' \\
S_i^\forall(x, x') \ \wedge\ T(y, y') \ \wedge\ T(y', y'') \ \wedge \\
C(x, y, z) \ \wedge\ C(x, y', z') \ \wedge\ C(x, y'', z'') \ \wedge\ C(x', y', z''') \ \wedge \\
\mathsf{Err}_{S_i^\forall}(z, z', z'', z''').
\end{aligned}
$$

It remains to check whether the Turing machine M reaches the rejecting state q_{rej} along some path of the computation tree. This can be done by introducing a last visible relation V_{rej} that contains all cell values of the form (a, q_{rej}), for some $a \in \Sigma$. The CQ that checks this property is

$$
Q_{\mathrm{rej}} \ = \ \exists\, x\, y\, z\, C(x, y, z) \ \wedge\ V_{\mathrm{rej}}(z).
$$

The final query is thus a disjunction of all of the above CQs:

$$
Q \ = \ Q_C \vee Q_{I, \mathrm{first}} \vee Q_{I, \mathrm{last}} \vee Q_{I, \mathrm{adj}} \vee Q_{C, \mathrm{adj}} \vee Q_{S^\exists} \vee Q_{S_1^\forall} \vee Q_{S_2^\forall} \vee Q_{\mathrm{rej}}.
$$

We are now ready to give the reduction. Denote by $\mathcal{V}_M$ the instance that captures the intended semantics of the visible relations T, First, Last, V, I, Err_C, $\mathsf{Err}_{I, \mathrm{first}}$, $\mathsf{Err}_{I, \mathrm{last}}$, $\mathsf{Err}_{I, \mathrm{adj}}$, $\mathsf{Err}_{C, \mathrm{adj}}$, $\mathsf{Err}_{S^\exists}$, $\mathsf{Err}_{S_1^\forall}$, and $\mathsf{Err}_{S_2^\forall}$. We have described these semantics previously and argued why they can be created in polynomial time. In the following, we prove that the Turing machine M has a successful computation tree where all paths visit the control state q_{acc} if and only if $\mathsf{PQI}(Q, \Sigma, S, \mathcal{V}_M) = \mathrm{false}$.

Suppose that M has a successful computation tree ρ. On the basis of ρ, and by following the intended semantics of the hidden relations $C, T^C, \mathrm{First}^C, S^\exists, S_1^\forall, S_2^\forall$, we can easily construct a full instance $\mathcal{F}$ that satisfies all sentences in Σ and agrees with $\mathcal{V}_M$ on the visible part. Furthermore, because we correctly encode a successful computation tree of M, the instance $\mathcal{F}$ violates every disjunct of Q, and hence $\mathsf{PQI}(Q, \Sigma, S, \mathcal{V}_M) = \mathrm{false}$.

Conversely, suppose that $\mathsf{PQI}(Q, \Sigma, S, \mathcal{V}_M) = \mathrm{false}$. Let $\mathcal{F}$ be an S-instance that agrees with $\mathcal{V}_M$ on the visible part, satisfies the sentences in Σ, and violates every disjunct of the UCQ Q. We first construct from $\mathcal{F}$ a graph, where every node encodes a configuration and, depending on whether the configuration is existential or universal, it has either one or two outgoing edges that represent some transitions of M. We will then argue that the unfolding of this graph from its initial node correctly represents an accepting computation tree of M. The nodes of the graph are identified by the values x that appear in facts of $\mathcal{F}$ of the form $S^\exists(x, x')$, $S_1^\forall(x, x')$, or $S_2^\forall(x, x')$. The initial node is identified by the unique value x_0 in the singleton visible relation I.

Thanks to the background theory Σ, every configuration identifier x also appears in the first column of the hidden relation First^C, and there exist similar occurrences in T^C and C, one for each cell of the tape. The content of C can then be used to determine the labeling of the tape cells, the

control state, and the head position for each configuration, as indicated by the intended semantics. For example, we set the content of a tape cell y in some configuration x to be a whenever there is a fact of the form $C(x, y, z)$, with z among (a, q), $(a, \triangleleft)$, or $(a, \triangleright)$. We observe that this is well defined (i.e., every tape position y at every configuration x has exactly one associated value) thanks to the sentences $T^C(x, y, y') \rightarrow \exists z\, C(x, y, z)$ and $C(x, y, z) \rightarrow V(z)$, and thanks to the fact that the query Q_C is not satisfied. Moreover, because the CQs $Q_{I,\text{first}}$, $Q_{I,\text{last}}$, and $Q_{I,\text{adj}}$ are also is not satisfied, the configuration at the initial node x_0 is correct. That is, encodes the tape content $\vdash \sqcup \cdots \sqcup \dashv$, with control state q_0, and head on the first position.

Next, the edges of the graph are constructed using the hidden binary relations $S^\exists$, $S_1^\forall$, and $S_2^\forall$ of $\mathcal{F}$. Formally, for every existential node x, the sentences constraining $S^\exists(x, x')$ imply the existence of at least one node x' forming a fact $S^\exists(x, x')$. We can thus choose any such node x' and declare (x, x') to be an edge of the graph. A similar argument applies to the universal nodes, with the only difference that we now introduce two edges instead of one. Moreover, using the assumption that the CQs $Q_{C,\text{adj}}$, $Q_{S^\exists}$, $Q_{S_1^\forall}$, and $Q_{S_2^\forall}$ are all violated, one can easily verify that the thus defined edges represent valid transitions between the encoded configurations. The above arguments imply that the unfolding of the graph from the initial node x_0 results in a valid computation tree of M. Finally, because the CQ Q_{rej} is also violated, the computation tree must be accepting.   □

Next, we show that PQI problems for UCQs can be reduced to PQI problems for CQs.

LEMMA 4.6. *Let Q be a Boolean UCQ without constants, let Σ be a set of first-order sentences over a schema $\mathbf{S}$, and let $\mathcal{V}$ be an instance for the visible part of $\mathbf{S}$. There exist a schema $\mathbf{S}'$, a Boolean CQ Q' without constants, a set Σ' of first-order sentences, and an $\mathbf{S}'_v$-instance $\mathcal{V}'$, all having polynomial size with respect to the original objects $\mathbf{S}$, Q, Σ, and $\mathcal{V}$, such that $\mathrm{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \text{true}$ if and only if $\mathrm{PQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \text{true}$.*

Moreover, if Σ consists of IDs (or, more generally, belongs to any of the languages considered in this article), then the same holds for Σ'.

PROOF. The general idea is as follows. For every visible (respectively, hidden) relation R of $\mathbf{S}$ of arity k, we add to $\mathbf{S}'$ a corresponding visible (respectively, hidden) relation R' of arity $k + 1$. The idea is that the additional attribute of R' represents a truth value (e.g., 0 or 1), which indicates the presence of a tuple in the original relation R. For example, the fact $R'(\bar{a}, 1)$ indicates the presence of the tuple $\bar{a}$ in the relation R, but $R'(\bar{a}, 0)$ does not. The sentences Σ will be rewritten accordingly, so as to propagate these truth values. We can then simulate the disjunctions in the query Q by using conjunctions and an appropriate look-up table Or. This technique has been used in a number of previous works (e.g., [34]) and will also be used later in this article. However, due to the nature of the PQI problem, we also need to add dummy facts $R'(\bot, \ldots, \bot, 0)$ to correctly transfer the validity from the UCQ Q to the CQ Q'. We give the full details in the following.

As mentioned, the new schema $\mathbf{S}'$ contains a copy R' of each relation R in $\mathbf{S}$, where R' is visible if and only if R is visible, and R' has arity $k + 1$ if and only if R has arity k. In addition, the schema $\mathbf{S}'$ contains the visible relations Or, Zero, One of arities 3, 1, 1, respectively, and some other visible relations Bottom_k of arity $k + 1$, for all k ranging from 0 to the maximal arity in $\mathbf{S}$.

Let us now describe the visible instance $\mathcal{V}'$ constructed from $\mathcal{V}$. We choose some fresh values 0, 1, and $\bot$ that do not belong to the active domain of $\mathcal{V}$. First, we include in $\mathcal{V}'$ the facts $\text{Or}(1, 1, 1)$, $\text{Or}(1, 0, 1)$, $\text{Or}(0, 1, 1)$, $\text{Zero}(0)$, $\text{One}(1)$, and $\text{Bottom}_k(\bot, \ldots, \bot, 0)$ for all arities k. Then, for each visible relation R of $\mathbf{S}$, we add to $\mathcal{V}'$ the fact $R(\bar{a}, 1)$ whenever $R(\bar{a})$ is a fact in $\mathcal{V}$.

As for the sentences in the background theory, we proceed as follows. If

$$R(\bar{x}) \;\rightarrow\; \exists \bar{y}\; S_1(\bar{z}_1) \wedge \ldots \wedge S_m(\bar{z}_m)$$

is a sentence in Σ, with $\bar{z}_1, \ldots, \bar{z}_m$ sequences of variables or constants from $\bar{x}, \bar{y}$, then we add to Σ' a corresponding sentence

$$R'(\bar{x}, b) \;\rightarrow\; \exists \bar{y} \; S_1'(\bar{z}_1, b) \wedge \ldots \wedge S_m'(\bar{z}_m, b).$$

Furthermore, for each relation R of arity k in S, we introduce the ID

$$\text{Bottom}_{k+1}(x_1, \ldots, x_k, y) \;\rightarrow\; R'(x_1, \ldots, x_k, y).$$

Recall that Bottom_{k+1} is a visible relation of S' that contains the single fact $(\bot, \ldots, \bot, 0)$. Therefore, the effect of the above sentence is to introduce dummy facts $R'(\bot, \ldots, \bot, 0)$ for each (visible or hidden) relation R'.

It now remains to transform the UCQ Q into a CQ Q'. Let $Q_1, \ldots, Q_n$ be the disjuncts (CQs) in Q. We define

$$Q' \;=\; \exists\, b_1 \ldots b_n \; b_0' \, b_1' \, \ldots \, b_n' \; \bigwedge_i Q_i'(b_i) \,\wedge\, \text{Zero}(b_0') \,\wedge\, \text{One}(b_n') \,\wedge\, \bigwedge_i \text{Or}(b_{i-1}', b_i, b_i'),$$

where each Q_i' is obtained from the i-th disjunct $Q_i = \exists \bar{y}\, S_1(\bar{z}_1) \wedge \ldots \wedge S_m(\bar{z}_m)$ of Q by letting $Q_i'(b_i) = \exists \bar{y}\, S_1'(\bar{z}_1, b_i) \wedge \ldots \wedge S_m'(\bar{z}_m, b_i)$. Note that the presence of the facts $R'(\bot, \ldots, \bot, 0)$ in every instance that extends $\mathcal{V}'$ and satisfies Σ' guarantees that the rewritten CQs $Q_i'(b_i)$ can always be satisfied by letting $b_i = 0$. In particular, the subquery $\bigwedge_i Q_i'(b_i)$ holds at least with all of the b_i's set to 0. The remaining part of the query Q' precisely requires that at least one of those b_i's is set to 1.

We are now ready to prove that $\text{PQI}(Q, \Sigma, S, \mathcal{V}) = \text{true}$ if and only if $\text{PQI}(Q', \Sigma', S', \mathcal{V}') = \text{true}$. Suppose that $\text{PQI}(Q', \Sigma', S', \mathcal{V}') = \text{true}$ and consider an S-instance $\mathcal{F}$ that satisfies the sentences in Σ and such that $\text{Visible}(\mathcal{F}) = \mathcal{V}$. Without loss of generality, we can assume that the active domain of $\mathcal{F}$ does not contain the values 0, 1, and $\bot$. We can easily transform $\mathcal{F}$ into an S'-instance $\mathcal{F}'$ by expanding all facts with the additional attributed value 1 and by adding new facts of the form $R'(\bot, \ldots, \bot, 0)$, for all relations $R' \in S'$, together with the visible facts $\text{Or}(1, 1, 1)$, $\text{Or}(1, 0, 1)$, $\text{Or}(0, 1, 1)$, $\text{Zero}(0)$, $\text{One}(1)$, and $\text{Bottom}_k(\bot, \ldots, \bot, 0)$ for all arities k. One easily verifies that $\mathcal{F}'$ satisfies the sentences in Σ' and agrees with $\mathcal{V}'$ on the visible part. Since $\text{PQI}(Q', \Sigma', S', \mathcal{V}') = \text{true}$, we know that $\mathcal{F}'$ also satisfies the query Q' and, in particular, it satisfies one of the conjuncts $Q_i'(b_i)$ of Q' with $b_i = 1$. This implies that $\mathcal{F}$ satisfies the corresponding Boolean CQ Q_i and hence Q as well.

Conversely, suppose that $\text{PQI}(Q, \Sigma, S, \mathcal{V}) = \text{true}$ and consider an S'-instance $\mathcal{F}'$ that satisfies the sentences in Σ' and such that $\text{Visible}(\mathcal{F}') = \mathcal{V}'$. By selecting from $\mathcal{F}'$ only the facts of the form $R'(\bar{a}, 1)$, with $R \in S$, and by projecting away the last attribute, we obtain an S-instance $\mathcal{F}$ that satisfies the sentences in Σ and such that $\text{Visible}(\mathcal{F}) = \mathcal{V}$. Since $\text{PQI}(Q, \Sigma, S, \mathcal{V}) = \text{true}$, we know that $\mathcal{F}$ satisfies at least one of the disjuncts Q_i of Q. This immediately implies that $\mathcal{F}'$ satisfies the CQ $Q_i'(b_i)$ with $b_i = 1$. As for the remaining conjuncts of the query Q', we recall that $\mathcal{F}'$ must contain facts of the form $R'(\bot, \ldots, \bot, 0)$ for all relations R'. Thanks to these facts, the CQs $Q_j'(b_j)$ hold on $\mathcal{F}'$ with $b_j = 0$, for all $j \neq i$, and hence Q' holds on $\mathcal{F}'$ as well. □

Putting the above results together, we obtain the following.

COROLLARY 4.7. *There are a Boolean CQ Q without constants and a set Σ of IDs over a schema S for which the problem $\text{PQI}(Q, \Sigma, S, \mathcal{V})$ is* EXPTIME-*hard in data complexity.*

We note that the above lower bound for data complexity makes use of a schema with arity above 2, even for UCQs. See, for example, the ternary relation C. We do not know whether our lower bound still holds for the arity 2 case. Our results contrasts with results of Franconi et al. [31],

which show that the data complexity lies in co-NP (and can be co-NP-hard) for certain description logics over an arity 2 schema.

We now turn to the combined complexity. We start by noting that the 2ExpTime upper bound of Theorem 4.1 is tight even for IDs. When the query is allowed to be a UCQ, a stronger result has been shown in Theorem 8 of Ngo et al. [47], as the constraints there are IDs on an arity 2 schema. In the case of linear TGDs, an alternative reduction that gives a 2ExpTime lower bound has been pointed out by a reviewer, going through results on disjunctive IDs [22]. In the appendix, we give our own argument that allows us to use a CQ rather than a UCQ, for IDs rather than linear TGDs. However, our result does not use an arity 2 schema, so it is orthogonal to Theorem 8 of Ngo et al. [47].

THEOREM 4.8 (FROM NGO ET AL. [47] IN THE CASE OF UCQs). *Checking* $PQI(Q, \Sigma, S, \mathcal{V})$*, where Q ranges over Boolean CQs without constants and Σ ranges over sets of IDs, is* 2ExpTime*-hard for combined complexity.*

4.2 A Chase Procedure for PQI with Arbitrary TGDs

In this section, we show that, for TGDs, the PQI problem (over arbitrary, possible infinite instances) can be characterized using a variant of the chase procedure [28, 48]. We will make use of this characterization in the next sections, where we consider the schema-level problems ∃PQI and ∃NQI. For simplicity, we focus here on TGDs only, but in fact, the results extend in a straightforward way to combinations of TGDs and EGDs.

Our chase procedure receives as input a relational schema S, a background theory Σ consisting of TGDs, and an initial instance $\mathcal{F}_0$ for the schema S, which does not need to satisfy the background theory Σ. The goal of the procedure is to produce a collection of instances (not necessarily finite) that satisfy Σ, extend the initial instance $\mathcal{F}_0$, and agree with this instance on the visible part. The goal is achieved by repeatedly adding new facts to the initial instance $\mathcal{F}_0$ so as to satisfy the sentences in Σ, in a way similar to the classical chase procedure for TGDs. However, non-deterministic choices are sometimes needed to map the newly generated tuples in a visible relation to some existing facts in $\mathcal{F}_0$. Our technique is actually a variant of the "disjunctive chase" of Deutsch et al. [27], which produces multiple instances.

Recall that without loss of generality, TGDs can be assumed to have exactly one atom in the right-hand side (as can be achieved by introducing auxiliary relations where needed).

In what follows, by a *homomorphism* (relative to a background theory and query), we will mean a function f mapping domain elements of one instance to domain elements of another instance such that (i) for every fact $R(\bar{a})$ of the first instance, its f-image $R(f(\bar{a}))$ is a fact of the second instance, and (ii) $f(a) = a$ for every constant a appearing in the query and/or background theory. For an instance I, we will also denote by $f(I)$ the instance consisting of all f-images of the facts of I.

The procedure builds a *chase tree* of instances, starting with the singleton tree consisting of the input S-instance $\mathcal{F}_0$ and extending the tree by repeatedly applying the following steps. It chooses an instance K at some leaf of the current tree; a TGD $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \rightarrow \exists \bar{y}\, S(\bar{z})$ in Σ, where $\bar{z}$ is a sequence of (possibly repeated) variables from $\bar{x}_1, \ldots, \bar{x}_m, \bar{y}$; and a homomorphism f that maps $R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)$ to some facts in K. Note that then the procedure constructs a new instance from K by adding the fact $S(f'(\bar{z}))$, where f' is an extension of f that maps, in an injective way, the existentially quantified variables in $\bar{y}$ to some values that are not in K. In the usual terminology of the chase, such an added value is called a *null*, and adding this fact is called *performing a chase step*. Immediately after this step, and only when the relation S is visible, the

procedure replaces the instance $K' = K \cup \{S(f'(\bar{z}))\}$ with copies of it of the form $g(K')$ such that $\mathsf{Visible}(g(K')) = \mathsf{Visible}(\mathcal{F}_0)$, for all possible homomorphisms g that map the values $f'(\bar{z})$ to some values in the active domain $\{a_1, \ldots, a_n\}$ of the visible instance $\mathsf{Visible}(\mathcal{F}_0)$ (and such that g acts as the identity function on all other values). Note that the active domain of $\{a_1, \ldots, a_n\}$ of $\mathsf{Visible}(\mathcal{F}_0)$ does not contain null values. In the language of prior works on the chase [27], this step would be a sequence of "disjunctive chase steps," for disjunctive EGDs of the form $S(\bar{z}) \rightarrow z_i = a_1 \vee \ldots \vee z_i = a_n$). The resulting instances $g(K')$ are then appended as new children of K in the tree-shaped collection. In the special case where there are no homomorphisms g such that $\mathsf{Visible}(g(K')) = \mathsf{Visible}(\mathcal{F}_0)$, we append a "dummy instance" $\perp$ as a child of K: this is used to represent the fact that the chase step from K led to an inconsistency (the dummy node will never be extended during the subsequent chase steps). If S is not visible, then the instance K' is simply appended as a new child of K.

This process continues iteratively using a strategy that is "fair," namely, that guarantees that whenever a dependency is applicable in a node on a maximal path of the chase tree, then it will be fired at some node (possibly later) on that same maximal path (unless the path ends with $\perp$). In the limit, the process generates a possibly infinite tree-shaped collection of instances. It remains to complete the collection with "limits" to guarantee that the sentences in the background theory are satisfied. Consider any infinite path $K_0, K_1, \ldots$ in the tree (if there are any). It follows from the construction of the chase tree that the instances on the path form a chain of homomorphic embeddings $K_0 \xrightarrow{h_0} K_1 \xrightarrow{h_1} \ldots$. Such chains of homomorphic embeddings admit a natural notion of limit, which we denote by $\lim_{n \in \mathbb{N}} K_n$. We omit the details of this construction here, which can be found, for instance, in the work of Chang and Keisler [25]. The limit instance $\lim_{n \in \mathbb{N}} K_n$ satisfies the background theory Σ. We denote by $\mathsf{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{F}_0)$ the collection of all non-dummy instances that occur at the leaves of the chase tree, plus all limit instances of the form $\lim_{n \in \mathbb{N}} K_n$, where $K_0, K_1, \ldots$ is an infinite path in the chase tree. This is well defined only once the ordering of steps is chosen, but for the results below, which order is chosen will not matter, so we abuse notation by referring to $\mathsf{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{F}_0)$ as a single object.

Note that the instances in $\mathsf{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{F}_0)$ are not necessarily finite. Every instance in $\mathsf{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{F}_0)$ satisfies the sentences in Σ and, in addition, agrees with $\mathcal{F}_0$ on the visible part of the schema.

LEMMA 4.9. *Let Σ consist of TGDs, let $\mathcal{F}_0$ be an instance of a schema S, and let $\mathcal{F}$ be another instance over the same schema that contains all facts of $\mathcal{F}_0$, agrees with $\mathcal{F}_0$ on the visible part (i.e., $\mathsf{Visible}(\mathcal{F}) = \mathsf{Visible}(\mathcal{F}_0)$), and satisfies all TGDs in Σ. Then, there exist an instance $K \in \mathsf{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{F}_0)$ and a homomorphism from K to $\mathcal{F}$.*

PROOF. We consider the chase tree for $\mathsf{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{F}_0)$, and based on the full instance $\mathcal{F}$, we identify inside this chase tree a suitable path $K_0, K_1, \ldots$ and a corresponding sequence of homomorphisms $h_0, h_1, \ldots$ such that for all $n \in \mathbb{N}$, h_n maps K_n to $\mathcal{F}$. Once these sequences are defined, the lemma will follow easily by letting $K = \lim_{n \in \mathbb{N}} K_n$ and $h = \lim_{n \in \mathbb{N}} h_n$—that is, $h(\bar{a}) = \bar{b}$ if $h_n(\bar{a}) = \bar{b}$ for all but finitely many $n \in \mathbb{N}$.

The base step is easy, as we simply let K_0 be the initial instance $\mathcal{F}_0$, which appears at the root of the chase tree, and let h_0 be the identity. As for the inductive step, suppose that K_n and h_n are defined for some step n, and suppose that $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \rightarrow \exists \bar{y}\, S(\bar{z})$ is the dependency that is applied at node K_n, where $\bar{z}$ is a sequence of variables from $\bar{x}_1, \ldots, \bar{x}_m, \bar{y}$. Let $R_1(f(\bar{x}_1))$, $\ldots, R_m(f(\bar{x}_m))$ be the facts in the instance K_n that have triggered the chase step, where f is a homomorphism from the variables in $\bar{x}_1, \ldots, \bar{x}_m$ to the domain of K_n. Since $\mathcal{F}$ satisfies the same dependency and contains the facts $R_1(h_n(f(\bar{x}_1))), \ldots, R_m(h_n(f(\bar{x}_m)))$, it must also contain a fact of the form $S(h'(f'(\bar{z})))$, where f' is the extension of f that is the identity on the existentially

quantified variables $\bar{y}$ and h' is some extension of h_n that maps the variables $\bar{y}$ to some values in the domain of $\mathcal{F}$.

Now, to choose the next instance K_{n+1}, we distinguish two cases, depending on whether S is visible or not. If S is not visible, then we know that the chase step appends a single instance $K' = K_n \cup \{S(f'(\bar{z}))\}$ as a child of K_n; accordingly, we let $K_{n+1} = K'$ and $h_{n+1} = h' \circ f'$. Otherwise, if S is visible, then we observe that h' is a homomorphism from $K' = K_n \cup \{S(f'(\bar{z}))\}$ to $\mathcal{F}$. In particular, h' maps the variables $\bar{z}$ to some values in the active domain of the visible part $\mathrm{Visible}(\mathcal{F}_0)$, and hence $h'(K')$ agrees with $\mathcal{F}_0$ on the visible part of the schema. This implies that the chase step adds at least the instance $h'(K')$ as a child of K_n. Accordingly, we can define $K_{n+1} = h'(K')$ and $h_{n+1} = f'$. Given the above constructions, it is easy to see that the homomorphism h_{n+1} maps K_{n+1} to $\mathcal{F}$.

Proceeding in this way, we either arrive at a leaf, and in this case we are done, or we obtain an infinite path of the chase tree $K_0 \xrightarrow{h_0} K_1 \xrightarrow{h_1} \ldots$, with homomorphisms $h'_i : K_i \to \mathcal{F}$ such that $h_i \circ h'_{i+1}$ extends h'_i, for all $i \in \mathbb{N}$. In the latter case, it can be shown that the limit $\lim_{n \in \mathbb{N}} K_n$ also homomorphically maps to $\mathcal{F}$. □

The following proposition characterizes the positive instances of the PQI problem when the background theory consists of TGDs without constants.

PROPOSITION 4.10. *If Q is a Boolean UCQ, Σ is a set of TGDs over a schema S, and $\mathcal{V}$ is a visible instance, then $\mathrm{PQI}_\infty(Q, \Sigma, S, \mathcal{V}) = \mathrm{true}$ if and only if every instance K in $\mathrm{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{V})$ satisfies Q.*

PROOF. Suppose that $\mathrm{PQI}_\infty(Q, \Sigma, S, \mathcal{V}) = \mathrm{true}$ and recall that every instance in $\mathrm{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{V})$ satisfies the TGDs in Σ and agrees with $\mathcal{V}$ on the visible part. In particular, this means that every instance in $\mathrm{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{V})$ satisfies the query Q.

Conversely, suppose that $\mathrm{PQI}_\infty(Q, \Sigma, S, \mathcal{V}) = \mathrm{false}$. This means that there is a (possibly infinite) S-instance $\mathcal{F}$ that has $\mathcal{V}$ as the visible part, satisfies the TGDs in Σ, but not the query Q. By Lemma 4.9, letting $\mathcal{F}_0 = \mathcal{V}$, we get an instance $K \in \mathrm{Chases}_{\mathrm{vis}}(\Sigma, S, \mathcal{V})$ and a homomorphism from K to $\mathcal{F}$. Since Q is preserved under homomorphisms, K does not satisfy Q. □

As we mentioned earlier, although we only consider TGDs in this article, these above results extend naturally to the case with EGDs. Chasing an EGD of the form $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \to x = x'$, where x, x' are two variables from $\bar{x}_1, \ldots, \bar{x}_m$, amounts to applying a suitable homomorphism that identifies the two values $h(x)$ and $h(x')$ whenever the facts $R_1(h(\bar{x}_1)), \ldots, R_m(h(\bar{x}_m))$ belong to the instance under consideration. Note that this operation can lead to a failure (i.e., a dummy instance) when $h(x)$ and $h(x')$ are distinct values from the active domain of the visible part $\mathcal{V}$.

4.3 The Schema-Level Problem ∃PQI

In this section, we focus on the schema-level problem ∃PQI—that is, the problem of deciding the existence of an instance $\mathcal{V}$ such that $\mathrm{PQI}(Q, \Sigma, S, \mathcal{V}) = \mathrm{true}$.

Choose an arbitrary domain element a, and let $\mathcal{V}_{\{a\}}$ be a fixed instance for the visible part of a schema S whose domain contains the single value a and whose visible relations are singleton relations of the form $\{(a, \ldots, a)\}$. We will show that, for certain languages for the background theories, if $\exists\mathrm{PQI}(Q, \Sigma, S) = \mathrm{true}$, then the witnessing instance can be taken to be $\mathcal{V}_{\{a\}}$. This can be viewed as an extension of the "critical instance" method that has been applied previously to chase termination problems: Proposition 3.7 of Marnette and Geerts [43] states a related result for disjunctive TGDs in isolation; Gogacz and Marcinkowski [32] call such an instance a "well of

positivity." We will use the terminology of the earlier work [35] and the later works [10, 17], calling this instance the *critical instance* and the element *a* the *critical element*.

THEOREM 4.11. *For every Boolean UCQ Q without constants, and every set Σ of TGDs without constants, $\exists PQI_\infty(Q, \Sigma, S) = \text{true}$ if and only if $PQI_\infty(Q, \Sigma, S, \mathcal{V}_{\{a\}}) = \text{true}$.*

Before proving Theorem 4.11, we first establish a lemma. Recall that the visible instance $\mathcal{V}_{\{a\}}$ is constructed over a singleton active domain and the TGDs in the background theory Σ have no constants. This implies that there are no disjunctive choices to perform while chasing with the TGDs starting from the initial instance $\mathcal{V}_{\{a\}}$. Moreover, it is easy to see that this chase always succeeds. In other words, it returns a collection $\text{Chases}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$ with exactly one instance—in particular, $\mathcal{V}_{\{a\}}$ is a realizable instance. By a slight abuse of notation, we denote by $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$ the unique instance in the collection $\text{Chases}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$.

LEMMA 4.12. *If Σ is a set of TGDs without constants over a schema S and $\mathcal{V}$ is an instance of the visible part of S, then every instance $K \in \text{Chases}_{\text{vis}}(\Sigma, S, \mathcal{V})$ maps homomorphically to $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$—that is, $h(K) \subseteq \text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$ for some homomorphism h.*

PROOF. Recall that the instances in $\text{Chases}_{\text{vis}}(\Sigma, S, \mathcal{V})$ are either leaves or limits of infinite paths of the chase tree. In the following, we prove that every instance K in the chase tree for $\text{Chases}_{\text{vis}}(\Sigma, S, \mathcal{V})$ maps to $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$ via some homomorphism h. In addition, we ensure that if K' is a descendant of K in the same chase tree, then the corresponding homomorphism h' is obtained by composing some homomorphism with an extension of h. This way of constructing homomorphisms is compatible with limits in the following sense: if $h_0, h_1, \ldots$ are homomorphisms mapping instances $K_0, K_1, \ldots$ along an infinite path of the chase tree, then there is a homomorphism $\lim_{n \in \mathbb{N}} h_n$ that maps the limit instance $\lim_{n \in \mathbb{N}} K_n$ to $\mathcal{F}$.

For the base case of the induction, we consider the initial instance $\mathcal{V}$ at the root of the chase tree, which clearly maps homomorphically to $\mathcal{V}_{\{a\}}$ (recall that there are no constants in the query or in the TGDs, and homomorphisms are free to map all domain elements to a). For the inductive case, we consider an instance K in the chase tree and suppose that it maps to $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$ via a homomorphism h. We also consider an instance K' that is a child of K and is obtained by chasing some dependency $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \rightarrow \exists \bar{y}\, S(\bar{z})$, where $\bar{z}$ is a sequence of variables from $\bar{x}_1, \ldots, \bar{x}_m, \bar{y}$. This means that there exist two homomorphisms f and g such that

(1) f maps the variables $\bar{x}_1, \ldots, \bar{x}_m$ to some values in K and maps injectively the variables $\bar{y}$ to fresh values;
(2) g either maps $f(\bar{z})$ to values in the active domain of $\mathcal{V}$ or is the identity on $f(\bar{z})$, depending on whether S is visible or not;
(3) $R_j\big(f(\bar{x}_j)\big) \in K$ for all $1 \le j \le m$;
(4) $K' = g\big(K \cup \{S(f(\bar{z}))\}\big)$.

Note that h maps each fact $R_j\big(f(\bar{x}_j)\big)$ in K to $R_j\big(h(f(\bar{x}_j))\big)$ in $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$. Since $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$ satisfies the chased dependency, it must also contain a fact of the form $S\big(h'(f(\bar{z}))\big)$, where h' is a homomorphism that extends h on the fresh values $f(\bar{y})$. Moreover, if S is visible, then h' maps all values $f(\bar{z})$ to the same value a, which is the only element of the active domain of $\mathcal{V}_{\{a\}}$.

We can now define a homomorphism that maps the instance $K' = g\big(K \cup \{S(f(\bar{z}))\}\big)$ to $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$. If S is not visible, then we recall that g is the identity on $f(\bar{z})$, and hence h' already maps $K' = g\big(K \cup \{S(f(\bar{z}))\}\big) = K \cup \{S(f(\bar{z}))\}$ to $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$. Otherwise, if S is visible, then we recall that g maps $f(\bar{z})$ to values in the active domain of $\mathcal{V}$, we let g' be the

function that maps all values of the active domain of $\mathcal{V}$ to a, and finally we define $h'' = h' \circ g'$. In this way, h'' maps $K' = g\big(K \cup \{S(f(\bar{z}))\}\big)$ to $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$. □

Proof of Theorem 4.11. One direction is trivial: if $\text{PQI}_\infty(Q, \Sigma, S, \mathcal{V}_{\{a\}})$ = true, then clearly $\exists \text{PQI}_\infty(Q, \Sigma, S)$ = true. For the converse direction, suppose that $\exists \text{PQI}_\infty(Q, \Sigma, S)$ = true. This implies the existence of a realizable instance $\mathcal{V}$ such that $\text{PQI}_\infty(Q, \Sigma, S, \mathcal{V})$ = true. By Proposition 4.10, every instance in $\text{Chases}_{\text{vis}}(\Sigma, S, \mathcal{V})$ satisfies the query Q. Moreover, by Lemma 4.12, every instance in $\text{Chases}_{\text{vis}}(\Sigma, S, \mathcal{V})$ maps homomorphically to $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$. Hence, the unique instance in $\text{Chases}_{\text{vis}}(S, \Sigma, \mathcal{V}_{\{a\}})$, i.e. $\text{chase}_{\text{vis}}(\Sigma, S, \mathcal{V}_{\{a\}})$, also satisfies Q. By applying Proposition 4.10 again, we conclude that $\text{PQI}_\infty(Q, \Sigma, S, \mathcal{V}_{\{a\}})$ = true. □

Although we only consider TGDs in this article, it is worth pointing out that the same results naturally extend to combinations of TGDs and EGDs without constants. By pairing Theorem 4.11 with the upper bound and the finite controllability for instance-level problems (Theorem 4.1), one immediately obtains the following corollary.

Corollary 4.13. $\exists \text{PQI}(Q, \Sigma, S)$ *with Q ranging over Boolean UCQs without constants and Σ ranging over sets of FGTGDs without constants, is decidable in* 2ExpTime, *and is finitely controllable.*

A matching lower bound can be obtained by reduction from OWQ.

Proposition 4.14. *Let $\mathcal{L}$ be any constraint language that includes IDs. There is a polynomial time reduction from* OWQ *for Boolean CQs (with or without constants) and constraints in $\mathcal{L}$, to $\exists \text{PQI}$ for Boolean CQs (with, respectively, without constants) and constraints in $\mathcal{L}$.*

Proof. Let Q be a Boolean CQ, let Σ be a set of constraints over a schema S belonging to $\mathcal{L}$, and let $\mathcal{F}$ be an instance of the schema S. We show how to reduce the OWQ problem for Q, Σ, S, and $\mathcal{F}$ to a problem $\exists \text{PQI}(Q', \Sigma', S')$. The idea is to create a copy of the instance $\mathcal{F}$ in the hidden part of the schema, which can then be extended arbitrarily.

Formally, we let the transformed schema S' consist of all relations in S, which are assumed to be hidden, plus an additional visible relation Good of arity 0. We then introduce a variable y_b for each value in the active domain of $\mathcal{F}$, and we let Σ' contain all sentences from Σ, plus the sentence Good $\to \exists \bar{y} \, Q_{\mathcal{F}}$, where $\bar{y}$ contains one variable y_b for each value b in the active domain of $\mathcal{F}$ and $Q_{\mathcal{F}}$ is the conjunction of the atoms of the form $A(y_{b_1}, \ldots, y_{b_k})$, for all facts $A(b_1, \ldots, b_k)$ in $\mathcal{F}$. Note that the visible instance $\mathcal{V}_{\text{Good}}$ that contains the atom Good is realizable, since it can be completed (using the chase) to an S'-instance $\mathcal{F}'$ that satisfies the sentences in Σ'. Let $Q' = Q \land$ Good. We claim that $\exists \text{PQI}(Q', \Sigma', S')$ = true if and only if Q is certain with respect to Σ on $\mathcal{F}$. In one direction, suppose that $\exists \text{PQI}(Q', \Sigma', S')$ = true holds. The witness visible instance having PQI can only be the instance $\mathcal{V}_{\text{Good}}$. Consider an instance $\mathcal{F}'$ containing all facts of $\mathcal{F}$ and satisfying the original sentences Σ. By setting Good to true in $\mathcal{F}'$, we have an instance satisfying Σ', and since $\mathcal{V}_{\text{Good}}$ has a PQI, then we know that this instance must satisfy Q' and hence Q. Thus, Q is certain with respect to Σ on $\mathcal{F}$ as required. Conversely, suppose that Q is certain with respect to Σ on $\mathcal{F}$. Letting $C_{\mathcal{F}}$ be the chase of $\mathcal{F}$ with respect to Σ, we see that $C_{\mathcal{F}}$ satisfies Q. We will show that there is a PQI for Q', Σ', S on $\mathcal{V}_{\text{Good}}$. Thus, fix an instance $\mathcal{F}'$ where Good and Σ' holds. The additional sentence implies that $\mathcal{F}'$ contains the image of $\mathcal{F}$ under some homomorphism h. But h extends to a homomorphism of $C_{\mathcal{F}}$ into $\mathcal{F}'$. Thus, $\mathcal{F}'$ satisfies Q and therefore satisfies Q'. Thus, there is a PQI on $\mathcal{V}_{\text{Good}}$ as required.

Therefore, we have reduced the OWQ problem for Q, Σ, and S to the problem $\exists \text{PQI}(Q', \Sigma', S')$. □

It was shown in the work of Cali et al. [23] that OWQ is 2ExpTime-hard for *guarded TGDs* without constants and CQs without constants (in combined complexity, when the schema is included

in the input of the problem). Guarded TGDs are a special case of connected FGTGDs. Therefore, by the above reduction, we obtain a lower bound for ∃PQI that matches the upper bound in Corollary 4.13.

COROLLARY 4.15. *The problem* ∃PQI(Q, Σ, S)*, where Q ranges over Boolean CQs without constants and Σ over sets of connected FGTGDs without constants, is* 2EXPTIME*-hard.*

Next, we show that allowing disjunctions or constants in the background theory sentences leads to undecidability. In particular, this shows that our "critical instance" method for reducing ∃PQI to PQI fails for these cases.

THEOREM 4.16. *The problem* ∃PQI(Q, Σ, S) *is undecidable as Q ranges over Boolean UCQs and Σ over sets of disjunctive linear TGDs without constants.*

PROOF. The proof uses a technique that will be exploited for many of our schema-level undecidability arguments. We will reduce the existence of a tiling to the ∃PQI problem. The tiling itself will correspond to the visible instance that has a PQI. The invisible relations will store "challenges" to the correctness of the tiling. The UCQ Q will have disjuncts that return true exactly when the challenge to correctness is passed. There will be challenges to the labeling of adjacent cells, challenges to the correctness of the initial tile, and challenges to the correct shape of the adjacency relationship—that is, challenges that the tiling is really grid-like. A correct tiling corresponds to every challenge being passed and thus corresponds to a visible instance where every extension satisfies Q.

To simplify the proof, we first provide undecidability of the "unrestricted" variant ∃PQI$_\infty$, which asks if there is an arbitrary instance of the visible schema such that every (possibly infinite) superinstance satisfying the sentences in Σ also satisfies Q. After that, we discuss how to adapt the undecidability argument so that it applies to finite instances only—that is, for the actual ∃PQI problem at hand.

We reduce the problem of tiling the infinite grid, which is known to be undecidable, to the problem ∃PQI$_\infty$. Recall that an instance of the tiling problem consists of a finite set T of available tiles, some horizontal and vertical constraints, given by two relations $H, V \subseteq T \times T$, and an initial tile $t_\perp \in T$ for the lower-left corner. The problem consists of deciding whether there is a tiling function $f : \mathbb{N} \times \mathbb{N} \to T$ such that

(1) $f(0, 0) = t_\perp$;
(2) $(f(i, j), f(i + 1, j)) \in H$ for all $i, j \in \mathbb{N}$;
(3) $(f(i, j), f(i, j + 1)) \in V$ for all $i, j \in \mathbb{N}$.

Given an instance $(T, H, V, t_\perp)$ of the tiling problem, we show how to construct a schema S, a query Q, and a set of disjunctive linear TGDs over S such that ∃PQI$_\infty$(Q, Σ, S) = true if and only if there is a tiling function for $(T, H, V, t_\perp)$.

The basic idea is that the visible instance that witnesses ∃PQI$_\infty$ should represent a candidate tiling, and the invisible instances represent challenges to the correctness of the tiling. Every cell of the grid is identified with some value, and we use two visible binary relations E_H, E_V to represent the horizontal and vertical edges of the grid. We also introduce a unary visible relation U_t, for each tile $t \in T$, to represent a candidate tiling function on the grid.

We begin by enforcing the existence of an initial node with the associated tile $t_\perp$. For this, we introduce another visible relation Init, of arity 0, and linear TGD

$$\mathsf{Init} \; \rightarrow \; \exists x \, U_{t_\perp}(x).$$

It is also easy to guarantee that every node is connected to at least another node in the relation E_H (respectively, E_V), and that this latter node has an associated tile that satisfies the horizontal

constraints H (respectively, the vertical constraints V). To do so, we use the following TGDs, which can be easily converted to disjunctive linear TGDs:

$$U_t(x) \;\rightarrow\; \exists y\, E_H(x, y) \;\wedge\; \bigvee_{(t, t') \in H} U_{t'}(y) \qquad\qquad \text{(for all tiles } t \in T)$$

$$U_t(x) \;\rightarrow\; \exists z\, E_V(x, z) \;\wedge\; \bigvee_{(t, t') \in V} U_{t'}(z). \qquad\qquad \text{(for all tiles } t \in T)$$

We now explain how to enforce a grid structure on the relations E_H and E_V, and how to guarantee that each node has exactly one tile associated with it. Of course, we cannot directly use disjunctive TGDs to guarantee that E_H and E_V correctly represent the horizontal and vertical edges of the grid. However, we can introduce additional hidden relations that make it possible to mark certain nodes so as to expose the possible violations. We first show how to expose violations to the fact that the horizontal edge relation is a function. The idea is to select nodes in E_H to challenge functionality. Formally, the horizontal challenge is captured by a hidden ternary relation $\mathsf{HChallenge}_{\mathsf{funct}}$, by the linear TGDs

$$\mathsf{Init} \;\rightarrow\; \exists x\, y\, y'\, \mathsf{HChallenge}_{\mathsf{funct}}(x, y, y')$$

$$\mathsf{HChallenge}_{\mathsf{funct}}(x, y, y') \;\rightarrow\; E_H(x, y) \;\wedge\; E_H(x, y')$$

and by the CQ

$$Q_H \;=\; \exists x\, y\, \mathsf{HChallenge}_{\mathsf{funct}}(x, y, y).$$

Note that if the visible fact Init is present and the relation E_H correctly describes the horizontal edges of the grid, then the above query Q_H is necessarily satisfied by any instance of $\mathsf{HChallenge}_{\mathsf{funct}}$ that satisfies the above sentences: the only way to give a non-empty instance for $\mathsf{HChallenge}_{\mathsf{funct}}$ is to use triples of the form (x, y, y). Conversely, if the relation E_H is not a function, namely, if there exist nodes x, y, y' such that $(x, y), (x, y') \in E_H$ and $y \neq y'$, then the singleton instance $\{(x, y, y')\}$ for the hidden relation $\mathsf{HChallenge}_{\mathsf{funct}}$ will satisfy the associated sentences of the background theory and violate the query Q_H. Note that we do not require that the relation E_H is injective (this could be still done but is not necessary for the reduction). Similarly, we can use a hidden relation $\mathsf{VChallenge}$ and analogous background theory sentences and query Q_V to challenge the functionality of E_V.

In the same way, we can challenge the confluence of the relations E_H and E_V. For this, we introduce a hidden relation $\mathsf{CChallenge}$ of arity 5, which is associated with the background theory sentences

$$\mathsf{Init} \;\rightarrow\; \exists x\, y\, z\, w\, w'\, \mathsf{CChallenge}(x, y, z, w, w')$$

$$\mathsf{CChallenge}(x, y, z, w, w') \;\rightarrow\; E_H(x, y) \;\wedge\; E_V(x, z) \;\wedge\; E_V(y, w) \;\wedge\; E_H(z, w')$$

and the CQ

$$Q_C \;=\; \exists x\, y\, z\, w\, \mathsf{CChallenge}(x, y, z, w, w).$$

As before, we can argue that there is a PQI for Q_C if and only if the horizontal and vertical edge relations are confluent—that is, $(x, w) \in E_H \circ E_V$ and $(x, w') \in E_V \circ E_H$ imply $w = w'$.

We need now to ensure that every node is labeled with at most one tile, or equally that there are no relations U_t and $U_{t'}$, for distinct tiles $t \neq t' \in T$, that have non-empty intersection. For that, we add the two following sentences, where A and B are hidden relations

$$\mathsf{Init} \;\rightarrow\; \exists x\, A(x) \vee B(x)$$

$$B(x) \;\rightarrow\; \bigvee_{t \neq t'} (\, U_t(x) \wedge U_{t'}(x)).$$

Finally, we add the CQ

$$Q_A = \exists x \, A(x).$$

Now that we described all of the visible and hidden relations of the schema $\mathbf{S}$, and the associated sentences Σ, we define the query for the $\exists \mathsf{PQI}_\infty$ problem as the conjunction of the atom Init and all previous UCQs (for this, we distribute the disjunctions and existential quantifications over the conjunctions):

$$Q = \mathsf{Init} \wedge Q_A \wedge Q_H \wedge Q_V \wedge Q_C.$$

It remains to show that $\exists \mathsf{PQI}_\infty(Q, \Sigma, \mathbf{S}) = \mathrm{true}$ if and only if there is a correct tiling of the infinite grid, namely, a function $f : \mathbb{N} \times \mathbb{N} \to T$ that satisfies conditions (1), (2), and (3) above.

Suppose that there is a correct tiling $f : \mathbb{N} \times \mathbb{N} \to T$. We construct the visible instance $\mathcal{V}$ that contains the fact Init and the relations E_H, E_V, and U_t with the intended semantics: $E_H = \{((i, j), (i + 1, j)) \mid i, j \in \mathbb{N}\}$, $E_V = \{((i, j), (i, j + 1)) \mid i, j \in \mathbb{N}\}$, and $U_t = \{(i, j) \mid f(i, j) = t\}$ for all $t \in T$. Since no error can be exposed on the relations E_H, E_V, and U_t, no matter how we construct a full instance $\mathcal{F}$ that agrees with $\mathcal{V}$ on the visible part and satisfies the sentences in Σ, we will have that $\mathcal{F}$ satisfies all components of the query Q, other than Q_A. In addition, in any such $\mathcal{F}$, B must be empty, since otherwise tiling predicates for distinct tiles would overlap, which is not the case. Since Init holds, we can conclude via the first sentence above that Q_A must hold.

Conversely, suppose that $\exists \mathsf{PQI}_\infty(Q, \Sigma, \mathbf{S}) = \mathrm{true}$ and let $\mathcal{V}$ be the witnessing visible instance. Clearly, $\mathcal{V}$ contains the fact Init (otherwise, the query would be immediately violated). We can use the content of $\mathcal{V}$ and the knowledge that $\exists \mathsf{PQI}_\infty(Q, \Sigma, \mathbf{S}) = \mathrm{true}$ to inductively construct a correct tiling of the infinite grid. More precisely, by the first sentence in Σ, we know that $\mathcal{V}$ contains the fact $U_{t_\perp}(x)$, for some node x. Accordingly, we define $i_x = 0$, $j_x = 0$, and $f(i_x, j_x) = t_\perp$. For the induction step, suppose that $f(i_x, j_x)$ is defined for a node x with the associated coordinates i_x and j_x. The sentences in Σ enforce the existence of two cells y and z and two tiles t and t' for which the following facts are in the visible instance: $E_H(x, y)$, $E_V(x, z)$, $U_t(y)$, and $U_{t'}(z)$. Accordingly, we let $i_y = i_x + 1$, $j_y = j_x$, $i_z = i_x$, $j_z = j_y + 1$, $f(i_y, j_y) = t$, and $f(i_z, j_z) = t'$. By the initial sentences in Σ, we know that the tiles associated with the new cells (i_y, j_y) and (i_z, j_z) are consistent with the tile in (i_x, j_x) and with the horizontal and vertical constraints H and V. We now argue that there is a unique choice for the nodes y and z. Indeed, suppose that this is not the case; for instance, suppose that there exist two distinct nodes y, y' that are connected to x via E_H. Then, we could construct a full instance in which the relation $\mathsf{HChallenge}_{\mathsf{funct}}$ contains the single triple (x, y, y'). This will immediately violate the CQ Q_H and hence Q. Similar arguments apply to the vertical successor z.

We now argue that there are unique choices for the tile t associated with a node y. Suppose not. Then we can let A be empty and B the set of all nodes with multiple tiles. All sentences in Σ are satisfied, but the query Q_A is not. This contradicts the assumption that we have a PQI.

Finally, we can argue along the same lines that, during the next steps of the induction, the E_V-successor of y and the E_H-successor of z coincide. The above properties are sufficient to conclude that the constructed function f is a correct tiling of the infinite grid.

This concludes the undecidability proof for $\exists \mathsf{PQI}_\infty$. The undecidability of $\exists \mathsf{PQI}$ follows from the following further observation: the same above reduction has the property that $\exists \mathsf{PQI}$ holds if and only if there is a *periodic* tiling of the grid—that is, a tiling that is obtained by iterating a tiling T_F of an n by n grid. In one direction, we observe that if there is a periodic tiling, we can form a finite witness to $\exists \mathsf{PQI}$ by basing E_H, E_V, U_t on the finite tiling T_F, including back edges from n to 1. In the other direction, we take a finite witness to $\exists \mathsf{PQI}$ and observe that the corresponding tiling will be periodic. □

It turns out that *disjunction can be simulated using constants (under UNA)*. The proof works by applying the technique of "coding Boolean operations and truth values in the schema" that has

been used to eliminate the need for disjunction in hardness proofs in several past works (e.g., [34]). It is also similar to the proof idea used in Lemma 4.6 from earlier in this article.

PROPOSITION 4.17. *There is a polynomial time reduction from* $\exists$PQI(Q, Σ, S), *where Q ranges over Boolean UCQs without constants and* Σ *over sets of disjunctive linear TGDs without constants, to* $\exists$PQI$(Q', \Sigma', \mathsf{S}')$, *where Q' ranges over Boolean UCQs without constants and* Σ' *over sets of linear TGDs with constants.*

PROOF. We transform the schema S to a new schema S' as follows. For every visible (respectively, hidden) relation R of S of arity k, we add to S' a corresponding visible (respectively, hidden) relation R' of arity $k + 1$. The idea is that the additional attribute of R' represents a truth value (i.e., either the constant 0 or the constant 1), which indicates the presence of a tuple in the original relation R. For example, the fact $R'(\bar{a}, 1)$ indicates the presence of the tuple $\bar{a}$ in the relation R. We can then simulate the disjunctions in the sentences of Σ by using conjunctions and an appropriate look-up table, which we denote by Or. Formally, we introduce three additional relations Or, Check, and Init, of arities 2, 1, and 0, respectively, and we let Or and Init be visible and Check be hidden in S'. Consider a disjunctive linear TGD in Σ. By normalizing (introducing additional relations if needed), we can assume that these are of the form

$$R(\bar{x}) \; \rightarrow \; \exists \bar{y} \; S(\bar{z}) \vee T(\bar{z}').$$

We add to Σ' the linear TGD with constants

$$R'(\bar{x}, 1) \; \rightarrow \; \exists \bar{y} \; b_1 \; b_2 \; S'(\bar{z}, b_1) \wedge T'(\bar{z}', b_2) \wedge \mathsf{Or}(b_1, b_2).$$

We further add to Σ' the following sentences:

$$\mathsf{Init} \; \rightarrow \; \mathsf{Or}(0, 1) \wedge \mathsf{Or}(1, 0) \wedge \mathsf{Or}(1, 1)$$

$$\mathsf{Init} \; \rightarrow \; \exists b_1 \; b_2 \; \mathsf{Or}(b_1, b_2) \wedge \mathsf{Check}(b_1) \wedge \mathsf{Check}(b_2).$$

Finally, we transform every CQ of Q of the form $\exists \bar{y} \; S(\bar{y})$ to a corresponding CQ of Q' of the form

$$\exists \bar{y} \; S'(\bar{y}, 1) \wedge \mathsf{Check}(1) \wedge \mathsf{Init}.$$

We can further rewrite the CQ above so as to avoid constants: we introduce another hidden unary relation One and the sentence $\mathsf{Init} \; \rightarrow \; \mathsf{One}(1)$, and we replace the conjunct $\mathsf{Check}(1)$ with $\exists b \; \mathsf{Check}(b) \wedge \mathsf{One}(b)$. In the following, we prove that $\exists$PQI$(Q, \Sigma, \mathsf{S}) = $ true if and only if $\exists$PQI$(Q', \Sigma', \mathsf{S}') = $ true.

For the easier direction, we consider a realizable S_v-instance $\mathcal{V}$ such that PQI$(Q, \Sigma, \mathsf{S}, \mathcal{V}) = $ true. We can easily transform $\mathcal{V}$ into a realizable S'_v-instance $\mathcal{V}'$ that satisfies PQI$(Q', \Sigma', \mathsf{S}', \mathcal{V}') = $ true. For this, it suffices to copy the content of the visible relations of $\mathcal{V}$ into $\mathcal{V}'$, by properly expanding the tuples with the constant 1, and then adding the facts Init, Or(0, 1), Or(1, 0), and Or(1, 1).

As for the converse direction, we consider a realizable S'_v-instance $\mathcal{V}'$ such that PQI$(Q', \Sigma', \mathsf{S}', \mathcal{V}') = $ true. By the definition of Q', it is clear that $\mathcal{V}'$ contains the fact Init and hence also the facts Or(0, 1), Or(1, 0), and Or(1, 1). We first claim that it suffices to show that for every fact Or(b_1, b_2) in $\mathcal{V}'$, we have $b_1 = 1$ or $b_2 = 1$. If this were the case, then we could easily transform $\mathcal{V}'$ into a realizable S_v-instance $\mathcal{V}$ that satisfies PQI$(Q, \Sigma, \mathsf{S}, \mathcal{V}) = $ true. For this, we simply select the facts $R'(\bar{a}, 1)$ in $\mathcal{V}'$, where R is a visible relation of S, and project away the constant 1.

Thus, it remains to show that for every fact Or(b_1, b_2) in $\mathcal{V}'$, we have $b_1 = 1$ or $b_2 = 1$. For the sake of contradiction, suppose that $\mathcal{V}'$ contains a fact of the form Or(b_1, b_2), with $b_1 \neq 1$ and $b_2 \neq 1$. Since $\mathcal{V}'$ is realizable, there is a full S'-instance $\mathcal{F}'$ such that $\mathcal{F}' \models \Sigma'$ and Visible$(\mathcal{F}') = \mathcal{V}'$. Note that $\mathcal{F}'$ may satisfy Q' and, in particular, the conjunct Check(1). However, removing the single fact

Check(1) from $\mathcal{F}'$ gives a new instance $\mathcal{F}''$ that still satisfies the sentences in Σ', agrees with $\mathcal{F}'$ on the visible part, and violates the query Q'. This contradicts the fact that $\text{PQI}(Q', \Sigma', S', \mathcal{V}') = $ true. □

Proposition 4.17 shows that disjunctions can be simulated using *constants in the constraints* (without using constants in the query). A variant of the same construction shows that disjunctions can be simulated using *constants in the query* (without using constants in the constraints).

PROPOSITION 4.18. *There is a polynomial time reduction from* $\exists\text{PQI}(Q, \Sigma, S)$, *where Q ranges over Boolean UCQs without constants and Σ over sets of disjunctive linear TGDs without constants, to* $\exists\text{PQI}(Q', \Sigma', S')$, *where Q' ranges over Boolean UCQs with constants and Σ' over sets of linear TGDs without constants.*

PROOF. The proof is similar to that of Proposition 4.17. We transform the schema S to a new schema S' as follows. For every visible relation R of S of arity k, we add to S' a corresponding visible relation R' of arity $k + 1$, and similarly for hidden relations. In addition, S' contains

- A visible ternary relation Or,
- A visible binary relation Neg,
- Hidden unary relations OrCheck and NegCheck,
- A visible zero-ary relation Init.

For each (normalized) disjunctive linear TGD in Σ of the form $R(\bar{x}) \rightarrow \exists\bar{y}\, S(\bar{z}) \vee T(\bar{z}')$, we add to Σ' the linear TGD

$$R'(\bar{x}, b_0) \rightarrow \exists\bar{y}\, b_0'\, b_1\, b_2\; \text{Neg}(b_0, b_0') \wedge S'(\bar{z}, b_1) \wedge T'(\bar{z}', b_2) \wedge \text{Or}(b_0', b_1, b_2).$$

We further add to Σ' the following sentences:

$$\text{Init} \rightarrow \exists b_1\, b_2\; \text{Neg}(b_1, b_2) \wedge \text{NegCheck}(b_1) \wedge \text{NegCheck}(b_2)$$

$$\text{Init} \rightarrow \exists b_1\, b_2\, b_3\; \text{Or}(b_1, b_2, b_3) \wedge \text{OrCheck}(b_1) \wedge \text{OrCheck}(b_2) \wedge \text{OrCheck}_2(b_3).$$

Finally, we transform every CQ Q_i of Q to a corresponding CQ of Q_i' of the form

$$Q_i \wedge \text{Neg}(0, 1) \wedge \text{Neg}(1, 0) \wedge \text{NegCheck}(0) \wedge \text{NegCheck}(1) \wedge \bigwedge_{(a,b,c)\in\{0,1\}^3\setminus\{0,0,0\}} (\text{Or}(a, b, c)) \wedge \text{OrCheck}(1).$$

Let $\mathcal{V}$ be any realizable instance where $\text{PQI}(Q', \Sigma', S')$ holds true. Using similar reasoning as in the proof of Proposition 4.17, we can show the following:

- The visible relation Neg must contain the tuples $(0, 1)$ and $(1, 0)$, and no other tuples (for if it contained any other tuples, one of the conjuncts $\text{NegCheck}(0) \wedge \text{NegCheck}(1)$ in the query could be forced to be false).
- The visible relation Or must contain all tuples $(a, b, c) \in \{0, 1\}^3 \setminus \{0, 0, 0\}$, and every other tuple in Or must contain the constant 1 in at least one position.

The remainder of the proof proceeds in the same way as with Proposition 4.17. □

The above results together yield the following theorem.

THEOREM 4.19. *The problem* $\exists\text{PQI}(Q, \Sigma, S)$ *is undecidable when*

- *Q ranges over Boolean UCQs without constants and Σ ranges over sets of linear TGDs with constants, or*
- *Q ranges over Boolean UCQs with constants and Σ ranges over sets of linear TGDs without constants.*

In other words, Corollary 4.13 fails when constants are allowed in either the query or in the constraints.

There remains the question of the complexity for IDs as well as for linear TGDs without constants. In earlier work [10], a special case of this problem is proven to be PSPACE-complete. We can can easily extend the ideas there to show PSPACE-completeness for the full problem; details are deferred to the appendix.

THEOREM 4.20. *The problem* $\exists PQI(Q, \Sigma, S)$, *where Q ranges over Boolean UCQs without constants and* Σ *over sets of IDs, is* PSPACE-*complete. Hardness holds even in the case of Boolean CQs without constants.*

In the conference paper [12], it was claimed that for more general linear TGDs, this problem was also PSPACE-complete. However, subsequent work provides an argument that this problem is in fact ExpTime-complete.

THEOREM 4.21 ([10]). *The problem* $\exists PQI(Q, \Sigma, S)$, *where Q ranges over Boolean UCQs without constants and* Σ *over sets of linear TGDs without constants, is* ExpTime-*complete. Hardness holds even in the case of Boolean CQs without constants.*

In earlier work [10], the setting and terminology is slightly different. There is a source schema and a target schema. We have constraints on the sources and the target schema is populated from the sources via mappings (views) from source to target. Thus, the relations in the source schema correspond to what we call *invisible relations* here, whereas the relations in the target schema are the visible relations. The ExpTime upper bound derives from Theorem 4 of earlier work [10], where it is only stated for the special case where there are linear TGDs on the sources, and the mappings correspond to full linear TGDs from source to target. A rewriting argument is given that the reduces the problem to OWQ for TGDs of a special form, where the body contains a guard atom and a conjunction of atoms in a fixed "side signature." Earlier work [10] cites an argument in the technical report [3] (Corollary G.5) to infer that this class of OWQ problems is in ExpTime. It is easy to see that the rewriting reduction in the earlier work [10] carries over to the setting with arbitrary linear TGDs. The ExpTime lower bound is implied immediately by Theorem 7 of the earlier work [10], which shows the bound for their special case, where the linear TGDs constrain the sources, and there are also very simple linear TGDs (atomic views) from source to target.

5 NEGATIVE QUERY IMPLICATION

5.1 Instance-Level Problems

Here we analyze the complexity of the problem $NQI(Q, \Sigma, S, \mathcal{V})$. As in the positive case, we obtain a general 2ExpTime upper bound for GNFO background theories.

THEOREM 5.1. *The problem* $NQI(Q, \Sigma, S, \mathcal{V})$, *as Q ranges over Boolean UCQs and* Σ *over sets of GNFO sentences, has* 2ExpTime *combined complexity, has* ExpTime *data complexity, and is finitely controllable.*

PROOF. As in the positive case, we reduce to unsatisfiability of a GNFO formula. We use a variation of the same formula, where $\neg Q$ is now replaced by Q:

$$\phi_{Q, \Sigma, S, \mathcal{V}}^{NQItoGNF} = Q \wedge \Sigma \wedge$$
$$\bigwedge_{R \in S_v} \left(\bigwedge_{R(\bar{a}) \in \mathcal{V}} R(\bar{a}) \wedge \forall \bar{x} \left(R(\bar{x}) \rightarrow \bigvee_{R(\bar{a}) \in \mathcal{V}} \bar{x} = \bar{a} \right) \right)$$

The data complexity analysis is as in Theorem 4.4, since the formulas agree on the part that varies with the instance.   □

We can obtain a matching lower bound by reducing PQI to NQI.

THEOREM 5.2. *For any class of sentences that include connected FGTGDs and for any UCQ Q,* $\text{PQI}(Q, \Sigma, S, \mathcal{V})$ *reduces in polynomial time to* $\text{NQI}(Q', \Sigma', S', \mathcal{V}')$. *When* Q, Σ, S *are fixed in the input to this reduction, then* Q', Σ', S' *are fixed in the output, and when* Q *is a CQ, then* Q' *is a CQ as well.*

PROOF. We first provide a reduction that uses not-necessarily-connected FGTGDs. Subsequently, we show how to modify the constructions to preserve connectedness.

The schema S' is obtained by copying both the visible and the hidden relations from S and by adding the following relations: a visible relation Error of arity 0 and a hidden relation Good of arity 0. The sentences Σ' will contain the sentences from Σ, plus one FGTGD of the form

$$Q_i(\bar{y}) \wedge \text{Good} \;\rightarrow\; \text{Error}$$

for each disjunct $\exists \bar{y}\, Q_i(\bar{y})$ of the UCQ Q. Finally, the query and the visible instance for NQI are defined as follows: $Q' = \text{Good}$ and $\mathcal{V}' = \mathcal{V}$ (in particular, we initialize the visible relation Error with the empty set).

We now verify that $\text{PQI}(Q, \Sigma, S, \mathcal{V}) = \text{false}$ if and only if $\text{NQI}(Q', \Sigma', S', \mathcal{V}') = \text{false}$. Suppose that $\text{PQI}(Q, \Sigma, S, \mathcal{V}) = \text{false}$, namely, that there is an S-instance $\mathcal{F}$ such that $\mathcal{F} \not\models Q$, $\mathcal{F} \models \Sigma$, and $\text{Visible}(\mathcal{F}) = \mathcal{V}$. Let $\mathcal{F}'$ be the S'-instance obtained from $\mathcal{F}$ by adding the single hidden fact Good. Clearly, $\mathcal{F}'$ satisfies the query Q' and also the sentences in Σ'. In particular, it satisfies every sentence $Q_i(\bar{y}) \wedge \text{Good} \rightarrow \text{Error}$ because $\mathcal{F}$ violates every disjunct $\exists \bar{y}\, Q_i$ of Q. Hence, we have $\text{NQI}(Q', \Sigma', S', \mathcal{V}') = \text{false}$. Conversely, suppose that $\text{NQI}(Q', \Sigma', S', \mathcal{V}') = \text{false}$, namely, that there is an S'-instance $\mathcal{F}'$ such that $\mathcal{F}' \models Q'$, $\mathcal{F}' \models \Sigma'$, and $\text{Visible}(\mathcal{F}') = \mathcal{V}'$. By copying the content of $\mathcal{F}'$ for those relations that belong to the schema S, we obtain an S-instance $\mathcal{F}$ that satisfies the sentences Σ. Moreover, because $\mathcal{F}'$ contains the fact Good but not the fact Error, $\mathcal{F}'$ violates every conjunct $\exists \bar{y}\, Q_i(\bar{y})$ of Q, and so does $\mathcal{F}$. This shows that $\text{PQI}(Q, \Sigma, S, \mathcal{V}) = \text{false}$.

We observe that the sentences in the above reduction use left-hand sides that are not connected. To preserve connectedness, it is sufficient to modify the above constructions by adding a dummy variable that is shared among all atoms. More precisely, we expand the relations of the schema S and the relation Good with a new attribute, and we introduce a new visible relation Check of arity 1. The dummy variable will be used to enforce connectedness in the left-hand sides, and the relation Check will gather all values associated with the dummy attribute. Using the visible instance, we can also check that the relation Check contains exactly one value. The sentences in the background theory are thus modified as follows. Every sentence $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \rightarrow \exists \bar{y}\, S(\bar{z})$ in Σ' is transformed into $R_1(\bar{x}_1, w) \wedge \ldots \wedge R_m(\bar{x}_m, w) \rightarrow \exists \bar{y}\, S(\bar{z}, w)$. In particular, note that the sentence $Q_i(\bar{y}) \wedge \text{Good} \rightarrow \text{Error}$ becomes $Q_i(\bar{y}, w) \wedge \text{Good}(w) \rightarrow \text{Error}(w)$, which is now a connected FGTGD. Furthermore, for every relation $R(\bar{x})$ in S, we add the sentence

$$R(\bar{x}, w) \;\rightarrow\; \text{Check}(w),$$

and we do the same for the relation Good:

$$\text{Good}(w) \;\rightarrow\; \text{Check}(w).$$

Finally, the query is transformed into $Q' = \exists w\, \text{Good}(w)$ and the visible instance $\mathcal{V}'$ is expanded with a fresh dummy value a on the additional attribute and with the visible fact $\text{Check}(a)$. □

Note that the above reduction does not introduce constants. Combining the above reduction with Corollary 4.7 and Theorem 4.8, we get the following hardness results for instance-based NQI.

COROLLARY 5.3. *There are a Boolean CQ Q without constants and a set Σ of connected FGTGDs without constants over a schema S for which the problem $\text{NQI}(Q, \Sigma, S, \mathcal{V})$ is* EXPTIME-*hard in data complexity (i.e., as $\mathcal{V}$ varies over instances).*

Corollary 5.4. *The problem* NQI$(Q, \Sigma, \mathsf{S}, \mathcal{V})$, *as* Σ *ranges over sets of connected FGTGDs without constants, and* Q *over CQs without constants, is* 2ExpTime-*hard.*

Thus far, the NQI results are similar to the positive ones. We will now show a strong contrast in the case of IDs and linear TGDs. Recall that the PQI problems were highly intractable even for fixed schema, query, and background theory. We show that NQI is computationally better behaved for such constraints.

We begin by showing that NQI$(Q, \Sigma, \mathsf{S}, \mathcal{V})$ can be solved easily by looking only at full instances that agree with $\mathcal{V}$ on the visible part and whose active domains are (almost) the same as that of $\mathcal{V}$. In what follows, we denote by adom$^+(I)$ the active domain of the (visible or full) instance I extended with the constants appearing in Σ and Q.

Definition 5.5. The problem NQI$(Q, \Sigma, \mathsf{S}, \mathcal{V})$ is said to be *active-domain controllable* over a class of inputs if it is equivalent to asking that for every instance $\mathcal{F}$ with adom$^+(\mathcal{F}) \subseteq$ adom$^+(\mathcal{V})$ if $\mathcal{F}$ satisfies Σ and $\mathcal{V} = $ Visible$(\mathcal{F})$, then $Q(\mathcal{F}) = $ false.

This concept will serve as a stepping stone: we will establish that (a suitable relaxation of) active-domain controllability holds for certain background theories. After that, we show that this implies a polynomial time algorithm for NQI. Indeed, it is intuitively already clear that active-domain controllability makes the problem NQI$(Q, \Sigma, \mathsf{S}, \mathcal{V})$ simpler, as in this case we can guess a full instance $\mathcal{F}$ over adom$^+(\mathcal{V})$ and check whether Q holds on $\mathcal{F}$. In fact, as we will see soon, we can do even better.

We give a simple argument that when we consider NQI under IDs, we can reduce in polynomial time to a class of NQI problems that is active-domain controllable. Let Σ be a set of IDs over a schema S, Q be a UCQ, and $\mathcal{V}$ be a visible instance. Our reduction will simply add an additional visible relation that does not occur in the sentences of the background theory and a dummy visible fact over a visible relation. This reduction does not change the truth of the problem, and it produces a problem instance where adom$^+(\mathcal{V})$ contains at least one element. We now argue that the latter restriction on inputs to NQI yields active-domain controllability.

Suppose that NQI$(Q, \Sigma, \mathsf{S}, \mathcal{V}) = $ false. The fact that NQI$(Q, \Sigma, \mathsf{S}, \mathcal{V}) = $ false implies the existence of a full instance $\mathcal{F}$ such that $\mathcal{F} \models \Sigma$, Visible$(\mathcal{F}) = \mathcal{V}$, and $\mathcal{F} \models Q$. Now take any element $a \in$ adom$^+(\mathcal{V})$ and let h be the homomorphism that is the identity over adom$^+(\mathcal{V})$ and maps any other value from adom$^+(\mathcal{F}) \setminus $ adom$^+(\mathcal{V})$ to a. Since the sentences Σ are IDs (in particular, since the left-hand side atoms do not have constants or repeated occurrences of the same variable), we know that $h(\mathcal{F}) \models \Sigma$. Similarly, we have $h(\mathcal{F}) \models Q$. Hence, $h(\mathcal{F})$ is an instance over adom$^+(\mathcal{V})$ that equally witnesses NQI$(Q, \Sigma, \mathsf{S}, \mathcal{V}) = $ false.

Note that our hardness results for PQI (in particular, Theorem 4.8) imply that PQI is *not* active-domain controllable even for IDs, since such a result would easily give membership in co-NP.

The following example shows that linear TGDs are not always active-domain controllable.

Example 5.6. Let S be the schema with a hidden relation R of arity 2, with two visible relations S, T of arities 1, 0, respectively, and with the sentences:

$$R(x, y) \; \rightarrow \; S(x) \qquad\qquad R(x, x) \; \rightarrow \; T.$$

Note that the sentences are linear TGDs and they are even *full*—no existential quantifiers on the right. The CQ is $Q = \exists x\, y\; R(x, y)$. Further, let the visible instance $\mathcal{V}$ consist of the single fact $S(a)$. Clearly, every full instance $\mathcal{F}$ over the active domain $\{a\}$ that satisfies both Σ and Q must also contain the facts $R(a, a)$ and T, and so such an instance cannot agree with $\mathcal{V}$ in the visible part. However, the instance that contains the facts $S(a)$ and $R(a, b)$, for a fresh value b, satisfies both Σ and Q and moreover agrees with $\mathcal{V}$. This shows that NQI$(Q, \Sigma, \mathsf{S}, \mathcal{V})$ is not active-domain controllable.

The example shows that we need to weaken the notion of active-domain controllability to allow some elements outside of the active domain. The following definition allows a fixed number of exceptions.

Definition 5.7. For a number k, the problem $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$ is said to be *active-domain controllable modulo k* over a class of inputs if it is equivalent to asking that for every instance $\mathcal{F}$ with $|\mathrm{adom}^+(\mathcal{F}) \setminus \mathrm{adom}^+(\mathcal{V})| \leq k$, if $\mathcal{F}$ satisfies Σ and $\mathcal{V} = \mathrm{Visible}(\mathcal{F})$, then $Q(\mathcal{F}) = \mathrm{false}$.

THEOREM 5.8. *Consider a schema S, and let k be the maximal arity of relations in S. Every problem* $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$, *where Σ consists of linear TGDs, is active-domain controllable modulo k.*

PROOF. The main idea is to compress an arbitrary counterexample instance to NQI by one with at most k elements outside the active domain, by taking k "representative elements" outside the active domain and replacing arbitrary tuples outside the active domain with these k elements. In doing this replacement, we should take into account equalities within each tuple.

Formally, given a finite set of constants C, we say that two tuples $t_1 \ldots t_n$ and $t_1' \ldots t_n'$ of the same length are *equality-equivalent for C* if $t_i = t_j$ if and only if $t_i' = t_j'$ and for every constant $c \in C$, $t_i = c$ if and only if $t_i' = c$. For example, suppose that $C = \{1\}$. Then the tuples $\langle 1, 1, 2, 4 \rangle$ and $\langle 1, 1, 3, 5 \rangle$ are equality-equivalent.

Suppose that $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V}) = \mathrm{false}$, namely, that there is an S-instance $\mathcal{F}$ such that $\mathcal{F} \models \Sigma$, $\mathcal{F} \models Q$, and $\mathrm{Visible}(\mathcal{F}) = \mathcal{V}$. We need to construct an instance $\mathcal{F}'$ whose active domain has only k elements outside the active domain of $\mathcal{V}$ that witnesses $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V}) = \mathrm{false}$.

We fix an extension $\mathbb{D}$ of the active domain of $\mathcal{V}$ that contains k additional fresh values, and let C be the set of constants occurring in Σ or Q. Recall that for a relation R, $\mathrm{arity}(R)$ denotes the arity of the relation. For each fact $R(\bar{a})$ in $\mathcal{F}$ and each tuple $\bar{b} \in \mathbb{D}^{\mathrm{arity}(R)}$, if $\bar{b}$ and $\bar{a}$ are equality-equivalent over C and also agree on each position whose value is in the active domain of $\mathcal{V}$, then we add the fact $R(\bar{b})$ to $\mathcal{F}'$. By definition, the instance $\mathcal{F}'$ agrees with $\mathcal{F}$ on the visible part and has only k elements outside the active domain of $\mathcal{V}$.

In the following, we show that $\mathcal{F}'$ satisfies the sentences of Σ and the query Q. Consider any linear TGD τ of Σ of the form

$$R(\bar{x}) \rightarrow \exists \bar{y}\, S(\bar{z})$$

and any fact $R(\bar{a})$ that is the image under some homomorphism h of the left-hand side atom $R(\bar{x})$. Let I be the set of positions $i \in \{1, \ldots, \mathrm{arity}(R)\}$ such that $\bar{a}(i) \in \mathrm{adom}^+(\mathcal{V})$. We know that there is $\bar{u}$ such that $R(\bar{u})$ holds in $\mathcal{F}$ such that $\bar{a}|I = \bar{u}|I$ and $\bar{a}$ is equality-equivalent to $\bar{u}$. Here, $\bar{a}|I$ denotes the restriction of the tuple a to the positions in I. Since $\mathcal{F}$ satisfies τ, and $\bar{u}$ is equality-equivalent to $\bar{a}$, we know that there is a fact $S(\bar{v})$ in $\mathcal{F}$ agreeing with $\bar{u}$ on the positions corresponding to exported variables of τ. Let $\bar{b}$ be any tuple in $\mathbb{D}^{\mathrm{arity}(R)}$ equality-equivalent to $\bar{a}$ and agreeing with $\bar{v}$ on all the positions corresponding to exported variables of τ. Since k is at least the arity of R, such a $\bar{b}$ must exist. Then $\bar{b}$ witnesses that τ holds for $\bar{a}$. This completes the proof that the sentences of Σ hold.

A similar argument shows that Q holds in $\mathcal{F}'$. Thus, $\mathcal{F}'$ witnesses that $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$ is active-domain controllable modulo k. □

Example 5.9. As an example of the prior argument, consider a TGD τ

$$R(x, y, y) \rightarrow \exists z\, S(y, z, z),$$

and suppose that the instance $\mathcal{F}$ has a tuple $R(a, b, b)$, where a is in the active domain of the visible instance and b is outside of the active domain of the visible instance. Thus, there is a homomorphism from the left side of τ to $R(a, b, b)$. Since $\mathcal{F}$ satisfies τ, it must contain $S(b, c, c)$ for some value c.

The instance $\mathcal{F}'$ produced by the prior argument will replace $R(a, b, b)$ by $R(a, c_1, c_1)$, where c_1 is one of the k additional constants. We explain why this replacement will not break the satisfaction of τ. There is a homomorphism h' of the left-hand side of τ to $R(a, c_1, c_1)$. If the witness c of $S(b, c, c)$ is in the active domain of the visible instance, then $\mathcal{F}'$ has $S(c_1, c, c)$, and thus we have the witness we need for τ with respect to h'. If c is not in the active domain of the visible instance, then $\mathcal{F}'$ will also have $S(c_1, c_2, c_2)$, for c_2 another of the additional constants. Either way, the required value is present.

Now we show how to exploit active-domain controllability and its modulo k variant to prove that NQI problems can be solved not only efficiently but also "definably" using well-behaved query languages. For this, we introduce a variant of Datalog programs, called *GFP-Datalog* programs, whose semantics is given by greatest fixpoints. GFP-Datalog programs are defined syntactically in the same way as Datalog programs [2]—that is, as finite sets of rules of the form $U(\bar{x}, \bar{c}) \leftarrow Q(\bar{x})$ where the heads can contain variables or constants, with the variables $\bar{x}$ being implicitly universally quantified. Q is a CQ whose free variables are exactly $\bar{x}$. As for Datalog programs, we distinguish between *extensional* (i.e., input) predicates and *intensional* (i.e., output) predicates. In the above rules, we restrict the left-hand sides to contain only intensional predicates. Given a GFP-Datalog program P, the *immediate consequence operator* for P is the function that, given an instance M consisting of both extensional and intensional relations, returns the instance M' where the extensional relations are as in M and the tuples of each intensional relation U are those satisfying Q_U in M, where Q_U is any rule body appearing on the right of a rule with U. The immediate consequence operator is monotone, and the usual semantics of Datalog is defined as its least fixpoint. The semantics of the GFP-Datalog program on instance I for the extensional relations is defined as the greatest fixpoint of this operator starting at the instance I^+ that extends I by setting each intensional relation "maximally"—that is, to the tuples of values from the active domain of I plus the constants appearing in the GFP-Datalog program. A program may also include a distinguished intensional predicate, the *goal predicate* G, in which case it defines the query that maps every instance to the set of tuples satisfying G in the greatest fixpoint. We now show that under active-domain controllability, we can use GFP-Datalog to decide $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$.

THEOREM 5.10. *If Q is a Boolean UCQ, Σ is a set of linear TGDs, and $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$ is active-domain controllable, then $\neg\mathrm{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$, viewed as a Boolean query over the visible part $\mathcal{V}$, is definable by a GFP-Datalog program that can be constructed in* PTIME *from Q, Σ, and S.*

PROOF. First observe that $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, -)$ can be seen as a Boolean function that takes as input an instance $\mathcal{V}$ for the visible relations of S and returns true if and only if the query Q does *not* hold on *every* instance $\mathcal{F}$ that satisfies the sentences Σ and such that Visible$(\mathcal{F}) = \mathcal{V}$. Accordingly, $\neg\mathrm{NQI}(Q, \Sigma, \mathsf{S}, -)$ is the negation of the function $\mathrm{NQI}(Q, \Sigma, \mathsf{S}, -)$ and thus maps an instance $\mathcal{V}$ to true when Q *does* hold on *some* instance $\mathcal{F}$ that satisfies Σ and agrees with $\mathcal{V}$ on the visible relations.

In the following, we implement the function $\neg\mathrm{NQI}(Q, \Sigma, \mathsf{S}, -)$ by means of a GFP-Datalog program. Thanks to active-domain controllability, it is sufficient to consider only full instances constructed over the active domain of $\mathcal{V}$. More precisely, it is sufficient to show that a witnessing instance $\mathcal{F}$ can be obtained as a greatest fixpoint starting from the values in the active domain of $\mathcal{V}$. We also describe the GFP-Datalog program that computes $\mathcal{F}$ starting from $\mathcal{V}$.

The extensional relations are the ones in the visible part $\mathcal{V}$, whereas the intensional relations are the ones in the hidden part of the schema S, plus an extra intensional relation A that collects the values in adom$^+(\mathcal{V})$. For each extensional (i.e., visible) relation R and each position $i \in \{1, \ldots, \mathrm{arity}(R)\}$, we add the rule $A(x_i) \leftarrow R(\bar{x})$, which collects all values of the active

domain into the relation A. We also have rules that put constants in A as well. In addition, for each intensional (i.e., hidden) relation R, we have the rule

$$R(\bar{x}) \leftarrow \bigwedge_{i} A(x_i) \wedge \bigwedge_{\substack{\text{linear TGD in } \Sigma \text{ of the} \\ \text{form } R(\bar{x}) \to \exists \bar{y} \, S(\bar{z})}} S(\bar{z}).$$

Intuitively, the above rule permits the existence of a fact $R(\bar{a})$ only when $\bar{a}$ consists of values from the active domain and every linear TGD $R(\bar{x}) \to \exists \bar{y} \, S(\bar{z})$ of Σ is satisfied by some fact $S(\bar{b})$ when substituting $\bar{x}$ for $\bar{a}$. This semantics is consistent with the goal of finding the biggest instance $\mathcal{F}$ over the active domain of $\mathcal{V}$ that satisfies the UCQ Q—so as to have $\mathsf{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V}) = \mathsf{false}$ —while guaranteeing that the linear TGDs remain valid.

We finally add the rule

$$\text{Goal} \leftarrow S_1(\bar{z}_1) \wedge \ldots \wedge S_n(\bar{z}_n)$$

for each CQ $\exists \bar{y} \, S_1(\bar{z}_1) \wedge \ldots \wedge S_n(\bar{z}_n)$ of Q and take Goal to be the final output of our program.

Let us now prove that the Datalog program does compute the function $\neg\mathsf{NQI}(Q, \Sigma, \mathsf{S}, -)$ under the greatest fixpoint semantics. Consider an instance $\mathcal{F}$ computed by the GFP-Datalog program starting from input $\mathcal{V}$. Clearly, the extensional (visible) part of $\mathcal{F}$ agrees with $\mathcal{V}$. We claim that $\mathcal{F}$ also satisfies the sentences in Σ. Indeed, if $R(\bar{x}) \to \exists \bar{y} \, S(\bar{z})$ is a linear TGD in Σ and $R(\bar{a})$ is a fact of $\mathcal{F}$, with $R(\bar{a})$ image of $R(\bar{x})$ via some homomorphism h, then $\mathcal{F}$ contains a fact of the form $S(\bar{b})$, where $\bar{b}$ is the image of $S(\bar{z})$ via some homomorphism h' that extends h. To conclude, we observe that the predicate Goal holds if and only if $\mathcal{F}$ satisfies some disjunct $S_1(\bar{z}_1) \wedge \ldots \wedge S_n(\bar{z}_n)$ of the UCQ Q, namely, if and only if $\mathsf{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V}) = \mathsf{false}$. □

In the case of linear TGDs that are active-domain controllable modulo k, we can similarly use a GFP Datalog program, but first pre-processing the active domain to contain the k additional constants. The extension of Theorem 5.10 clearly holds.

THEOREM 5.11. *For every k, if Q is a Boolean UCQ and Σ a set of linear TGDs (possibly with constants) from a class where $\mathsf{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$ is active-domain controllable modulo k, then $\mathsf{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$ can be computed by evaluating a GFP program P on $\mathcal{V}'$, where $\mathcal{V}'$ is an instance that can be computed in* PTIME *from $\mathcal{V}$, whereas P is a GFP-Datalog program that can be constructed in* PTIME *from Q, Σ, and S.*

Recall that the naïve fixpoint algorithm for a GFP-Datalog program takes exponential time in the maximum arity of the intensional relations, but only polynomial time in the size of the extensional relations and the number of rules. This is true even if one extends the active domain by k elements, where k is the maximal arity. Thus, we can get bounds on the NQI problem for IDs using the simple argument for active-domain controllability for IDs given above along with Theorem 5.10. We can likewise get bounds for linear TGDs using Theorem 5.8 and Theorem 5.11.

COROLLARY 5.12. *When Σ ranges over sets of linear TGDs and Q over Boolean UCQs, $\mathsf{NQI}(Q, \Sigma, \mathsf{S}, \mathcal{V})$ has data complexity in* PTIME *and combined complexity in* EXPTIME.

Example 5.13. Returning to the medical example from the introduction, Example 1.1, we see that the GFP-Datalog program is quite intuitive: since we have an ID from Appointment into Patient and the visible instance does not contain the fact Patient(Smith), all tuples of the form (Smith, a, d) are removed from the relation Appointment. The program then simply evaluates the query on the resulting instance, which returns false, indicating that an NQI does hold on the original visible instance.

We give a tight EXPTIME lower bound for the combined complexity of NQI with linear TGDs (and even IDs).

THEOREM 5.14. *The combined complexity of* $\mathrm{NQI}(Q, \Sigma, S, \mathcal{V})$*, where Q ranges over Boolean UCQs without constants and Σ ranges over IDs, is* EXPTIME-*hard.*

PROOF. We reduce the acceptance problem for an alternating PSPACE Turing machine M to $\mathrm{NQI}(Q, \Sigma, S, \mathcal{V})$. As in the proof of Theorem 4.5, we assume that the transition function of M maps each universal configuration to a set of exactly two target configurations. Moreover, we assume that there is at least one target configuration for each existential configuration. In particular, M never halts. The computation begins with the head on the second position and never visits the first and last position of the tape. The acceptance condition of M is defined by distinguishing two special control states, q_{acc} and q_{rej}, that once reached will "freeze" M in its current configuration. We say that M accepts (the empty input) if for all paths in the computation tree, the state q_{acc} is eventually reached; otherwise, we say that M rejects.

Differently from the proofs of Theorem 4.5 and Theorem 4.8, the configurations of M can be described by simply specifying the label of each cell of the tape, the position of the head, and the control state of the Turing machine M. We thus define *cell values* as elements of $V = (\Sigma \times Q) \uplus \Sigma$, where Σ is the alphabet of M and Q is the set of its control states. If a cell has value (a, q), this means that the associated letter is a, the control state of M is q, and the head is on this cell. Otherwise, if a cell has value a, this means that the associated letter is a and the head of M is not on this cell.

Now, let n be the size of the tape of M. We begin by describing the initial configuration of M. This is encoded by a visible relation C_0 of arity $n + 1$, where the first attribute gives the identifier of the initial configuration and the remaining n attributes give the values of the tape cells. As the relation C_0 is visible, we can immediately fix its content to be a singleton consisting of the tuple $(x_0, y_1, y_2, y_3, \ldots, y_n)$, where x_0 is the identifier of the initial configuration, $y_1 = \bot$, $y_2 = (\bot, q_0)$, $y_3 = \cdots = y_n = \bot$. As for the other configurations of M, we store them into two distinct hidden relations $C^{\exists}$ and $C^{\forall}$, depending on whether the control states are existential or universal. Each fact in one of these two relation consists of $n + 1$ attributes, where the first attribute specifies an identifier and the remaining n attributes specify the cell values. We can immediately give the first sentence, which requires the initial configuration to be existential and stored also in the relation $C^{\exists}$:

$$C_0(x, y_1, \ldots, y_n) \;\rightarrow\; C^{\exists}(x, y_1, \ldots, y_n).$$

To represent the computation tree of M, we encode pairs of subsequent configurations. In doing so, we not only store the identifiers of the configurations but also their contents, in such a way that we can later check the correctness of the transitions using IDs. We use different relations to record whether the current configuration is existential or universal and, in the latter case, whether the successor configuration is the first or the second one in the transition set (recall that the transition rules of M define exactly two successor configurations from each universal configuration). Formally, we introduce three hidden relations $S^{\exists}$, $S_1^{\forall}$, and $S_2^{\forall}$, all of arity $2n + 2$. We can easily enforce that the first $n + 1$ and the last $n + 1$ attributes in every tuple of $S^{\exists}$, $S_1^{\forall}$, and $S_2^{\forall}$ describe configurations in $C^{\exists}$ and $C^{\forall}$:

$$
\begin{array}{llll}
S^{\exists}(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\exists}(x, \bar{y}) \qquad\qquad & S^{\exists}(x, \bar{y}, x', \bar{y}') \rightarrow C^{\exists}(x', \bar{y}') \\[4pt]
S_1^{\forall}(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\forall}(x, \bar{y}) & S_1^{\forall}(x, \bar{y}, x', \bar{y}') \rightarrow C^{\forall}(x', \bar{y}') \\[4pt]
S_2^{\forall}(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\forall}(x, \bar{y}) & S_2^{\forall}(x, \bar{y}, x', \bar{y}') \rightarrow C^{\forall}(x', \bar{y}').
\end{array}
$$

Similarly, we guarantee that every existential (respectively, universal) configuration has one (respectively, two) successor configuration(s) in $S^{\exists}$ (respectively, $S_1^{\forall}$ and $S_2^{\forall}$):

$$C^{\exists}(x, \bar{y}) \quad \rightarrow \quad \exists\, x'\, \bar{y}'\, S^{\exists}(x, \bar{y}, x', \bar{y}')$$
$$C^{\forall}(x, \bar{y}) \quad \rightarrow \quad \exists\, x'\, \bar{y}'\, S_1^{\forall}(x, \bar{y}, x', \bar{y}')$$
$$C^{\forall}(x, \bar{y}) \quad \rightarrow \quad \exists\, x'\, \bar{y}'\, S_2^{\forall}(x, \bar{y}, x', \bar{y}').$$

We now turn to explaining how we can enforce the correctness of the transitions represented in the relations $S^{\exists}$, $S_1^{\forall}$, and $S_2^{\forall}$. Compared to the proof of Theorem 4.5, the goal is simpler in this setting, as we can simply compare the values z_{-1}, z_0, z_{+1} for the cells at positions $i-1, i, i+1$ in a configuration with the value z' for the cell at position i in the successor configuration. We thus introduce new visible relations $N^{\exists}$, $N_1^{\forall}$, and $N_2^{\forall}$ of arity 4. Each of these relations is initialized with the possible quadruples of cell values z_{-1}, z_0, z_{+1}, z' that are allowed by the transition function of M. Consider, for example, the case where the transition function specifies that when M is in the universal control state q and reads the letter a, then the first of the two subcomputations spawned by M begins by rewriting a with a', moving the head to the left, and switching to control state q'. In this case, we add to $N_1^{\forall}$ all tuples of the form $(a_{-1}, (a, q), a_{+1}, a')$ or $(a_{-2}, a_{-1}, (a, q), (a_{-1}, q'))$, with $a_{-2}, a_{-1}, a_{+1} \in \Sigma$. Accordingly, we introduce the following IDs, for all $1 < i < n$:

$$S^{\exists}(x, \bar{y}, x', \bar{y}') \quad \rightarrow \quad N^{\exists}(y_{i-1}, y_i, y_{i+1}, y_i')$$
$$S_1^{\forall}(x, \bar{y}, x', \bar{y}') \quad \rightarrow \quad N_1^{\forall}(y_{i-1}, y_i, y_{i+1}, y_i')$$
$$S_2^{\forall}(x, \bar{y}, x', \bar{y}') \quad \rightarrow \quad N_2^{\forall}(y_{i-1}, y_i, y_{i+1}, y_i').$$

Furthermore, we constrain the values of the extremal cells to never change:

$S^{\exists}(x, \bar{y}, x', \bar{y}')$	$\rightarrow$	$E(y_1, y_1')$	$S^{\exists}(x, \bar{y}, x', \bar{y}')$	$\rightarrow$	$E(y_n, y_n')$
$S_1^{\forall}(x, \bar{y}, x', \bar{y}')$	$\rightarrow$	$E(y_1, y_1')$	$S_1^{\forall}(x, \bar{y}, x', \bar{y}')$	$\rightarrow$	$E(y_n, y_n')$
$S_2^{\forall}(x, \bar{y}, x', \bar{y}')$	$\rightarrow$	$E(y_1, y_1')$	$S_2^{\forall}(x, \bar{y}, x', \bar{y}')$	$\rightarrow$	$E(y_n, y_n')$,

where E is another visible binary relation interpreted by the singleton instance $\{(\bot, \bot)\}$.

It remains to specify the query that checks that the Turing machine M reaches the rejecting state q_{rej} along some path of its computation tree. For this, we introduce a last visible relation V_{rej} that contains all cell values of the form (a, q_{rej}), with $a \in \Sigma$. The query that checks this property is

$$Q = \bigvee_{1 < i < n} \exists\, x\, \bar{y}\, \big(C^{\exists}(x, \bar{y}) \,\wedge\, V_{\text{rej}}(y_i) \big).$$

Let $\mathcal{V}$ be the instance that captures the intended semantics of the visible relations V, C_0, $N^{\exists}$, $N_1^{\forall}$, $N_2^{\forall}$, E, and V_{rej}, The proof that $\text{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \text{true}$ if and only if M accepts (namely, has a computation tree where all paths visit the control state q_{acc}) goes along the same lines of the proof of Theorem 4.5. □

5.2 Existence Problems

Here we consider the complexity of the schema-level question, $\exists \text{NQI}(Q, \Sigma, \mathbf{S})$. We first give a result that holds whenever the background theories Σ that are *preserved under disjoint unions*: whenever $\mathcal{F}$ and $\mathcal{F}'$ are instances satisfying Σ, with the active domain of $\mathcal{F}$ disjoint from the active domain of $\mathcal{F}'$, then the instance $\mathcal{F} \cup \mathcal{F}'$ obtained by just unioning all of the facts still satisfies Σ. An example of sentences with this property are *connected disjunctive TGDs without constants*: if the left-hand side of such a disjunctive TGD is satisfied in $\mathcal{F} \cup \mathcal{F}'$, it must be satisfied either in $\mathcal{F}$ or in $\mathcal{F}'$.

We will show that for theories with this property, the existence of an NQI can be checked by considering a single "negative critical instance," namely the empty visible instance $\emptyset$. This instance is easily seen to be realizable for background theories defined by TGDs: the variant of the chase procedure that we introduced in Section 4.3 terminates immediately when initialized with the

empty instance $\mathcal{F}_0 = \emptyset$ and returns the singleton collection $\mathrm{Chases}_{\mathrm{vis}}(\Sigma, \mathbf{S}, \emptyset)$ consisting of the empty S-instance satisfying Σ.

Theorem 5.15. *If Q is a Boolean UCQ without constants and Σ is a background theory that is preserved under disjoint unions of instances, then $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S}) = \mathrm{true}$ if and only if $\mathrm{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathrm{true}$.*

Proof. It is immediate to see that $\mathrm{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathrm{true}$ implies that $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S}) = \mathrm{true}$. We prove the converse implication by contraposition.

Suppose that $\mathrm{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathrm{false}$, namely, that there is an S-instance $\mathcal{F}$ satisfying Σ and Q and such that $\mathrm{Visible}(\mathcal{F}) = \emptyset$. We aim at proving that $\mathrm{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathrm{false}$ for all realizable visible instances $\mathcal{V}$. Let $\mathcal{V}$ be such a realizable instance and let $\mathcal{F}'$ be an S-instance that satisfies Σ and such that $\mathrm{Visible}(\mathcal{F}') = \mathcal{V}$. We define the new instance $\mathcal{F}''$ as a disjoint union of $\mathcal{F}$ and $\mathcal{F}'$. Since the background theory Σ is preserved under disjoint unions, $\mathcal{F}''$ satisfies Σ. Moreover, $\mathcal{F}''$ satisfies the query Q, by monotonicity of UCQs. Since $\mathcal{V} = \mathrm{Visible}(\mathcal{F}') = \mathrm{Visible}(\mathcal{F}'')$, we have $\mathrm{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathrm{false}$. Finally, since $\mathcal{V}$ was chosen in an arbitrary way, this proves that $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S}) = \mathrm{false}$. □

Using the "negative critical instance" result above and Theorem 5.1, we immediately get the following corollary.

Corollary 5.16. $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S})$ *is decidable in* 2ExpTime *for Boolean UCQs without constants and GNFO sentences that are closed under disjoint unions (in particular, for connected disjunctive FGTGDs without constants).*

Combining this result with Corollary 5.12 also gives an ExpTime bound for linear TGDs without constants, for Boolean UCQs without constants. In fact, we can improve this upper bound by observing that the NQI problem over the empty visible instance reduces to classical OWQ.

Proposition 5.17. *For any Boolean UCQ $Q = \bigcup_i Q_i$, $\mathrm{NQI}(Q, \Sigma, \mathbf{S}, \emptyset)$ holds if and only if $\mathrm{OWQ}(Q', \Sigma, \mathrm{CanonInst}(Q_i))$ holds for each Q_i, where*

$$Q' = \bigvee_{R \in \mathbf{S}_v} \exists \bar{x} \, R(\bar{x})$$

and $\mathrm{CanonInst}(Q_i)$ is the canonical instance of the CQ Q_i.

Proof. Suppose that $\mathrm{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathrm{true}$. This means that every S-instance that satisfies the sentences in Σ and has empty visible part must violate each query Q_i. By contraposition, every S-instance that satisfies the sentences Σ and contains $\mathrm{CanonInst}(Q_i)$ (i.e., satisfies Q_i), must contain some visible facts and hence satisfy the UCQ Q'. This implies that $\mathrm{OWQ}(Q', \Sigma, \mathrm{CanonInst}(Q_i)) = \mathrm{true}$.

The proof that $\mathrm{OWQ}(Q', \Sigma, \mathrm{CanonInst}(Q_i)) = \mathrm{true}$ for all i implies that $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathrm{true}$ follows symmetric arguments. □

We know from previous results [8] that OWQ for Boolean UCQs (without constants) and linear TGDs (without constants) is in PSpace. From the above reduction, we immediately get that the problem $\mathrm{NQI}(Q, \Sigma, \mathbf{S}, \emptyset)$, and hence (by Theorem 5.15) the problem $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S})$, for a set of linear TGDs without constants is also in PSpace.

Corollary 5.18. *The problem $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S})$, as Q ranges over Boolean UCQs without constants and Σ over sets of linear TGDs without constants, is in PSpace.*

To conclude our upper bounds, we show that $\exists \mathrm{NQI}$ can be solved in PTime for IDs.

THEOREM 5.19. *The problem* $\exists NQI(Q, \Sigma, S)$ *is in* PTIME *when Q ranges over Boolean UCQs without constants and* Σ *ranges over sets of IDs.*

The informal explanation for the distinction with linear TGDs is that when Σ contains linear TGDs, there are reductions in both directions between $\exists NQI$ and OWQ problems with respect to Σ for an arbitrary Q: the reduction in one direction is above, and Theorem 5.21 below will provide a reduction in the other direction. But in the case where Σ has only IDs, then $\exists NQI$ reduces to OWQ problems over Σ where the query is extremely simple, of the form $\exists \bar{x} \, R(\bar{x})$ where the x_i are all distinct. Such query answering problems can be solved in PTIME. We now formalize this.

PROOF OF THEOREM 5.19. By Theorem 5.15, $\exists NQI(Q, \Sigma, S) = $ true holds if and only if $NQI(Q, \Sigma, S, \emptyset) = $ true. Note that the latter holds if and only if $NQI(Q', \Sigma, S, \emptyset) = $ true for all CQs Q' in Q. In what follows, we may therefore restrict attention to a single CQ Q'. Furthermore, by Proposition 5.22 together with finite controllability, $NQI(Q', \Sigma, S, \emptyset) = $ true holds if and only if either Q' contains a visible atom or $\mathrm{Chases_{vis}}(\Sigma, S, \mathrm{CanonInst}(Q)) = \emptyset$. Clearly, we can test in polynomial time if Q' contains a visible atom. It remains to show that we can test in polynomial time if $\mathrm{Chases_{vis}}(\Sigma, S, \mathrm{CanonInst}(Q)) = \emptyset$. Note that $\mathrm{Chases_{vis}}(\Sigma, S, \mathrm{CanonInst}(Q))$ is guaranteed to contain at most one instance (because there are no visible facts in $\mathrm{CanonInst}(Q)$).

We define a directed graph over the set of relation symbols as follows: there is a directed edge from relation R to relation S if there is a ID in Σ containing R in its left-hand side and containing S in its right-hand side.

It is easy to show that the following are equivalent:

(1) $\mathrm{Chases_{vis}}(\Sigma, S, \mathrm{CanonInst}(Q))$
(2) CanonInst contains a fact such that there is a directed path from the relation of that fact (in the above graph) to a visible relation.

This places the problem in PTIME, as it suffices to evaluate a Boolean combination of directed graph reachability statements. □

We now turn to lower bounds. We first show that the upper bounds for connected FGTGDs and linear TGDs without constants are tight.

THEOREM 5.20. $\exists NQI(Q, \Sigma, S)$ *is* 2EXPTIME-*hard as Q ranges over Boolean CQs without constants and* Σ *over sets of connected FGTGDs without constants.*

THEOREM 5.21. $\exists NQI(Q, \Sigma, S)$ *is* PSPACE-*hard as Q ranges over Boolean CQs without constants and* Σ *over sets of linear* TGDs *without constants.*

The first theorem will be proven by reducing the OWQ problem to $\exists NQI$, and then applying a prior 2EXPTIME-hardness result from Calì et al. [23]. The PSPACE lower bound will be shown by a reduction from the implication problem for IDs, shown PSPACE-hard by Casanova et al. [24].

We begin with the reduction from OWQ. To prove this reduction, we first provide a characterization of the NQI problem over the empty visible instance, which is based, like Proposition 4.10, on our chase procedure.

PROPOSITION 5.22. *If Q is a Boolean CQ and* Σ *is a set of TGDs without constants over a schema* S, *then* $NQI_\infty(Q, \Sigma, S, \emptyset) = $ true *if and only if either Q contains a visible atom or it does not and in this case* $\mathrm{Chases_{vis}}(\Sigma, S, \mathrm{CanonInst}(Q)) = \emptyset$.

PROOF. Suppose that Q does not contain visible atoms and $\mathrm{Chases_{vis}}(\Sigma, S, \mathrm{CanonInst}(Q))$ is nonempty. Let K be some instance in $\mathrm{Chases_{vis}}(\Sigma, S, \mathrm{CanonInst}(Q))$, and observe that, by construction, K satisfies the sentences in Σ and the query Q and has the same visible part as $\mathrm{CanonInst}(Q)$, which is empty. This means that K is a witness of the fact that $NQI(Q, \Sigma, S, \emptyset) = $ false.

Conversely, suppose that $\mathsf{NQI}_\infty(Q, \Sigma, \mathsf{S}, \emptyset) = \text{false}$ This means that there is an S-instance $\mathcal{F}$ with no visible facts that satisfies the sentences in Σ and the query Q. Since $\mathcal{F} \models Q$, there is a homomorphism g from $\mathsf{CanonInst}(Q)$ to $\mathcal{F}$. Moreover, since Q contains no visible atoms, the two instances $\mathcal{F}$ and $\mathsf{CanonInst}(Q)$ agree on the visible part. By Lemma 4.9, letting $\mathcal{F}_0 = \mathsf{CanonInst}(Q)$, we get the existence of an instance K in $\mathsf{Chases}_{\mathsf{vis}}(\Sigma, \mathsf{S}, \mathsf{CanonInst}(Q))$.   □

PROPOSITION 5.23. *There is a polynomial time reduction from the OWQ problem over a set of connected FGTGDs without constants and a connected Boolean CQ without constants to an $\exists \mathsf{NQI}$ problem over a set of connected FGTGDs without constants and a Boolean CQ without constants.*

PROOF. Consider the OWQ problem over a schema S, a set Σ of sentences without constants and closed under disjoint union, a Boolean CQ Q, and an S-instance $\mathcal{F}$. We reduce this problem to an $\exists \mathsf{NQI}$ problem over a new schema S', a new set of sentences Σ', and a new Boolean CQ Q'. The schema S' is obtained from S by adding a relation Good of arity 0, which is assumed to be the only visible relation in S'. The set of sentences Σ' is equal to Σ unioned with the sentence

$$S_1(\bar{x}_1) \wedge \ldots \wedge S_m(\bar{x}_m) \ \rightarrow \ \mathsf{Good},$$

where $S_1(\bar{x}_1), \ldots, S_m(\bar{x}_m)$ are the atoms in the CQ Q. The query Q' is defined as the *canonical query* of the instance $\mathcal{F}$, obtained by replacing each value v with a variable y_v and by quantifying existentially over all of these variables. Note that $\mathsf{CanonInst}(Q')$ is isomorphic to the input instance $\mathcal{F}$.

Now, assume that the original sentences in Σ were connected FGTGDs and the CQ Q was also connected. By construction, the sentences in Σ' turn out to be also connected FGTGDs. In particular, the satisfiability of these sentences are preserved under disjoint unions, and hence from Theorem 5.15, $\exists \mathsf{NQI}(Q', \Sigma', \mathsf{S}') = \text{true}$ if and only if $\mathsf{NQI}(Q', \Sigma', \mathsf{S}', \emptyset) = \text{true}$. Thus, it remains to show that $\mathsf{NQI}(Q', \Sigma', \mathsf{S}', \emptyset) = \text{true}$ if and only if $\mathsf{OWQ}(Q, \Sigma, \mathcal{F}) = \text{true}$.

By contraposition, suppose that $\mathsf{OWQ}(Q, \Sigma, \mathcal{F}) = \text{false}$. This means that there is a S-instance $\mathcal{F}'$ that contains $\mathcal{F}$, satisfies the sentences in Σ, and violates the query Q. In particular, $\mathcal{F}'$, seen as an instance of the new schema S', without the visible fact Good, satisfies the query Q' and the sentences in Σ' (including the sentence that derives Good from the satisfiability of Q). The S'-instance $\mathcal{F}'$ thus witnesses the fact that $\mathsf{NQI}(Q', \Sigma', \mathsf{S}', \emptyset) = \text{false}$.

Conversely, suppose that $\mathsf{NQI}(Q', \Sigma', \mathsf{S}', \emptyset) = \text{false}$. By finite controllability (Theorem 5.1), we also have that $\mathsf{NQI}_\infty(Q', \Sigma', \mathsf{S}', \emptyset) = \text{false}$. Recall that the sentences in Σ' do not use constants and Q' contains no visible facts. We can thus apply Proposition 5.22 and derive $\mathsf{Chases}_{\mathsf{vis}}(\Sigma', \mathsf{S}', \mathsf{CanonInst}(Q')) \neq \emptyset$. Note that $\mathsf{CanonInst}(Q')$ is clearly isomorphic to the original instance $\mathcal{F}$. In particular, there is an instance K in $\mathsf{Chases}_{\mathsf{vis}}(\Sigma', \mathsf{S}', \mathsf{CanonInst}(Q'))$ that contains the original instance $\mathcal{F}$, satisfies the sentences in Σ', and does not contain the visible fact Good. From the latter property, we derive that K violates the query Q. Thus, K, seen as an instance of the schema S, witnesses the fact that $\mathsf{OWQ}(Q, \Sigma, \mathcal{F}) = \text{false}$.   □

We are now ready to prove Theorem 5.20, namely, the 2ExpTime-hardness of the problem $\exists \mathsf{NQI}(Q, \Sigma, \mathsf{S})$, where Q ranges over Boolean CQs and Σ ranges over sets of connected FGTGDs.

PROOF OF THEOREM 5.20. Theorem 6.2 of Calì et al. [23] shows 2ExpTime-hardness of OWQ for FGTGDs. We note that there are two variants of OWQ, corresponding to finite and infinite instances. However, by finite controllability of FGTGDs, inherited from the finite model property of GNFO (see Theorem 3.1), these two variants agree. An inspection of the proof shows that only connected FGTGDs without constants and a connected CQ without constants are required. Thus, the theorem follows immediately from Proposition 5.23.   □

We now turn toward proving Theorem 5.21, namely, the PSpace lower bound for ∃NQI under linear TGDs. Recall that the reduction in Proposition 5.23 does not preserve smaller classes of sentences, such as linear TGDs. We thus prove the theorem using a separate reduction.

Proof of Theorem 5.21. We reduce from the implication problem for IDs, which is known to be PSpace-hard from Casanova et al. [24]. Consider a set of IDs Σ and an additional ID $\delta = S_\star(\bar{x}_\star) \rightarrow \exists \bar{y}\ T_\star(\bar{z}_\star)$, where $\bar{x}_\star, \bar{y}$ are sequences of pairwise distinct variables and $\bar{z}_\star$ is a sequence of variables from $\bar{x}_\star$ unioned with $\bar{y}$. We denote by $F(\delta)$ the sequence of variables shared between $\bar{x}_\star$ and $\bar{z}_\star$—that is, the exported variables of this dependency. We let m denote the length of this vector. Note that we annotated relations and variables in δ with the subscript $\star$ to make it clear when we refer later to these particular objects.

We create a new schema S' that contains, for each relation R of arity k in the original schema S, a relation R' of arity $k + m$. We also add to S' a copy of each relation R from S, without changing the arity. Furthermore, we add a 0-ary relation Good, which is the only visible relation of S'. Consider an ID in Σ of the form

$$R(\bar{x}) \rightarrow \exists \bar{y}\ S(\bar{z}),$$

where $\bar{z}$ enumerates variables from $\bar{x}$ and $\bar{y}$. We introduce a corresponding "expanded ID" in Σ' of the form

$$R'(\bar{x}, \bar{x}') \rightarrow \exists \bar{y}\ S'(\bar{z}, \bar{x}'),$$

where the variables in $\bar{x}'$ are m variables distinct from the variables in $\bar{x}$, added in the last m places of R' and S'. Thus, we are carrying around the values of $\bar{x}'$ as "parameters." We also add the sentences

$$S_\star(\bar{x}_\star) \rightarrow S'_\star(\bar{x}_\star, F(\delta))$$

$$T'_\star(\bar{z}_\star, F(\delta)) \rightarrow \text{Good},$$

where the elements of $\bar{z}_\star$ are arranged as in the atom $T_\star(\bar{z}_\star)$ that appears on the right-hand side of the ID δ. Note that the sentence that copies the content from $S_\star$ to $S'_\star$ and duplicates the exported positions is not an ID but is still a linear TGD. The query of our ∃NQI problem is defined as

$$Q' = \exists \bar{x}_\star\ S_\star(\bar{x}).$$

Intuitively, the first added dependency above initializes the visible chase process by copying nulls in $S_\star$ atom into the additional parameter positions of $S'_\star$. The second additional dependency will verify that these values propagate to $T'_\star$ via the usual chase procedure.

The sentences that we just defined are preserved under disjoint unions. Thus, by Theorem 5.15, we know that ∃NQI(Q', Σ', S') = true if and only if NQI$(Q', \Sigma', S', \emptyset)$ = true. In the following, we prove that the latter holds if and only if the ID δ is implied by the set of IDs in Σ.

In one direction, suppose that the implication holds. From this, we can easily infer that in the schema S', the following dependency holds:

$$S'_\star(\bar{x}_\star, F(\delta)) \rightarrow \exists \bar{y}\ T'_\star(\bar{z}_\star, F(\delta)).$$

Consider now a full S'-instance $\mathcal{F}'$ with an empty visible part. We show that the query Q' is not satisfied, namely, $\mathcal{F}'$ cannot satisfy $\exists \bar{x}_\star\ S_\star(\bar{x}_\star)$. If $\mathcal{F}'$ did satisfy $S_\star(\bar{c}, \bar{x}_\star)$, then, by the copy of the sentences on the primed relations, this would yield $\exists \bar{x}_\star\ S'_\star(\bar{c}, F(\delta))$. Hence, by the sentences in the background theory, we infer that $\exists \bar{z}_\star\ T'_\star(\bar{z}_\star, F(\delta))$ holds and thus that Good holds. However, this would contradict the hypothesis that $\mathcal{F}'$ has an empty visible part.

In the other direction, suppose that the implication fails and consider a witness S-instance $\mathcal{F}$ that contains the fact $S_\star(\bar{x}_\star)$ but not the corresponding $T_\star$ fact. We create a full S'-instance $\mathcal{F}'$ with an empty visible part where Q' holds, thus showing that ∃NQI$(Q', \Sigma', S', \emptyset)$ = false. We first copy in $\mathcal{F}'$ the content of all relations R from $\mathcal{F}$. In particular, $\mathcal{F}'$ contains the fact $S_\star(\bar{x}_\star)$, but no $T_\star$

fact. The primed relations R' in $\mathcal{F}'$ are set to contain all and only the facts of the form $R'(\bar{x}, F(\delta))$, where $R(\bar{x})$ is a fact in $\mathcal{F}$. Finally, we set Good to be the empty relation in $\mathcal{F}'$. Clearly, Q' holds in $\mathcal{F}'$ and the visible part is the empty instance. It is also easy to verify that all sentences in Σ' are satisfied by $\mathcal{F}'$, and this completes the proof. □

Note that the reduction above does not create a schema with IDs but rather with general linear TGDs (variables can be repeated on the right).

At this point, we look at the connectedness requirement, which was used in our upper bounds. We show that it is critical for decidability.

THEOREM 5.24. *The problem* $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S})$ *is undecidable as Q ranges over Boolean CQs without constants and Σ over sets of FGTGDs without constants.*

PROOF. We give a reduction from the *model conservativity problem* for $\mathcal{EL}$ TBoxes, which is shown undecidable in the work of Lutz and Wolter [42]. Intuitively, $\mathcal{EL}$ is a logic that defines FGTGDs over relations of arity 2, called *TBoxes*. Given some TBoxes ϕ_1 and ϕ_2 over two schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, with $\mathbf{S}_1 \subseteq \mathbf{S}_2$, we say that ϕ_2 is a *model conservative extension* of ϕ_1 if every $\mathbf{S}_1$-instance $\mathcal{V}$ that satisfies ϕ_1 can be extended to an $\mathbf{S}_2$-instance that satisfies ϕ_2 without changing the interpretation of the predicates in $\mathbf{S}_1$—that is, by only adding an interpretation for the relations that are in $\mathbf{S}_2$ but not in $\mathbf{S}_1$. The model conservativity problem consists of deciding whether ϕ_2 is a model conservative extension of ϕ_1. The proof in the work of Lutz and Wolter [42] shows that this problem is undecidable for both finite instances and arbitrary instances.

We reduce the above problem to the complement of $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S})$, for suitable Q, Σ, and $\mathbf{S}$, as follows. Given some TBoxes ϕ_1 and ϕ_2 over the schemas $\mathbf{S}_1 \subseteq \mathbf{S}_2$, let $\mathbf{S}$ be the schema obtained from $\mathbf{S}_2$ by adding a new predicate Good of arity 0 and by letting the visible part be $\mathbf{S}_1$ (in particular, the relation Good is hidden). Further, let $\Sigma = \{\phi_1, \mathrm{Good} \to \phi_2\}$, where $\mathrm{Good} \to \phi_2$ is shorthand for the collection of FGTGDs obtained by adding Good as a conjunct to the left-hand side of each dependency of ϕ_2 (note that this makes the dependency unconnected). Finally, consider the query $Q = \mathrm{Good}$. We have that $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S}) = \mathrm{true}$ if and only if there is an $\mathbf{S}_1$-instance $\mathcal{V}$ satisfying ϕ_1, none of whose $\mathbf{S}_2$-expansions satisfies ϕ_2. □

In particular, this shows that our critical-instance method depends on connectedness. If constants are allowed, $\exists \mathrm{NQI}$ becomes undecidable even for connected FGTGDs, which indicates that our critical-instance method fails in the presence of constants.

THEOREM 5.25. *The problem* $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S})$ *is undecidable when*

(1) *Q ranges over Boolean CQs without constants and Σ ranges over sets of connected FGTGDs with constants or*
(2) *Q ranges over Boolean CQs with constants and Σ ranges over sets of connected FGTGDs without constants.*

PROOF. Both claims are proved by reduction from Theorem 5.24. Let Σ be any set of FGTGDs and Q a Boolean CQ, over a schema $\mathbf{S}$.

For the first item, let $\mathbf{S}'$ be the schema obtained from $\mathbf{S}$ by adding, for each k-ary (visible or hidden) relation R, an additional hidden $k + 1$-ary relation R'. Fix a fresh constant a. Let Σ' be the following set of *connected* FGTGDs over $\mathbf{S}'$: for every FGTGD in Σ, Σ' contains the connected FGTGD obtained from it by replacing every atom $R(\bar{x})$ by $R(\bar{x}, y)$, where y is a fixed variable shared across all atoms in the FGTGD. In addition, Σ' contains the connected FGTGDs $R(\bar{x}) \to R'(\bar{x}, a)$ and $R'(\bar{x}, a) \to R(\bar{x})$ for each relation R.

It is easy to see that $\exists \mathrm{NQI}(Q, \Sigma, \mathbf{S})$ and $\exists \mathrm{NQI}(Q, \Sigma', \mathbf{S}')$ coincide: every full instance $\mathcal{F}$ witnessing $\mathrm{NQI}(Q, \Sigma, \mathbf{S}) = \mathrm{true}$ gives rise to a full instance $\mathcal{F}'$ witnessing $\mathrm{NQI}(Q, \Sigma', \mathbf{S}') = \mathrm{true}$, where $\mathcal{F}'$

extends $\mathcal{F}$ with all facts $R'(\bar{b}, a)$ for $R(\bar{b})$ a fact of $\mathcal{F}$. Conversely, every full instance witnessing NQI(Q, Σ', S') = true gives rise (by restricting to the relations in the original schema S) to a full instance witnessing ∃NQI(Q, Σ, S).

For the second item, we provide a similar reduction: let S' be the schema obtained from S by increasing the arity of every (visible or hidden) relation by 1. Fix a constant a, and let Q' be the CQ obtained from Q by replacing every atom $R(\bar{x})$ by $R(\bar{x}, a)$. Furthermore, let Σ' be the set of *connected* FGTGDs obtained from Σ by replacing every atom $R(\bar{x})$ by $R(\bar{x}, y)$, where y is a fixed variable shared across all atoms in the FGTGD.

Again, it is easy to see that ∃NQI(Q, Σ, S) and ∃NQI(Q, Σ', S') coincide: every full instance $\mathcal{F}$ witnessing NQI(Q, Σ, S) = true gives rise to a full instance $\mathcal{F}'$ witnessing NQI(Q, Σ', S') = true, where $\mathcal{F}'$ consists of all facts $R(\bar{b}, a)$ for $R(\bar{b})$ a fact of $\mathcal{F}$. Conversely, every full instance witnessing NQI(Q, Σ', S') = true gives rise to a full instance $\mathcal{F}'$ witnessing ∃NQI(Q, Σ, S), where $\mathcal{F}'$ consists of all facts $R(\bar{b})$ for $R(\bar{b}, a)$ a fact of $\mathcal{F}$ having a as its last argument. □

6 EXTENSIONS AND SPECIAL CASES

Next we present some results concerning natural extensions of the framework.

Non-Boolean queries. Throughout this work, we have restricted to Boolean queries. The natural extension of the notion of query implication for non-Boolean queries is to consider inference of information concerning membership of any visible tuple in the query output. For example, PQI($Q, \Sigma, S, \mathcal{V}$) would hold if there is a tuple $\bar{t}$ over the active domain of $\mathcal{V}$ such that $\bar{t} \in Q(\mathcal{F})$ for all instances $\mathcal{F}$ of S satisfying the background theory Σ and having visible part $\mathcal{V}$. As usual, the schema-level problem ∃PQI(Q, Σ, S) (respectively, ∃NQI(Q, Σ, S)) for a non-Boolean query Q amounts to deciding whether there is a realizable visible instance $\mathcal{V}$ witnessing PQI($Q, \Sigma, S, \mathcal{V}$) (respectively, NQI($Q, \Sigma, S, \mathcal{V}$)).

The complexity upper bounds for the instance-level problems carry over to non-Boolean queries in a rather simple way. For example, given $S, \Sigma, \mathcal{V}$ as usual, and given a non-Boolean query Q and a visible tuple $\bar{t}$, the problem of deciding whether $\bar{t}$ appears in every potential output $Q(\mathcal{F})$, for any instance $\mathcal{F}$ satisfying Σ and having visible part $\mathcal{V}$, reduces to the problem PQI($Q_{\bar{t}}, \Sigma, S, \mathcal{V}$), where $Q_{\bar{t}}$ is the Boolean query obtained by substituting the i-th free variable of Q with the i-th constant in $\bar{t}$, for all i's. A similar reduction holds for negative implication. Thus, the instance-level problem in the non-Boolean case reduces to a series of instance-level problems in the Boolean case, one for each choice of a tuple $\bar{t}$ over the active domain of $\mathcal{V}$. Our upper bounds can be applied to the latter problems, since they hold in the presence of constants in the query. Moreover, the iteration over the tuples $\bar{t}$ can be absorbed in the complexity classes of our upper bounds: for data complexity, the iteration is polynomial, whereas for combined complexity, the number of tuples can be exponential, but our bounds are at least exponential. Further, GFP-Datalog definability for negative implications also extends straightforwardly to the non-Boolean case: Theorem 5.8 extends with the same statement and proof, whereas the argument in Theorem 5.10 is easily extended to show that there is a GFP-Datalog program that returns the complement of NQI(Q, Σ, S) within the active domain.

Turning to schema-level results for ∃PQI as listed in Figure 1, we can generalize our Theorem 4.11, which reduces ∃PQI to PQI on a critical instance, to non-Boolean queries, by revising it to state ∃PQI(Q, Σ, S) = true if and only if there is a PQI for the tuple $(a, \ldots, a)$ and the instance $\mathcal{V}_{\{a\}}$. Consequently, Theorem 4.20 and Corollary 4.13 extend to non-Boolean queries without constants. We believe that Theorem 4.21 (which we derived from earlier work [10]) can be lifted to non-Boolean queries without constants as well, but we have not verified this.

The more problematic case for non-Boolean queries is that of ∃NQI. We do not know if the upper bounds for ∃NQI carry over from the Boolean case. We leave this for future work.

Beyond unions of CQs. So far, we have considered only the case where the query Q does not contain negation or universal quantification. It is natural to extend the query language even further, to **Boolean combinations of Boolean conjunctive queries (BCCQs)**. We note that the problem PQI($Q, \Sigma, S, \mathcal{V}$), as Q ranges over BCCQs, subsumes both PQI($Q, \Sigma, S, \mathcal{V}$) and NQI($Q, \Sigma, S, \mathcal{V}$) for Q a UCQ. Thus, all lower bounds for either of these two problems are inherited by the BCCQ problem. The corresponding instance-level problems are still decidable. Indeed, this holds even when Q is a GNFO sentence, since we can use the same translation to GNFO satisfiability applied in Theorems 4.1 and 5.1. However, for the schema-level problems ∃PQI and ∃NQI, we immediately run into problems.

THEOREM 6.1. *The problem* ∃PQI(Q, Σ, S) *for a Boolean combination Q of Boolean CQs is undecidable, even when the sentences in the background theory are IDs. The same holds for* ∃NQI(Q, Σ, S).

PROOF. As in the previous undecidability results, we reduce a tiling problem with tiles T, initial tile $t_\perp \in T$, and horizontal and vertical constraints $H, V \subseteq T \times T$ to the problem ∃PQI(Q, Σ, S). Again, for convenience, we deal with the infinite variant of the problem. The idea will be that the visible instance witnessing ∃PQI represents the tiling, and invisible instances represent challenges to the correctness of the tiling.

We model the infinite grid to be tiled by visible relations E_H and E_V, and the tiling function by a collection of unary visible relations U_t, for all tiles $t \in T$.

The invisible relations represent markings of the grid for possible errors. There are several kinds of challenges. We focus on the horizontal consistency challenge, which selects two nodes in the E_H relation, to challenge whether the nodes satisfy the horizontal constraint. Formally, the challenge is captured by a binary invisible predicate HorChallenge(x, y), with an associated sentence in the background theory

$$\text{HorChallenge}(x, y) \ \rightarrow \ E_H(x, y).$$

The query Q will be satisfied only when the following *negated* CQs hold, for all pairs $(t, t') \notin H$:

$$\neg \exists \, x \, y \ \text{HorChallenge}(x, y) \ \wedge \ U_t(x) \ \wedge \ U_{t'}(y).$$

Note that this can only happen if the relation HorChallenge has selected two horizontally adjacent nodes whose tiles violate the horizontal constraints. The vertical constraints are enforced in a similar way using an invisible relation VertChallenge and another negated CQ.

Recall that in the infinite grid, we have unique vertical and horizontal successors of each node, and the horizontal and vertical successor functions commute. Thus far, we have not enforced that E_V and E_H have this property. We will use additional hidden relations and IDs to enforce that every element is related to at least one other via E_H and E_V.

We first show how to enforce that every element has at most one horizontal successor ("functionality challenge"). We introduce a hidden relation HorFuncChallenge(x, y, y') and a background theory sentence

$$\text{HorFuncChallenge}(x, y, y') \ \rightarrow \ E_H(x, y)$$
$$\text{HorFuncChallenge}(x, y, y') \ \rightarrow \ E_H(x, y').$$

We also add to the query Q the conjunct:

$$\big(\neg \exists \, x \, y \, y' \ \text{HorFuncChallenge}(x, y, y') \big) \ \vee \ \big(\exists \, x \, y \ \text{HorFuncChallenge}(x, y, y) \big).$$

We claim that if there is a visible instance witnessing ∃PQI, then E_H is functional. Indeed, if E_H were not functional in the visible instance, then we could choose a node x with two distinct

E_H-successors y and y', add only the tuple (x, y, y') to HorFuncChallenge, and obtain a full instance that satisfies the sentences of the background theory but not the query Q. Conversely, suppose that E_H is functional in a visible instance $\mathcal{V}$, and consider any full instance $\mathcal{F}$ that satisfies the background theory and agrees with $\mathcal{V}$ on the visible part. If there are no tuples in HorFuncChallenge, the conjunct above is clearly satisfied by its first disjunct. If there is some tuple (x, y, y') in HorFuncChallenge, then by the background theory, we must have $E_H(x, y)$ and $E_H(x, y')$, and hence, by functionality, $y = y'$. In this case, the conjunct above holds via the second disjunct. The functionality of the vertical relation E_V is enforced in an analogous way.

Commutativity of E_H and E_V can be also enforced using a similar technique. We add a hidden relation ConfChallenge(x, y, z, u, v) with the following sentences in the background theory:

$$\begin{aligned}
\text{ConfChallenge}(x, y, z, u, v) &\rightarrow E_H(x, y) \\
\text{ConfChallenge}(x, y, z, u, v) &\rightarrow E_V(y, u) \\
\text{ConfChallenge}(x, y, z, u, v) &\rightarrow E_V(x, z) \\
\text{ConfChallenge}(x, y, z, u, v) &\rightarrow E_H(z, v).
\end{aligned}$$

A potential tuple in ConfChallenge(x, y, z, u, v) represents the join of a triple of nodes moving first horizontally and then vertically from x (i.e., x, y, u) and a triple going first vertically and then horizontally from x (i.e., x, z, v). For the relations to commute, we must satisfy the query

$$\left(\neg \exists\, x\, y\, z\, u\, v\ \text{ConfChallenge}(x, y, z, u, v) \right) \;\vee\; \left(\exists\, x\, y\, z\, u\ \text{ConfChallenge}(x, y, z, u, u) \right)$$

in the full instance. Thus, we add the above conjunct to Q.

Putting the various components of Q for different challenges together as a Boolean combination of CQ completes the proof of the theorem. $\qquad\square$

The case of CQ views. As mentioned earlier, the database community has studied the PQI problem in the case where the background theory consist exactly of CQ-view definitions that determine each visible relation in terms of invisible relations. Formally, a CQ-view-based scenario consists of a schema $\mathbf{S} = \mathbf{S}_v \cup \mathbf{S}_h$, namely, the union of a schema for the visible relations and a schema for the hidden relations, and a set of sentences Σ between visible and hidden relations that must be of a particular form. For each visible relation $R \in \mathbf{S}_v$, Σ must contain two dependencies of the form

$$\begin{aligned}
R(\bar{x}) &\rightarrow \exists \bar{y}\ \phi_R(\bar{x}, \bar{y}) \\
\phi_R(\bar{x}, \bar{y}) &\rightarrow R(\bar{x}),
\end{aligned}$$

where ϕ_R is a conjunction of atoms over the hidden schema $\mathbf{S}_h$, Furthermore, all sentences in Σ must be of the above forms. Note that this CQ-view scenario is incomparable in expressiveness to GNFO sentences.

The PQI problem is decidable, because given a visible instance $\mathcal{V}$, the sentences can be rewritten as $\Sigma_1 \wedge \Sigma_2$, where Σ_1 consists of TGDs from the view relations to the base relations, and Σ_2 consists of sentences of the form $V(\bar{x}) \rightarrow \bigvee_{\bar{a} \in V(\mathcal{V})} \bar{x} = \bar{a}$. Let us consider what will happen in the the visible chase of $\mathcal{V}$ with these dependencies. In the first round, we will fire the Σ_1 dependencies, which will create facts over the invisible source relations for each view atom. In the subsequent rounds, we will fire Σ_2 dependencies that non-deterministically merge source elements with elements in the view relations. We continue with these firing/merging steps with Σ_2 constraints until no new rules fire. The process must terminate because in these merging rounds, no new elements will be created.

The decidability of the $\exists$PQI problem follows immediately from these observations and Theorem 4.11, which applies to background theories capturing CQ-view definitions. We can also see

that the problem ∃PQI is finitely controllable. The construction of Theorem 4.11 creates a counterexample to PQI on the critical instance from a counterexample to PQI on an arbitrary visible instance that is one of the members of the visible chase. If we have an arbitrary finite visible instance, then the visible chase will contain only finite instances, by the termination algorithm above. Hence, the argument of Theorem 4.11 will produce a finite counterexample to PQI on the critical instance.

Note that subsequent to our work, an earlier work [16] showed that ∃PQI becomes undecidable for UCQ view definitions. This result is related to (but formally orthogonal to) our undecidability results for disjunctive TGDs earlier in the article.

The NQI problem for CQ views is also decidable, because we can bound the size of a counterexample witness full instance. Given a full instance $\mathcal{F}$ that gives a given view image $\mathcal{V}$ where CQ Q is non-empty, only polynomial many facts in $\mathcal{F}$ over the invisible schema are needed to ensure that the view image is $\mathcal{V}$, and only polynomial many facts over $\mathcal{F}$ are needed to ensure that Q is non-empty. Given $\mathcal{V}$ and Q, an algorithm can thus non-deterministically guess all such sets of facts and check that one of them returns non-empty. When we turn to ∃NQI, we find that even CQ views lead to undecidability.

THEOREM 6.2. *The* ∃NQI *problem under background knowledge given as CQ-view definitions is undecidable.*

The proof, deferred to the electronic appendix, uses a variant of the encoding technique used in the previous undecidability results in this article.

7 CONCLUSION

This work gives a detailed examination of inference of information from complete knowledge about a subset of the signature coupled with background knowledge about the full signature. Both the information and the background knowledge are expressed by logical sentences. In future work, we will look at mechanisms for "restricted access" that are finer grained than just exposing the full contents of a subset of the schema relations. One such mechanism consists of language-based restrictions—the ability to evaluate open formulas over the schemas in a fragment of the logic. Another mechanism consists of functional interfaces—for example, the "access method" interfaces studied in other works [18, 19].

APPENDICES

A PROOF OF EXPONENTIAL TIME SATISFIABILITY FOR GNFO WITH FIXED WIDTH, FIXED CQ-RANK, AND FIXED ARITY OF SCHEMA

In this appendix, we give details of the following result:

Satisfiability of GNFO sentences is decidable in exponential time if the following parameters are fixed: the *width* and the *CQ-rank*.

A doubly exponential bound on satisfiability of GNFO was proven in the works where GNFO was introduced [6, 7]. However, the argument was by reduction to satisfiability of the *guarded fragment*, which was already known to be decidable in doubly exponential time.

The guarded fragment is the fragment of first-order logic built up from relational atoms and equality atoms via the Boolean operators ∧, ∨, ¬ along with the guarded quantifiers: $\exists \bar{x}\, R(\bar{x}) \wedge \varphi$ and $\forall \bar{x}\, R(\bar{x}) \rightarrow \varphi$, where the free variables of φ are contained in $\bar{x}$. It is easily seen that every sentence of the guarded fragment is expressible in GNFO, and the results of Bárány et al. [6, 7] show that the fragments do not differ in the complexity of satisfiability.

Conversions of GNFO formulas to automata, and comments about what controls their complexity, are implicit in a number of other works [13, 14, 20]. But the conversions are performed for richer logics than GNFO. This means first that they introduce many complications that are unnecessary for GNFO, and second that they do not provide the precise statements for GNFO that we require in our analysis of inference problems.

Here, we give a direct reduction of satisfiability of GNFO to emptiness testing for a tree automaton. The translation allows us to track the complexity of satisfiability in a more fine-grained way, including the collapse to exponential time when the width and the CQ-rank is fixed.

We will start in Section A.1 explaining the tree-like model property, and in Section A.2 giving background on the automaton formalism we use. In Section A.3, we show decidability of GNFO by restricting to sentences of a special kind, namely, sentences in normal form and without equality or constants. We present a result that provides an exponential time bound for formulas in normal form once the width, CQ-rank, and additionally the maximum arity of relations is fixed. In Section A.4, we extend the result to GNFO sentences in normal form with equalities and constants. Finally, in Section A.5, we lift the assumption that the arity is also fixed. We close in Section A.6 with some remarks relating the results here with the bounds in the absence of any normal form restriction.

A.1 Tree-Like Models and Automata

The first step in showing decidability of GNFO satisfiability is to show that for any sentence φ there is a number k, easily computed from φ, such that if φ is satisfiable, it is satisfiable over structures that are "k-tree-like." Here, structure is k-tree-like if it is coded by a tree, where each vertex in the tree represents at most k elements in the structure.

In this section, we will explain the tree-like model property. In doing so, we will restrict to GNFO sentences that do not have equality or constants. The extension to equality and constants will be given in Section A.4.

We start by describing what these tree codes look like in detail. These are essentially unordered, unranked trees with nodes labeled by unary predicates from a fixed finite signature.

More precisely, for a number k, we let $N_k = \{1, \ldots, 2k\}$. This is a finite set of *names* that will be used to describe the elements represented in a given node of the tree.

Given a relational signature σ and a number k, the k-code signature, $\Sigma_{\sigma,k}^{\text{code}}$ contains

- A unary predicate D_a for all $a \in N_k$
- Unary predicates $R_{\bar{a}}$ for all $R \in \sigma$ of arity j and all $\bar{a} \in (N_k)^j$.

Informally, $D_a(v)$ indicates that a is a name in the node v in the tree code, whereas $R_{\bar{a}}(v)$ indicates that R holds for the elements represented by the names $\bar{a}$ at v.

Neighboring nodes may describe overlapping pieces of the structure. This will be implicitly coded based on repeated use of names: if some name appears in two neighboring nodes, then the same element is being described in both nodes. This is why N_k has $2k$ names, even though at most k names are used in a single node.

For a vertex v in a $\Sigma_{\sigma,k}^{\text{code}}$ tree $\mathcal{T}$, let $\text{names}(v) := \{a \in N_k : D_a \text{ holds of } v\}$. This denotes the set of *names* used for elements in node v.

A *consistent $\Sigma_{\sigma,k}^{code}$-tree* is a $\Sigma_{\sigma,k}^{\text{code}}$-tree such that every node v satisfies

- $|\text{names}(v)| \leq k$
- For all $R_{\bar{a}} \in \Sigma_{\sigma,k}^{\text{code}}$, if $R_{\bar{a}}(v)$, then $\bar{a} \subseteq \text{names}(v)$.

When σ is clear from context, such a tree will also be called a k-code.

We now describe the structure coded by a k-code formally. Given a consistent tree $\mathcal{T}$ and a local name a, we say that nodes u and v are a-connected if there is a sequence of nodes $u = w_0, w_1, \ldots, w_j = v$ such that w_{i+1} is a parent or child of w_i, and $a \in \text{names}(w_i)$ for all $i \in \{0, \ldots, j\}$. Note that the property of being a-connected is an equivalence relation. We write $[v]_a$ for the equivalence class of a-connected nodes of v. Moreover, for $\bar{a} = a_1 \ldots a_n$, we often abuse notation and write $[v]_{\bar{a}}$ for the tuple $[v]_{a_1}, \ldots, [v]_{a_n}$.

Definition A.1. The *decoding* of $\mathcal{T}$ is the σ-structure $\text{decode}(\mathcal{T})$ with universe

$$\{[v]_a : v \in \text{dom}(\mathcal{T}) \text{ and } a \in \text{names}(v)\},$$

and with each relation R instantiated by the set

$$R^{\text{decode}(\mathcal{T})} = \{[w]_{\bar{a}} : w \in R_{\bar{a}}\}.$$

So, intuitively, the elements of the decoding of $\mathcal{T}$ are represented by equivalence classes of a-connected nodes of $\mathcal{T}$, for every name a, and the facts of any relation R of the signature are witnessed by tuples of equivalence classes of $R_{\bar{a}}$-labeled nodes of $\mathcal{T}$, for every a tuple $\bar{a}$ of names of the same arity as R.

We are now ready to state the result that satisfiable GNFO sentences have k-tree-like models, for some appropriate k. The original works on GNFO [6, 7] show that every satisfiable GNFO sentence (even with equality but without constants) has a satisfying model with a *tree decomposition* where every vertex of the tree is associated with k elements of the model and k is the width of a suitable normal form of the formula. This result can be transferred to our setting. We will not need to introduce the definition of tree decomposition here, but it is easy to see (and explained in other works (e.g., [13]) that structures with such a decomposition have codes of the form given above. We also recall (cf. Proposition 4.2) that any GNFO formula can be put in a normal form that satisfies the following grammar:

$$\varphi ::= \bigvee_i \exists \bar{x}_i \ \bigwedge_j \psi_{ij}$$
$$\psi ::= \alpha \ | \ \alpha \wedge \neg \varphi,$$

where α is an atomic formula and free variables of φ are contained in free variables of α. Finally, we recall that the *width* of a formula φ in normal form is the maximum number of free variables of any of its subformulas. Hence, we have the following proposition.

PROPOSITION A.2 ([7]). *Suppose that φ is a GNFO sentence in normal form without constants and having width k. If φ is satisfiable, then it is satisfiable in a structure that is the decoding of some k-code.*

Tree codes like this can generally have unbounded (possibly infinite) degree. It is well known that if a first-order sentence φ is satisfiable, there is a structure M that is countable such that $M \models \varphi$—this follows from the Lowenheim-Skolem theorem [25]. Using this fact, one can refine the proof of Proposition A.2 to show that M is satisfiable in a countable model that has a k-tree code where the branching degree is countable.

For technical reasons, it is more convenient to use full binary trees for our encodings. Any tree code $\mathcal{T}$ where each node has at most countably many children can be converted to a binary tree that encodes the same model, in the following way. First, for each node u, we add infinitely many new children to u, each child being the root of an infinite full binary tree where each node has the same label as u in $\mathcal{T}$. This ensures that each node of $\mathcal{T}$ now has infinitely many (but still countably many) children. Second, we convert $\mathcal{T}$ into a full binary tree, using the classical "first-child, next-sibling" encoding. More precisely, we transform the tree inductively, starting from the root and

replacing every node u with children $v_1, v_2, \ldots$ by the tree that has an infinite rightward spine consisting of copies of u, and where the i-th copy of u along the spine has node v_i as left child.

A.2 Automata Background

We will consider automata that process infinite complete binary trees—that is, infinite trees in which the outdegree of every vertex is 2. We assume a set of unary predicates $A_1 \ldots A_n$ for such input trees, and let Σ be $\{A_1 \ldots A_n\}$.

We will look at automata that can move up and down in such trees. Let Direction_2 be the set of (movement) *directions*: Stay, Down_1, Down_2, and Up.

For any set J, let $B^+(J)$ be the set of positive Boolean combinations of propositions in J. Given a set $I \subseteq J$ and a formula $\varphi \in B^+(J)$, the notion of φ holding in I ($I \models \varphi$) is defined as usual in propositional logic: a single element $j \in J$ holds in I if $j \in I$, a disjunction holds in I if one of its disjuncts holds, and a conjunction holds if all of its conjuncts hold. We will be interested in positive Boolean combinations over $\text{Direction}_2 \times Q$; these formulas will be used to describe possible moves of the automaton.

We will convert GNFO sentences in normal form to *two-way alternating Büchi tree automata* (2ABTA for short) that process infinite binary trees labeled over $\mathcal{P}(\Sigma)$. An automaton of this type is specified as a tuple $(Q, \Sigma, q_0, \delta, \Omega)$, where

- Q is a finite set of states,
- Σ is as above,
- $q_0 \in Q$ is the *initial state*,
- $\delta \in Q \times \mathcal{P}(\Sigma) \to B^+(\text{Direction}_2 \times Q)$ is the *transition function*,
- Ω is an acceptance condition, which we discuss in the following.

A *run* of the automaton on a tree $\mathcal{T}$ is another tree $\mathcal{R}$ (not necessarily binary) with a labeling function $\lambda_{\mathcal{R}}$ that maps each vertex of $\mathcal{R}$ to a pair consisting of a vertex of $\mathcal{T}$ and a state $q \in Q$. We now describe further properties that are required for the run $\mathcal{R}$ to be *accepting*.

First, we require that the root of $\mathcal{R}$ is labeled by a pair (v, q_0) consisting of a vertex v_0 of $\mathcal{T}$ and the initial state of the automaton. We say that the run $\mathcal{R}$ *starts at vertex* v_0 (which does not need to be the root of $\mathcal{T}$ in general).

Second, we require that the relationship between parent and children labels in $\mathcal{R}$ be consistent with the transition function δ of the automaton. Consider any vertex w of $\mathcal{R}$ together with its label and the labels of its children, as specified by $\lambda_{\mathcal{R}}$, say $\lambda_{\mathcal{R}}(w) = (v_w, q_w)$ and $\lambda_{\mathcal{R}}(w') = (v_{w'}, q_{w'})$ for each child w' of w. For every direction $d \in \text{Direction}_2$, let $d(v)$ be the vertex reached from v by a move along d, namely, let $d(v)$ be either v, the left child of v, the right child of v, or the parent of v, depending on whether $d = \text{Stay}$, $d = \text{Down}_1$, $d = \text{Down}_2$, or $d = \text{Parent}$. Consistency with the transition function δ is enforced by requiring that

- For every child w' of w, there is a direction $d_{w'} \in \text{Direction}_2$ such that $v_{w'} = d_{w'}(v_w)$,
- $I_w \models \delta(q, S_v)$, where I_w is the set of all pairs $(d_{w'}, q_{w'})$, with w' child of w, and S_v is the set of predicates labeling v in $\mathcal{T}$.

Finally, we require that every branch of $\mathcal{R}$ obeys the acceptance condition Ω. There are a number of different acceptance conditions defined for automata over infinite trees. Here, we will make use of the *Büchi acceptance condition*. This is specified by a set $\Omega \subseteq Q$ of accepting states. The requirement is that along each branch in $\mathcal{R}$, there is a state in Ω that occurs infinitely often along the branch.

Given an automaton $\mathcal{A}$, the *language recognized by $\mathcal{A}$*, denoted $L(\mathcal{A})$, is the set of trees $\mathcal{T}$ that admit an accepting run of $\mathcal{A}$ starting at the root of $\mathcal{T}$. The *non-emptiness problem* for a class of

automata is the analog of the satisfiability problem for a logic: given an automaton $\mathcal{A}$ in the class, determine if $L(\mathcal{A}) \neq \emptyset$.

Vardi [50] showed that non-emptiness is decidable in ExpTime for 2ABTA (in fact, this was shown for parity automata, which includes Büchi automata).

THEOREM A.3 ([50]). *It is decidable in* ExpTime *whether* $L(\mathcal{A}) \neq \emptyset$ *for any given* 2ABTA $\mathcal{A}$. *More specifically, the running time of the decision procedure is* $p(|\mathcal{A}|)^{p(s)}$, *where s is the number of states of $\mathcal{A}$ and p is a polynomial independent of $\mathcal{A}$.*

In view of the above result, if we can reduce our satisfiability problem to an emptiness check for a 2ABTA with size doubly exponential in the size of the formula and number of states exponential in the size of the formula, we will obtain a doubly exponential bound on satisfiability. Similarly, if we can construct a 2ABTA with size exponential in the formula and number of states polynomial in the formula, we will obtain a singly exponential bound on satisfiability.

A.3 Decision Procedure for Normal-Form GNFO Without Equality and Constants

In giving the automata constructions in this section, we will assume that φ is a GNFO sentence in a normal form, similar to the one introduced by Bárány et al. [7]. Throughout this section, we also assume that the formulas do not use equality or constants. In particular, this means that every negation is either guarded by an atomic formula or involves a subformula with a single free variable.

We begin by focusing on the case where formulas do not have equality or constants. We recall that the formulas φ of GNFO in *normal form* are generated using the following grammar:

$$\varphi ::= \bigvee_i \exists \bar{x}_i \ \bigwedge_j \psi_{ij}$$
$$\psi ::= \alpha \ | \ \alpha \wedge \neg \varphi \ |$$
$$\varphi \text{ if } \varphi \text{ has at most one free variable } |$$
$$\neg \varphi \text{ if } \varphi \text{ has at most one free variable,}$$

where α is an atomic formula, and in the case of $\alpha \wedge \varphi$ and $\alpha \wedge \neg \varphi$, $\text{Free}(\alpha) \supseteq \text{Free}(\varphi)$. The φ are referred to as *UCQ-shaped formulas*, with each of the disjuncts being a *CQ-shaped formula*. Note that $\bigwedge_i \varphi_i$ is a sentence in normal form when the φ_i's are sentences in normal form.

A formula is *answer-guarded* if it has at most one free variable or is of the form $\alpha \wedge \chi$ where α is an atom that contains all free variables of χ. The idea of the normal form is that the grammar generates formulas that alternate been UCQ-shaped templates and guarded formulas. The normal form guarantees that each ψ, and thus in particular each ψ_{ij} component of a UCQ-shaped template, is answer guarded.

As mentioned in the body of the article, Bárány et al. [7] showed that every GNFO formula can be efficiently put in normal form. For completeness, we also include a procedure in Proposition 4.2. We also recall below the definitions of the parameters that are important for our complexity analysis.

The *width* of a GNFO formula φ in normal form, denoted width(φ), is the maximum number of free variables in any of its subformulas.

The *CQ-rank* of φ, denoted rank(φ), is the maximum number of conjuncts ψ_i in any CQ-shaped subformula $\exists \bar{x} \ \bigwedge_i \psi_i$ of φ, where $\bar{x}$ is non-empty. Note that the ψ_i in such a CQ-shaped formula are of the form α, $\alpha \wedge \varphi''$, or $\alpha \wedge \neg \varphi''$, but for the purposes of counting conjuncts for the CQ-rank, each ψ_i is treated as a single conjunct.

In our preliminary analysis, we will also fix the *maximum arity* of relations in the signature. This will fix the size of the signature underlying tree codes and thus the size of the automaton that

will check satisfiability of the given GNFO formula. We will see later (Section A.5) how to lift this restriction on the arity.

We now explain how to construct a two-way alternating Büchi tree automaton (2ABTA) for a GNFO sentence φ in normal form without equality or constants.

The rough idea will be that the automaton has states for all subformulas of φ—the "subformula closure" of φ. The automaton being at a vertex v of the input tree $\mathcal{T}$ with a state that corresponds to subformula ψ indicates that it is verifying that ψ holds at v in $\mathcal{T}$. The statement above is not precise because in GNFO, the notion of "subformula" needs to be more expansive than the usual one, in order to be able to correctly verify the CQ-shaped formulas.

Before we define the correct subformula closure, we need to think more carefully about CQ-shaped formulas and how they can be satisfied in a tree-like structure. For this, it is convenient to introduce *specializations*.

Specializations. Consider a CQ-shaped formula

$$\rho(\bar{x}) = \exists \bar{y} \bigwedge_{j \in \{1, \ldots, r\}} \psi_j(\bar{x}, \bar{y}).$$

A *specialization* of ρ is a formula ρ' obtained from ρ by the following operations:

- Select a subset $\bar{y}_0$ of $\bar{y}$; call variables from $\bar{x} \cup \bar{y}_0$ the *inside variables* and variables from $\bar{y} \setminus \bar{y}_0$ the *outside variables*;
- Select a partition $\bar{y}_1, \ldots, \bar{y}_s$ of the outside variables, with the property that for every ψ_j, either ψ_j has no outside variables or all of its outside variables are contained in some partition element $\bar{y}_j$;
- Let χ_0 be the conjunction of the ψ_j's whose free variables are among $\bar{x}$ and the inside variables $\bar{y}_0$, and let χ_i for $i \in \{1, \ldots, s\}$ be the conjunction of the ψ_j's that use some outside variables and satisfy $\mathsf{Free}(\psi_j) \subseteq \bar{x} \cup \bar{y}_0 \cup \bar{y}_i$;
- Set $\rho'(\bar{x}, \bar{y}_0)$ to be

$$\chi_0(\bar{x}, \bar{y}_0) \wedge \bigwedge_{i \in \{1, \ldots, s\}} \exists \bar{y}_i \, \chi_i(\bar{x}, \bar{y}_0, \bar{y}_i).$$

Roughly speaking, each specialization ρ' of ρ describes a possible way that a CQ-shaped formula could be satisfied by elements $\bar{x}$ represented in a node of a tree code, as described in Section A.1. The inside variables represent witnesses for the existential quantifiers that are found in the node itself. The partition of the outside variables represent the different directions from the node where the additional non-local witnesses are to be found: moving either to an ancestor or to one of the children. Since each atom of the CQ-shaped formula must be realized in a single node, each conjunct $\psi_j(\bar{x}, \bar{y})$ must be witnessed "homogeneously" with respect to the covering of the outside variables, as captured in the second item above.

It is easy to see that if a specialization is realized, then so is the original formula, since the realization of the specialization gives witnesses for the existential quantifiers.

LEMMA A.4. *Let $\rho(\bar{x}) = \exists \bar{y} \bigwedge_j \psi_j(\bar{x}, \bar{y})$ be a CQ-shaped formula. For all structures M and for all specializations $\rho'(\bar{x}, \bar{y}_0)$ of ρ, if $M \models \rho'(\bar{a}, \bar{b})$, then $M \models \rho(\bar{a})$.*

Since a formula is trivially a specialization of itself, where all variables $\bar{y}$ are chosen to be outside variables and the partition consists of a single set $\bar{y}_1 = \bar{y}$, the converse of Lemma A.4, with the specialization quantified existentially, is vacuously true. What is more useful, however, is that whenever a formula is realized in a tree code, some "non-trivial" specialization of it is also realized in the same tree code. Formally, we say that a specialization ρ' as above is *non-trivial* if either χ_0 is

non-empty or the partition $\bar{y}_1, \ldots, \bar{y}_s$ of the outside variables $\bar{y}$ is such that none of the $\bar{y}_i$ coincides with $\bar{y}$.

Let us briefly explain the use of specializations as a way to evaluate CQ-shaped formulas in a tree code. Consider the CQ-shaped formula $\rho = \exists y_1, y_2, y_3\, A(y_1, y_2) \wedge B(y_1, y_3) \wedge C(y_2, y_3)$, with A, B, C atomic binary predicates. Suppose that ρ holds in a model $\mathrm{decode}(\mathcal{T})$ that is the decoding of some tree code $\mathcal{T}$, with $\bar{y} = y_1 y_2 y_3$ interpreted by a tuple $\bar{a} = a_1 a_2 a_3$ of elements. The specialization $\rho'(y_1, y_2, y_3) = A(y_1, y_2) \wedge B(y_1, y_3) \wedge C(y_2, y_3)$, obtained from letting all quantified variables $\bar{y}$ be inside variables, also holds in the model $\mathrm{decode}(\mathcal{T})$, with the interpretation $\bar{a}$ for $\bar{y}$. However, this specialization may not be realized in a single node of the tree code $\mathcal{T}$, as it may happen that the atomic predicate A is witnessed at different nodes than those witnessing B and C. However, we could consider another non-trivial specialization, say $\rho''(y_1, y_2) = A(y_1, y_2) \wedge (\exists y_3\, B(y_1, y_3) \wedge C(y_2, y_3))$, with y_1, y_2 inside variables and y_3 outside variable, which could then be realized by two (a_1, a_2)-connected nodes witnessing, respectively, $A(a_1, a_2)$ and $B(a_1, a_3) \wedge C(a_2, a_3)$.

We formalize the above idea into a lemma. We recall from Proposition A.2 that every satisfiable GNFO formula in normal form (and, in particular, every satisfiable CQ-shaped formula) of width k has a model of the form $M = \mathrm{decode}(\mathcal{T})$, for some k-code $\mathcal{T}$. In particular, by the definition of decoding, the elements of M correspond to equivalence classes $[v]_a$ of a-connected notes $v \in \mathcal{T}$. Hereafter, we simply call $\mathcal{T}$ a *tree code of M*, thus avoiding to specify the width k of the formula. The following result captures the idea that in realizing a CQ-shaped formula, we need also to realize some simpler specialization of it.

LEMMA A.5. *Let $\rho(\bar{x}) = \exists \bar{y}\, \bigwedge_j \psi_j(\bar{x}, \bar{y})$ be a CQ-shaped formula. Given a structure M and its tree code $\mathcal{T}$, if there is a vertex $v \in \mathcal{T}$ that includes names $\bar{a}$ and such that $M \models \rho([v]_{\bar{a}})$, then there is a non-trivial specialization $\rho'(\bar{x}, \bar{y}_0)$ of ρ and a vertex $w \in \mathcal{T}$ that includes $\bar{a}$ and some additional names $\bar{b}_0$ and such that $[w]_{\bar{a}} = [v]_{\bar{a}}$ and $M \models \rho'([w]_{\bar{a}, \bar{b}_0})$.*

SKETCH. The idea behind the proof is that if the formula holds at a node with certain witnesses $\bar{a}$ for the free variables $\bar{x}$, we can traverse the nodes of the tree code, while preserving all of those witnesses $\bar{a}$, until we arrive at a node w where either some of the witnesses for the existentially quantified variables $\bar{y}$ are found locally in w or the witnesses for $\bar{y}$ are found in different directions from w. In the first case, we have realized a specialization in which χ_0 is non-empty; in the second case, we have realized a specialization in which the partition of the outside variables is non-trivial.   □

Let $\eta(\bar{x}) = \exists \bar{y}\, \bigwedge_j \psi_j(\bar{x}, \bar{y})$ be a CQ-shaped formula and let $\eta(\bar{a}) = \exists \bar{y}\, \bigwedge_j \psi_j(\bar{a}, \bar{y})$ be formed by substituting names $a_i \in N_k$ (recall that $N_k = \{1, \ldots, 2k\}$) for each free variable x_i of $\eta(\bar{x})$. We will write $\mathrm{Spec}(\eta(\bar{a}), N_k)$ for the set of all specializations of $\eta(\bar{a})$ with elements from N_k substituted for each new inside variable. For convenience, in the construction below, each formula in $\mathrm{Spec}(\eta(\bar{a}), N_k)$ will be represented by the set of its outermost conjuncts. In other words, any specialization of $\eta(\bar{a})$ is of the form $\chi_0(\bar{a}, \bar{b}_0) \wedge \bigwedge_{i \in \{1, \ldots, s\}} \exists \bar{y}_i\, \chi_i(\bar{a}, \bar{b}_0, \bar{y}_i)$, with $\chi_0(\bar{a}, \bar{b}_0)$ conjunction of all $\psi_j(\bar{a}, \bar{b})$'s that use only names $\bar{a}$ and names $\bar{b}_0$ corresponding to inside variables. This specialization will be represented as the set

$$S = \{\psi_j(\bar{a}, \bar{b}_0) : \psi_j(\bar{a}, \bar{b}_0) \text{ conjunct in } \chi_0(\bar{a}, \bar{b}_0)\} \cup \{\exists \bar{y}_i\, \chi_i(\bar{a}, \bar{b}_0, \bar{y}_i) : i \in \{1, \ldots, s\}\}.$$

We are now ready to define the notion of subformula that we are interested in. Fix some GNFO sentence φ in normal form. The closure $\mathrm{cl}(\varphi, N_k)$ that is relevant for the automaton construction to decide satisfiability of φ consists of the subformulas of φ along with formulas that are part of the specializations of the CQ-shaped formulas. Formally, $\mathrm{cl}(\varphi, N_k)$ is the smallest set of formulas containing φ, true, and false, and satisfying the following closure properties:

- If $\bigvee_i \psi_i \in \mathrm{cl}(\varphi, N_k)$, then $\psi_i \in \mathrm{cl}(\varphi, N_k)$ for all i;
- If $\alpha \wedge \psi \in \mathrm{cl}(\varphi, N_k)$, then $\alpha, \psi \in \mathrm{cl}(\varphi, N_k)$;
- If $\alpha \wedge \neg\psi \in \mathrm{cl}(\varphi, N_k)$, then $\alpha, \psi \in \mathrm{cl}(\varphi, N_k)$;
- If $\neg\psi \in \mathrm{cl}(\varphi, N_k)$ (for a unary ψ), then $\psi \in \mathrm{cl}(\varphi, N_k)$;
- If $\eta(\bar{a}) = \exists\bar{y} \bigwedge_j \psi_j(\bar{a}, \bar{y}) \in \mathrm{cl}(\varphi, N_k)$, then $\psi \in \mathrm{cl}(\varphi, N_k)$ for all $S \in \mathrm{Spec}(\eta(\bar{a}), N_k)$ and $\psi \in S$.

We are now ready to give a translation of GNFO sentences into automata and show that the automaton size is controlled by the size of the subformula closure.

PROPOSITION A.6. *For every GNFO sentence φ in normal form, every signature σ containing relations of φ, and every $k \in \mathbb{N}$, there is a 2ABTA $\mathcal{A}_\varphi$ on $\Sigma_{\sigma,k}^{code}$-trees such that $\mathcal{A}_\varphi$ accepts a consistent $\Sigma_{\sigma,k}^{code}$-tree $\mathcal{T}$ if and only if the decoding $\mathrm{decode}(\mathcal{T})$ satisfies φ. Moreover, the number of states of the automaton is bounded by the size of $\mathrm{cl}(\varphi, N_k)$, whereas the overall size and the time needed to construct the automaton is at most $f(|\varphi| \cdot |\mathcal{P}(\Sigma_{\sigma,k}^{code})|) \cdot |N_k|^{f(\mathrm{width}(\varphi)\,\mathrm{rank}(\varphi))}$ for some polynomial f independent of φ and k.*

The 2ABTA automaton $\mathcal{A}_\varphi$ for φ is defined as follows:

- The state set is $\mathrm{cl}(\varphi, N_k) \times \{+, -\}$, so every state is a pair of the form $\langle \varphi, + \rangle$ or $\langle \varphi, - \rangle$, where $\psi \in \mathrm{cl}(\varphi, N_k)$ and $+, -$ are polarities that indicate whether ψ comes from a positive or negative part of φ;
- The initial state is $\langle \varphi, + \rangle$;
- The transition function δ is defined further below;
- The accepting states are $\langle \mathrm{true}, + \rangle$, $\langle \mathrm{false}, - \rangle$, $\langle \exists\bar{y}\, \eta(\bar{a}, \bar{y}), - \rangle$, for every relation R and every CQ-shaped formula $\exists\bar{y}\, \eta(\bar{x}, \bar{y}) \in \mathrm{cl}(\varphi, N_k)$.

We now describe the transition function δ. Below, τ is a set of symbols from $\Sigma_{\sigma,k}^{code}$ (e.g., the label of a vertex of the tree code). Given such a set τ and a tuple $\bar{a}$ of names in N_k, we say that $\bar{a}$ is *represented* in τ if τ includes D_{a_i} for every name a_i in $\bar{a}$. Thus, a vertex v labeled with τ that represents $\bar{a}$ has each a_i in $\bar{a}$ as one of its local names:

$$\delta(\langle \mathrm{true}, + \rangle, \tau) := (\mathrm{Stay}, \langle \mathrm{true}, + \rangle)$$
$$\delta(\langle \mathrm{true}, - \rangle, \tau) := (\mathrm{Stay}, \langle \mathrm{true}, - \rangle)$$
$$\delta(\langle \mathrm{false}, + \rangle, \tau) := (\mathrm{Stay}, \langle \mathrm{false}, + \rangle)$$
$$\delta(\langle \mathrm{false}, - \rangle, \tau) := (\mathrm{Stay}, \langle \mathrm{false}, - \rangle)$$

$$\delta(\langle R(\bar{a}), + \rangle, \tau) := \begin{cases} (\mathrm{Stay}, \langle \mathrm{false}, + \rangle) & \text{if } \bar{a} \text{ is not represented in } \tau \\ (\mathrm{Stay}, \langle \mathrm{true}, + \rangle) & \text{if } R_{\bar{a}} \in \tau \\ \bigvee_{d \in \mathrm{Direction}_2}(d, \langle R(\bar{a}), + \rangle) & \text{otherwise} \end{cases}$$

$$\delta(\langle R(\bar{a}), - \rangle, \tau) := \begin{cases} (\mathrm{Stay}, \langle \mathrm{true}, + \rangle) & \text{if } \bar{a} \text{ is not represented in } \tau \\ (\mathrm{Stay}, \langle \mathrm{false}, + \rangle) & \text{if } R_{\bar{a}} \in \tau \\ \bigwedge_{d \in \mathrm{Direction}_2}(d, \langle R(\bar{a}), - \rangle) & \text{otherwise} \end{cases}$$

$$\delta(\langle \bigvee_i \psi_i, + \rangle, \tau) := \bigvee_i (\mathrm{Stay}, \langle \psi_i, + \rangle)$$
$$\delta(\langle \bigvee_i \psi_i, - \rangle, \tau) := \bigwedge_i (\mathrm{Stay}, \langle \psi_i, - \rangle)$$

$$\delta(\langle \alpha \wedge \neg\psi, + \rangle, \tau) := (\text{Stay}, \langle \alpha, + \rangle) \wedge (\text{Stay}, \langle \psi, - \rangle)$$
$$\delta(\langle \alpha \wedge \neg\psi, - \rangle, \tau) := (\text{Stay}, \langle \alpha, - \rangle) \vee (\text{Stay}, \langle \psi, + \rangle)$$
$$\delta(\langle \neg\psi, + \rangle, \tau) := (\text{Stay}, \langle \psi, - \rangle)$$
$$\delta(\langle \neg\psi, - \rangle, \tau) := (\text{Stay}, \langle \psi, + \rangle)$$
$$\delta(\langle \alpha \wedge \psi, + \rangle, \tau) := (\text{Stay}, \langle \alpha, + \rangle) \wedge (\text{Stay}, \langle \psi, + \rangle)$$
$$\delta(\langle \alpha \wedge \psi, - \rangle, \tau) := (\text{Stay}, \langle \alpha, - \rangle) \vee (\text{Stay}, \langle \psi, - \rangle)$$

$$\text{for } \eta(\bar{a}) = \exists \bar{y} \bigwedge_j \psi_j(\bar{a}, \bar{y})$$

$$\delta(\langle \eta(\bar{a}), + \rangle, \tau) := \begin{cases} (\text{Stay}, \langle \text{false}, + \rangle) & \text{if } \bar{a} \text{ is not} \\ & \text{represented in } \tau \\ \\ \bigvee_{d \in \text{Direction}_2}(d, \langle \eta(\bar{a}), + \rangle) \\ \vee \bigvee_{S \in \text{Spec}(\eta(\bar{a}), \text{names}(\tau))} \bigwedge_{\psi \in S}(\text{Stay}, \langle \psi, + \rangle) & \text{otherwise} \end{cases}$$

$$\delta(\langle \eta(\bar{a}), - \rangle, \tau) := \begin{cases} (\text{Stay}, \langle \text{true}, + \rangle) & \text{if } \bar{a} \text{ is not} \\ & \text{represented in } \tau \\ \\ \bigwedge_{d \in \text{Direction}_2}(d, \langle \eta(\bar{a}), - \rangle) \\ \wedge \bigwedge_{S \in \text{Spec}(\eta(\bar{a}), \text{names}(\tau))} \bigvee_{\psi \in S}(\text{Stay}, \langle \psi, - \rangle) & \text{otherwise.} \end{cases}$$

The correctness of the automaton construction is captured in the following result.

LEMMA A.7. *For each $\langle \psi(\bar{a}), + \rangle \in \text{cl}(\varphi, N_k)$, $\psi(\bar{x})$ holds in $\text{decode}(\mathcal{T})$ with valuation $[v]_{\bar{a}}$ for $\bar{x}$ if and only if the automaton above accepts when launched in $\mathcal{T}$ from vertex v with initial state $\langle \psi(\bar{a}), + \rangle$. Likewise, for each $\langle \psi(\bar{a}), - \rangle \in \text{cl}(\varphi, N_k)$, $\psi(\bar{x})$ does not hold in $\text{decode}(\mathcal{T})$ with valuation $[v]_{\bar{a}}$ for $\bar{x}$ if and only if the automaton above accepts when launched in $\mathcal{T}$ from vertex v with initial state $\langle \psi(\bar{a}), - \rangle$.*

PROOF. The lemma is proven by structural induction. The base cases are simple to verify by construction. Lemmas A.4 and A.5 are utilized in the inductive case for CQ-shaped formulas. □

We now calculate the size of $\text{cl}(\varphi, N_k)$.

LEMMA A.8. *Let $\varphi \in$ GNFO in normal form, and let $k \in \mathbb{N}$. Then, $|\text{cl}(\varphi, N_k)| \leq |\varphi| \cdot 2^{\text{rank}(\varphi)} \cdot (2k)^{\text{width}(\varphi)}$.*

PROOF. Let $w = \text{width}(\varphi)$ and $r = \text{rank}(\varphi)$. Note that in the definition of the closure set, the only formulas that appear are either actual subformulas of φ (with names from N_k substituted for free variables) or are formulas that come from specializations of CQ-shaped formulas (again, with names from N_k).

Specializations of CQ-shaped subformulas that do not begin with existential quantification (i.e., CQ-shaped formulas without projection) only contribute as actual subformulas of φ to the closure set. The other specializations η contribute up to 2^r additional CQ-shaped formulas that are based on taking some subset of the (at most) r conjuncts of η.

Since each of these formulas has at most w free variables taking names from $N_k = \{1, \dots, 2k\}$, this means that the overall size of the closure set is at most $|\varphi| \cdot 2^r \cdot (2k)^w$. □

Let us derive complexity bounds for the automaton construction. As usual, let $m = \text{arity}(\sigma)$, $w = \text{width}(\varphi)$, and $r = \text{rank}(\varphi)$ be, respectively, the maximum arity of relations in σ, the width, and the CQ-rank of the formula φ. Since these parameters are all bound by the size of the formula

φ, this means that the cardinality of the closure set $\mathrm{cl}(\varphi, N_k)$, and hence the number of states of the automaton $\mathcal{A}_\varphi$, is at most exponential in the size of φ. In fact, the number of states of $\mathcal{A}_\varphi$ is polynomial when the maximal arity, the width, and the CQ-rank are fixed.

The size of the input alphabet $\mathcal{P}(\Sigma_{\sigma,k}^{\mathrm{code}})$ for the automaton is at most $2^{|\sigma| \cdot (2k)^{\mathrm{arity}\,\sigma}}$, which is doubly exponential in general but singly exponential when the maximal arity is fixed.

The size of each transition function formula is at most linear in $2^w \cdot |N_k|^w \cdot w^w \cdot |\mathrm{cl}(\varphi, N_k)|$. In particular, note that the transition function formula for a CQ-shaped formula ψ respects this bound since $|\mathrm{Spec}(\psi, N_k)|$ is at most $2^w \cdot |N_k|^w \cdot w^w$ (the maximum number of ways to choose the inside variables, names for these inside variables, and the partition of the outside variables), and each $S \in \mathrm{Spec}(\psi, N_k)$ is of size at most $|\mathrm{cl}(\varphi, N_k)|$. This means that the size of the transition function is linear in $|Q| \cdot |\mathcal{P}(\Sigma_{\sigma,k}^{\mathrm{code}})| \cdot 2^w \cdot |N_k|^w \cdot w^w \cdot |Q|$. This is doubly exponential in general but singly exponential when the maximal arity, width, and CQ-rank are fixed.

Therefore, the overall size of $\mathcal{A}_\varphi$ and the time taken to construct it are at most doubly exponential in the size of φ but singly exponential when the maximal arity, width, and CQ-rank are fixed.

From an automaton to decidability. We are now almost done with our satisfiability procedure. Combining Proposition A.6 with Proposition A.2, we see that φ is satisfiable if and only if there is a consistent k-tree code that satisfies $\mathcal{A}_\varphi$, where $k = \mathrm{width}(\varphi)$.

Recall that a consistent $\Sigma_{\sigma,k}^{\mathrm{code}}$-tree is just an arbitrary $\Sigma_{\sigma,k}^{\mathrm{code}}$-tree such that every node v satisfies $|\mathrm{names}(v)| \leq k$ and for all $R_{\bar{a}} \in \Sigma_{\sigma,k}^{\mathrm{code}}$, if $R_{\bar{a}}(v)$, then $\bar{a} \subseteq \mathrm{names}(v)$. It is straightforward to see that there is a 2ABTA automaton $\mathcal{A}_{\mathrm{consistent}}$ that accepts exactly the trees that are consistent in the above sense. The size of $\mathcal{A}_{\mathrm{consistent}}$ is doubly exponential (due to the size of the alphabet) and singly exponential if the maximal arity of each relation is fixed. The running time needed to form the automaton is likewise doubly exponential in general and singly exponential when the arity of relations is fixed. Further, the number of states is just 2—an initial state and a "rejection" state representing a violation of consistency.

By the closure properties of 2ABTA, we know that we can form an automaton $\mathcal{A}_{\varphi,\mathrm{consistent}}$ that accepts the intersection $L(\mathcal{A}_\varphi) \cap L(\mathcal{A}_{\mathrm{consistent}})$ in time proportional to the sum of the sizes of $\mathcal{A}_\varphi$ and $\mathcal{A}_{\mathrm{consistent}}$. The number of states of this automata is just the sum of the number states of $\mathcal{A}_\varphi$ and $\mathcal{A}_{\mathrm{consistent}}$. Hence, by applying Theorem A.3, we can conclude the following.

THEOREM A.9. *There is a* 2ExpTime *satisfiability testing algorithm for GNFO sentences in normal form without equality and constants. When the width, CQ-rank, and maximal arity of the relations are fixed, it shrinks to* ExpTime.

A.4 Handling Equality and Constants

The extension to handle equalities, in the absence of constants, is not difficult. We consider the same tree codes as before.

We claim again that if a sentence φ in GNFO with width bounded by k is satisfiable, then it is satisfied in a structure with a k-code.

The conversion to normal form is the same, treating equality like any other relation.

In the automaton construction, we need additional cases for equality.

$$\delta(\langle a = b, + \rangle, \tau) := \begin{cases} (\mathrm{Stay}, \langle \mathrm{true}, + \rangle) & \text{if } a \text{ is the same as } b \\ (\mathrm{Stay}, \langle \mathrm{false}, + \rangle) & \text{if } a \text{ is not the same as } b \end{cases}$$

$$\delta(\langle a = b, - \rangle, \tau) := \begin{cases} (\mathrm{Stay}, \langle \mathrm{false}, + \rangle) & \text{if } a \text{ is the same as } b \\ (\mathrm{Stay}, \langle \mathrm{true}, + \rangle) & \text{if } a \text{ is not the same as } b \end{cases}$$

The size bounds and running time of the construction remain the same.

Constants. We now deal with GNFO formulas that can have logical constant symbols and equalities. The requirement that negation be guarded is the same as before: the free variables of the negated formula must occur in a guard, saying nothing about the constants. Thus, in the extension of GNFO with constants, we can freely express that two constants are not equal, and in particular we can express that *all* distinct constants are unequal (the "UNA" we deal with in the body of the article). To handle constants requires some additional effort. One route to decidability, taken in the work of Bárány et al. [7], is to reduce satisfiability of GNFO with equality and constants to satisfiability without constants. The idea of the reduction by Bárány et al. [7] is to extend the signature with additional predicates that hold the constants. However, using such a reduction as a black box does not give us the fine-grained bounds we desire in terms of parameters like CQ-rank. We thus provide a more direct argument.

We consider k-tree codes in which the constants $\mathrm{Const}(\sigma)$ are represented in each node, along with at most k local names. The codes will also now include some equality facts, but with the following restrictions:

- There are no equality facts relating non-constants to each other, and no equality facts relating constants to non-constants.
- The equality facts on constants are identical across vertices of the tree. They satisfy transitivity and reflexivity, as well as *congruence*: if we have a fact $R(\ldots c \ldots)$ holding in a vertex, where c is a constant, and we also have an equality fact $c = d$, then we have the fact $R(\ldots d \ldots)$.

We can extend $\mathcal{A}_{\mathrm{consistent}}$ to check whether a tree is a code satisfying these additional restrictions.

We must change the notion of decoding of a tree to account for equalities. For a consistent tree $\mathcal{T}$ using local names and constants $\mathrm{Const}(\sigma)$, we let $\mathrm{Const}(\sigma)_{=,\mathcal{T}}$ be the equivalence classes of constants under the equality relation in $\mathcal{T}$. The decoding $\mathrm{decode}(\mathcal{T})$ is now the σ-structure with universe

$$\{[v, a] : v \in \mathrm{dom}(\mathcal{T}) \text{ and } a \in \mathrm{names}(v)\} \cup \mathrm{Const}(\sigma)_{=\mathcal{T}}$$

such that for each relation R, we have $R^{\mathrm{decode}(\mathcal{T})}([v_1, a_1], \ldots, [v_j, a_j], e_1 \ldots e_l)$, where a_i are local names and e_i are equivalence classes of constants, if and only if there is $w \in \mathrm{dom}(\mathcal{T})$ such that $R_{\bar{a}, c_1 \ldots c_l}(w)$ holds, $[w, a_i] = [v_i, a_i]$ for all $i \le j$ and c_i is in class e_i for each $i \le l$.

We further claim the following extension of Proposition A.2 to formulas with constants.

PROPOSITION A.10. *Let φ be a GNFO sentence in normal form, having width k, and possibly using equality and constants. If φ is satisfiable, then it is satisfiable in a structure that is the decoding of some k-code.*

PROOF. Consider an expanded signature where for each relation R of arity n and partial function h from the positions of R into constants, we have a relation R_h of arity $n - |dom(h)|$. We can rewrite φ to a φ' in this signature that does not contain constants, replacing atoms $R(x_1 \ldots x_n)$ by a disjunction of atoms over R_h where h varies over every partial function, and replacing subformulas with negation guarded by an R-atom with a disjunction of subformulas guarded by an R_h-atom. Note that φ' will be larger than φ, but its width will still be k. Thus, applying Proposition A.2, we see that φ has a model M' with a k-code in the expanded signature. But then we can reverse this process on M, replacing atoms R_h in M with an atom R but using the additional constants as arguments. We can similarly add the equality facts to the codes. Since equality in M' must satisfy congruence, reflexivity, and transitivity, we will obtain a structure satisfying the additional properties.   □

The closure is now defined as before, but based on $N_k \cup \text{Const}(\sigma)$ rather than N_k.
In the automaton construction, we need a few modifications:
We need a base case for equality atoms.

- For a non-negated equality of a local name with a constant, the automaton should ensure rejection: it does this by switching to state $\langle \text{false}, + \rangle$, since there are no accepting runs from such states. Similarly for a negated equality of a local name with a constant, the automaton should ensure acceptance by switching to state $\langle \text{true}, + \rangle$.
- For an equality between constants, the automaton simply checks whether the equality is present in the vertex; if this is true, the automaton should ensure acceptance. It does this by switching to state $\langle \text{true}, + \rangle$. Otherwise, it ensures rejection by switching to state $\langle \text{false}, + \rangle$.

In other words, for a name $a \in N_k$ and for constants $c, d \in \text{Const}(\sigma)$, we have the following transitions:

$$\delta(\langle a = c, + \rangle, \tau) := (\text{Stay}, \langle \text{false}, + \rangle)$$
$$\delta(\langle a = c, - \rangle, \tau) := (\text{Stay}, \langle \text{true}, + \rangle)$$

$$\delta(\langle c = d, + \rangle, \tau) := \begin{cases} (\text{Stay}, \langle \text{true}, + \rangle) & \text{if } c = d \in \tau \\ (\text{Stay}, \langle \text{false}, + \rangle) & \text{if } c = d \notin \tau \end{cases}$$

$$\delta(\langle c = d, - \rangle, \tau) := \begin{cases} (\text{Stay}, \langle \text{false}, + \rangle) & \text{if } c = d \in \tau \\ (\text{Stay}, \langle \text{true}, + \rangle) & \text{if } c = d \notin \tau. \end{cases}$$

We also modify the CQ-shaped formula case, to allow the automaton to draw witnesses from the constants:

$$\delta(\langle \exists \bar{y}\, \eta(\bar{a}, \bar{y}), + \rangle, \tau) := \begin{cases} (\text{Stay}, \langle \text{false}, + \rangle) & \text{if } \bar{a} \text{ is not represented in } \tau \\ \bigvee_{S \in \text{Spec}(\exists \bar{y}\, \eta(\bar{a}, \bar{y}), \text{names}(\tau) \cup \text{Const}(\sigma))} \bigwedge_{\psi \in S} (\text{Stay}, \langle \psi, + \rangle)\ \vee \\ \quad \bigvee_{d \in \text{Direction}_2} (d, \langle \exists \bar{y}\, \eta(\bar{a}, \bar{y}), + \rangle) & \text{otherwise} \end{cases}$$

$$\delta(\langle \exists \bar{y}\, \eta(\bar{a}, \bar{y}), - \rangle, \tau) := \begin{cases} (\text{Stay}, \langle \text{true}, + \rangle) & \text{if } \bar{a} \text{ is not represented in } \tau \\ \bigwedge_{S \in \text{Spec}(\exists \bar{y}\, \eta(\bar{a}, \bar{y}), \text{names}(\tau) \cup \text{Const}(\sigma))} \bigvee_{\psi \in S} (\text{Stay}, \langle \psi, - \rangle)\ \wedge \\ \quad \bigwedge_{d \in \text{Direction}_2} (d, \langle \exists \bar{y}\, \eta(\bar{a}, \bar{y}), - \rangle) & \text{otherwise.} \end{cases}$$

Using these modifications, we can now extend Lemma A.7.

LEMMA A.11. *For each $\langle \psi(\bar{a}, \bar{c}), + \rangle \in \text{cl}(\varphi, N_k)$, $\psi(\bar{x}, \bar{y})$ holds in $\text{decode}(\mathcal{T})$ at vertex v with valuation $[v, \bar{a}]$ for $\bar{x}$ and constants $c_1 \ldots c_l$ for $\bar{y}$ if and only if the automaton above accepts when launched in $\mathcal{T}$ from vertex v with initial state $\langle \psi(\bar{a}, \bar{c}), + \rangle$. Likewise, for each $\langle \psi(\bar{a}, \bar{c}), - \rangle \in \text{cl}(\varphi, N_k)$, $\psi(\bar{x}, \bar{y})$ does not hold in $\text{decode}(\mathcal{T})$ at vertex v with the valuation above if and only if the automaton above accepts when launched in $\mathcal{T}$ from vertex v with initial state $\langle \psi(\bar{a}, \bar{c}), - \rangle$.*

Recall that the proof of Lemma A.7 worked by induction on ψ. In the proof, we first need to consider base cases for equality. For example, suppose that $x_1 = x_2$ holds in $\text{decode}(\mathcal{T})$ with valuation $x_1 = [v, a_1]\ x_2 = [v, a_2]$ for local names a_1, a_2. The only way the equality can hold is if a_1 is actually the same name as a_2. Thus, the automaton run from $\langle a = b, + \rangle$ will transition to $\langle \text{true}, + \rangle$, and will accept. The converse direction is similar.

However, suppose that $x_1 = x_2$ holds in $\text{decode}(\mathcal{T})$ with valuation $x_1 = [c]_{=, \mathcal{T}}$, $x_2 = [d]_{=, \mathcal{T}}$ for constants c, d. This holds exactly when the equality fact $c = d$ is present in the label of v. But then looking at the transition function for $\langle c = d, + \rangle$ we see that the automaton accepts.

We must also reconsider the base cases for atomic relations. Suppose that $R(x_1, \ldots x_j, y_1 \ldots y_l)$ holds in $\text{decode}(\mathcal{T})$ with valuation $x_i = [v, a_i], y_i = [c_i]_{=, \mathcal{T}}\}$ for local names $\bar{a}$ and constants $\bar{c}$.

By definition of our decoding, along with the congruence closure of the codes, this means that we must have a fact $R([v, \bar{a}], \bar{c})$ holding in some node v' in the tree. We now argue as in the case without constants that iterating the transition function for an atom, the automaton will accept from v.

PROPOSITION A.12. φ *is satisfiable if and only if the modified automaton* $\mathcal{A}_\varphi$ *accepts a consistent tree. This in turn can be checked by taking the automaton* $\mathcal{A}_{consistent}$ *for checking consistency, forming an automaton* $\mathcal{A}'_\varphi$ *accepting the intersection of* $\mathcal{A}_{consistent}$ *with* $\mathcal{A}_\varphi$, *and checking non-emptiness of* $\mathcal{A}'_\varphi$.

Thus, we obtain the following result.

THEOREM A.13. *There is a* 2EXPTIME *algorithm for deciding satisfiability of sentences in GNFO, even allowing equality and constants. For a sentence in normal form with fixed width, CQ-rank, and fixed arity of relations, we get an* EXPTIME *algorithm for satisfiability.*

A.5 Lifting the Fixed Arity Restriction

From what we have seen in the previous sections, we can almost infer the EXPTIME bound claimed in Theorem 4.3 in the body of the article. The only gap is that thus far we have restricted the maximal arity of relations. This is actually unproblematic in the application to the data complexity bounds in the body. But it is an unnecessary restriction for lowering the complexity, as we now show.

We argue that we can reduce satisfiability for GNFO formulas of unbounded arity to the setting with fixed arity, preserving the bounds on the other parameters.

Let φ be any GNFO formula in normal form. We will show how to construct, in polynomial time, another GNFO formula ψ, such that

(1) φ and ψ are equi-satisfiable,
(2) width(ψ) ≤ width(φ) + rank(φ)
(3) rank(ψ) ≤ rank(φ) · width(φ)
(4) ψ uses only binary relations.

For each n-ary relation R occurring in φ, we introduce binary relations R_i for $i = 1, \ldots, n$. We replace each atomic formula $R(t_1, \ldots, t_n)$ in φ by $\exists y \bigwedge_i R_i(y, t_i)$, for a fresh variable y. Let χ be the resulting formula. Note that χ may not be in normal form, because the newly introduced existential quantifiers may occur directly below a conjunction.

Our formula ψ is obtained from χ by the following operations:

(1) Let $w = \text{width}(\varphi)$, and consider any atomic formula $R(t_1, \ldots, t_n)$ occurring in φ. Then, among the terms $t_1, \ldots, t_n$, there are at most w distinct variables. Therefore, the corresponding formula $\exists y \bigwedge_i R_i(y, t_i)$ in χ can be rewritten as follows: whenever a term t_i is a constant, the corresponding conjunct $R_i(y, t_i)$ (having only one free variable) can be rewritten to $\neg\neg R_i(y, t_i)$. If terms $t_{i_1}, \ldots, t_{i_k}$ are all the same variable, then we can rewrite the corresponding conjunction $R_{i_1}(y, t_{i_1}) \wedge \cdots \wedge R_{i_k}(y, t_{i_k})$ to $R_{i_1}(y, t_{i_1}) \wedge \neg \bigvee_{j=2,\ldots,n} R_{i_1}(y, t_{i_1}) \wedge \neg R_{i_j}(y, t_{i_j}))$. All of this has the effect of "hiding behind a guarded double negation" all but at most w of the conjuncts in question.
(2) We pull out, as needed, these existential quantifiers, using the equivalence $\alpha \wedge \exists y \beta \equiv \exists y (\alpha \wedge \beta)$, to bring the formula in normal form.

It is easy to see that φ' is in normal form, and that the construction is in polynomial time. The argument that φ' is equi-satisfiable to φ is also easy: given a structure M' for the signature of φ', we create a structure M by taking a tuple $y, t_1 \ldots t_n$ witnessing $\bigwedge_i R_i(y, t_i)$ in M' and creating a fact

$R(t_1 \ldots t_n)$ in M. An induction argument shows that if $M' \models \varphi$ then $M \models \varphi$. In the other direction, given a structure M for the language of φ we create M' by "shredding" R facts into a sequence of facts for $R_1 \ldots R_n$.

The width of ψ may be larger than the width of φ, due to the newly introduced existential variables. However, we introduce at most one such variable per conjunct, and therefore it is easy to see that $\mathrm{width}(\psi) \leq \mathrm{width}(\varphi) + \mathrm{rank}(\varphi)$.

Finally, it is easy to see that $\mathrm{rank}(\psi)$ is bounded by $w \cdot \mathrm{rank}(\varphi)$, where $w = \mathrm{width}(\varphi)$, due to our construction in the first item above.

Applying this reduction, we obtain the following main result, which immediately implies Theorem 4.3 in the body of the article.

THEOREM A.14. *There is a* 2ExpTime *algorithm for deciding satisfiability of sentences in GNFO, and for every fixed width and CQ-rank, there is an* ExpTime *satisfiability testing algorithm for GNFO sentences in normal form.*

A.6 Additional Remarks: Relationship to Bounds for General GNFO

We note that the previous result allows us to re-prove the bounds for satisfiability of GNFO sentences that are not in normal form from Bárány et al. [6, 7]. We include this only because it might be useful to have a self-contained presentation of the GNFO-to-automata translation. The idea is that general GNFO sentences can be converted to normal form in such a way that we blow up the size of the formula, but the size of the closure set remains at most exponential in the size of the original formula. The following proposition shows this in detail and can be seen as a strengthening of Proposition 4.2.

PROPOSITION A.15. *Let ψ be a GNFO formula with $m = |\psi|$. We can construct, in exponential time, a sentence* $\mathrm{convert}(\psi)$ *in normal form equivalent to ψ such that*
- $|\mathrm{convert}(\psi)| \leq 2^{f(m)}$,
- $\mathrm{width}(\mathrm{convert}(\psi)) \leq m$,
- $\mathrm{rank}(\mathrm{convert}(\psi)) \leq m$,
- $|\mathrm{cl}(\mathrm{convert}(\psi), N_m)| \leq 2^{f(m)}$,

where f is a polynomial function independent of ψ.

PROOF. We first push disjunctions outside, as much as possible, by distributing with conjunctions and existential quantifications. We then push existential quantifications outside through conjunctions (this may require some variable renaming). After this rewriting, every subformula under a conjunction must be either an atom or a guarded negation, and thus it satisfies the form ψ of the above grammar. □

Now when we apply the automaton construction of Proposition A.6 to the output, we will get an automaton with state set $\mathrm{cl}(\mathrm{convert}(\psi), N_{|\psi|})$. By the above, the size of this is bounded by an exponential in the size of the original formula ψ. The size of the automaton alphabet is unaffected by this transformation. Thus, again we can apply Theorem A.3 to get a doubly exponential algorithm for testing satisfiability.

COROLLARY A.16 ([7]). *There is a* 2ExpTime *satisfiability testing algorithm for GNFO sentences without equality.*

B PSpace-COMPLETENESS OF ∃PQI FOR IDS: PROOF OF THEOREM 4.20

Recall the statement of Theorem 4.20:

The problem ∃PQI$(Q, \Sigma, \mathbf{S})$, where Q ranges over Boolean UCQs without constants and Σ over sets of IDs, is PSpace-complete. Hardness holds even in the case of Boolean CQs without constants.

We first prove the upper bound, using a rewriting technique similar to the one used in earlier work [10], Theorem 5. By Theorem 4.11, ∃PQI(Q, Σ, S) holds if and only if PQI($Q, \Sigma, S, \mathcal{V}_{\{a\}})$) holds. Therefore, it suffices to show that we can test the latter in PSpace.

By the *position graph*, we will mean the directed graph whose nodes are all pairs (R, i) where R is a relation in S and $i \leq \text{arity}(R)$, and such that there is an edge from (R, i) to (S, j) if Σ contains an ID in which some variable x appears in position (R, i) in the left and position (S, j) in the right side, at the expense of allowing constants in the IDs. We call a position (R, i) *exposed* if, in the position graph, there is a path from (R, i) to some (S, j) for S a visible relation.

It follows from the construction of chase$_{\text{vis}}(\mathcal{V}_{\{a\}})$ that.

Claim 1. In chase$_{\text{vis}}(\mathcal{V}_{\{a\}})$, the only value appearing in exposed positionsis the critical element a.

Based on this observation, we will show that we can eliminate all IDs whose right-hand side contains a visible relation. First, we modify our schema S to remove all exposed attributes from hidden relations. We denote the resulting schema by S'. We also add a new unary relation IsCrit(x). Note that, as with any unary predicate, the extension of IsCrit in the critical instance will consist only of the critical element a.

We modify the constraints in Σ accordingly, by dropping the argument corresponding to exposed attributes from all atoms. It may happen, in doing so, that some universally quantified variable appearing in the right-hand side of an ID no longer occurs in the left-hand side (because all its occurrences were in exposed positions). In this case, we replace the variable in question by the constant a. The resulting set of constraints (over schema S') is denoted by Σ'. Note that the constraints in Σ' are IDs, except they may contain constants in their right-hand sides. Finally, in the query Q, we modify all atoms $R(x_1, \ldots, x_n)$ over hidden relations by dropping the x_i that are in an exposed position and adding conjuncts IsCrit(x_i) instead. We denote the resulting query by Q'.

Claim 2. PQI($Q, \Sigma, S, \mathcal{V}_{\{a\}})$) holds if and only if PQI($Q', \Sigma', S', \mathcal{V}_{\{a\}})$) holds.

(Note that, in the statement of Claim 2, we allow ourselves to be a little sloppy in our notation: the first $\mathcal{V}_{\{a\}}$ is over schema S, whereas the second one is over S').

The proof of Claim 2 is straightforward: a counterexample to PQI($Q, \Sigma, S, \mathcal{V}_{\{a\}})$) is transformed into a counterexample for PQI($Q', \Sigma', S', \mathcal{V}_{\{a\}})$) by projecting out the exposed positions, whereas a counterexample for PQI($Q', \Sigma', S', \mathcal{V}_{\{a\}})$) is transformed into a counterexample for PQI($Q, \Sigma, S, \mathcal{V}_{\{a\}})$) by inserting constant a in all exposed positions (as justified by Claim 1).

Next, observe that (i) whenever a constraint in Σ' has a visible relation in its right-hand side, then the left- and right-hand side of the constraint in question do not share any variables; (ii) if $\mathcal{F}$ is any instance (over schema S') that contains all of the facts in $\mathcal{V}_{\{a\}}$, then all such constraints are satisfied in $\mathcal{F}$. It follows from these two observations that we can remove from Σ' all constraints with right-hand sides that are visible relations, and that doing so we do not affect chase$_{\text{vis}}(\mathcal{V}_{\{a\}})$. Let Σ'' therefore be the result of dropping from Σ' all constraints that derive into visible relations. Then, we have the following claim.

Claim 3. PQI($Q, \Sigma, S, \mathcal{V}_{\{a\}})$) holds if and only if PQI($Q', \Sigma'', S', \mathcal{V}_{\{a\}})$) holds.

Since Σ'' does not contain any IDs whose right-hand sides are atoms over visible relations, we have that PQI($Q', \Sigma'', S', \mathcal{V}_{\{a\}})$) holds, if and only if the query containment $Q_1 \subseteq Q_2$ holds relative to the constraints Σ'', where Q_1 is the canonical query of the instance $\mathcal{V}_{\{a\}}$ and Q_2 is the query Q'.

It was shown in the work of Casanova et al. [24] that query containment under IDs (without constants) is PSpace-complete. Inspection of the upper bound proof in their work [24] (which is a straightforward non-deterministic polynomial-space bounded algorithm based on the chase)

shows that the presence of constants does not affect the argument. Therefore, we conclude that our problem is in PSpace.

Next, we prove the lower bound. This is already claimed in earlier work [10] but without proof details, so we spell out an argument here. For this, we provide a reduction from the implication problem for IDs: given a finite collection of IDs $\sigma_1, \ldots, \sigma_n$ and an ID σ, do $\sigma_1, \ldots, \sigma_n$ logically imply σ? This problem is known to be PSpace-complete [24]. Given $\sigma_1, \ldots, \sigma_n, \sigma$, with σ of the form $\forall \bar{x}\, R(\bar{x}) \rightarrow \exists \bar{y} S(\bar{z})$, we construct an instance $\exists PQI(Q, \Sigma, S)$ as follows:

- Q is the query $\exists \bar{x} \bar{y}\, R'(\bar{x}) \wedge S(\bar{z})$.
- Σ consists of the IDs $\sigma_1, \ldots, \sigma_n$ together with the IDs visible() $\rightarrow \exists \bar{x} R'(\bar{x})$ and $R'(\bar{x}) \rightarrow R(\bar{x})$.
- S is the schema consisting of all relations occurring in $\sigma_1, \ldots, \sigma_n, \sigma$ as well as R' and visible. All relations are treated as hidden, except for visible.

It is easy to show that $\exists PQI(Q, \Sigma, S)$ holds if and only if $\sigma_1, \ldots, \sigma_n$ logically imply σ.

C   2ExpTime-HARDNESS OF PQI IN COMBINED COMPLEXITY: PROOF OF THEOREM 4.8

Recall the following statement:

Checking $PQI(Q, \Sigma, S, \mathcal{V})$, where Q ranges over Boolean CQs without constants and Σ ranges over sets of IDs, is 2ExpTime-hard for combined complexity.

Proof. This proof builds on ideas from the proof for Corollary 4.7 in the body of the article. Specifically, we reduce the acceptance problem for an alternating ExpSpace Turing machine M to the negation of $PQI(Q, \Sigma, S, \mathcal{V})$, where Q is a Boolean UCQ and Σ consists of IDs. Note that to further reduce the problem to a PQI problem with a Boolean CQ, one can exploit Lemma 4.6.

The additional technical difficulty here is to encode a tape of exponential size. Of course, this cannot be done succinctly using an instance with visible relations. However, we can represent the exponential tape by a set of tuples of bits. More precisely, given an alternating ExpSpace Turing machine M and an input for M of length n, we identify each cell of the tape of M by an n-tuple of bits. Note that, differently from the reduction in Theorem 4.5, here we can let the schema, the sentences, and the query depend on M and n, since the goal here is to prove a lower bound for combined complexity.

For the sake of simplicity, we first explain how to create a single tape of exponential length, without being concerned about the content of the cells and the different configurations that can be reached by M. For this, we introduce three visible relations Zero, One, and Bit, instantiated with $\{0\}$, $\{1\}$, and $\{0, 1\}$, respectively. We also introduce hidden relations $T_i, T_{i,\text{zero}}, T_{i,\text{one}}$ of arity i, for all $i = 1, \ldots, n$, and an additional hidden relation T_0 of arity 0. Intuitively, the intended semantics of each relation T_i is to contain all i-tuples of bits, whereas $T_{i,\text{zero}}$ (respectively, $T_{i,\text{one}}$) is the restriction of T_i to the tuples ending with 0 (respectively, 1). We enforce this semantics using a simple induction on $i = 1, \ldots, n$ and the following IDs:

$$
\begin{array}{rclcrcl}
 & \text{true} & \rightarrow & T_0() & T_{i,\text{zero}}(y_1, \ldots, y_i) & \rightarrow & \text{Zero}(y_i) \\
(\forall j \leq i) & T_i(y_1, \ldots, y_i) & \rightarrow & \text{Bit}(y_j) & T_{i,\text{one}}(y_1, \ldots, y_i) & \rightarrow & \text{One}(y_i) \\
\\
T_{i-1}(y_1, \ldots, y_{i-1}) & \rightarrow & \exists y_i\, T_{i,\text{zero}}(y_1, \ldots, y_i) & & T_{i,\text{zero}}(y_1, \ldots, y_i) & \rightarrow & T_i(y_1, \ldots, y_i) \\
T_{i-1}(y_1, \ldots, y_{i-1}) & \rightarrow & \exists y_i\, T_{i,\text{one}}(y_1, \ldots, y_i) & & T_{i,\text{one}}(y_1, \ldots, y_i) & \rightarrow & T_i(y_1, \ldots, y_i).
\end{array}
$$

It is clear that every instance satisfying the above sentences will have $T_n = \text{Bit}^n$, so the tuples in T_n can be used to represent the cells of a tape of exponential length.

Cells are naturally ordered in the tape, and so must be the tuples in T_n. We use the lexicographic order on n-tuples of bits and show how to access this order by means of a formula. Formally, we need to write a UCQ that checks whether two cells, identified by some n-tuples $\bar{y} = (y_1, \ldots, y_n)$ and $\bar{y}' = (y_1', \ldots, y_n')$ in T_n, are adjacent according to the lexicographic ordering. A well-known technique consists in determining the smallest index $1 \leq i \leq n$ such that $y_i \neq y_i'$. Then, given such i, one verifies that $y_i = 0$, $y_i' = 1$, $y_j = 1$, and $y_j' = 0$ for all $j > i$. We give beforehand the formula that checks these conditions. The formula is the disjunction over all $i = 1, \ldots, n$ of the following CQs:

$$Q_{\mathrm{adj},\, i}(\bar{y}, \bar{y}') \;=\; \bigwedge_{1 \leq j < i} (y_j = y_j') \,\wedge\, \mathsf{Zero}(y_i) \,\wedge\, \mathsf{One}(y_i') \,\wedge\, \bigwedge_{i < j \leq n} \mathsf{One}(y_j) \,\wedge\, \bigwedge_{i < j \leq n} \mathsf{Zero}(y_j').$$

Here, for convenience of description, we allow equalities in a CQ, but they can be replaced in favor of an explicit substitution. It is not difficult to see that the UCQ $\bigvee_{1 \leq i \leq n} Q_{\mathrm{adj},\, i}$ defines precisely those pairs of tuples that are consecutive in the lexicographic order. Moreover, we will need to easily identify the first and the last cell of the tape. For this, we introduce two visible relations First and Last, both of arity n, and instantiate them with the singletons $\{(0, \ldots, 0)\}$ and $\{(1, \ldots, 1)\}$, respectively.

Now that we know how to represent exponentially many cells in the tape and check their adjacency, we proceed as in the proof of Theorem 4.5. We begin by encoding configurations of M. Intuitively, the goal is to create a copy C of the relation T_n, expanded with configuration identifiers and cell values, in such a way that a fact of the form $C(x, y_1, \ldots, y_n, z)$ denotes the existence of a configuration identified by x, where the tape cell represented by $\bar{y} = (y_1, \ldots, y_n)$ carries the value z. As usual (cf. proof of Theorem 4.5), we define cell values as elements from a visible unary relation $V = \Sigma_Q \uplus \Sigma_\lhd \uplus \Sigma_\rhd$, where Σ is the alphabet of the Turing machine, $\Sigma_Q = \Sigma \times Q$. $\Sigma_\lhd = \Sigma \times \{\lhd\}$, $\Sigma_\rhd = \Sigma \times \{\rhd\}$, Q is the set of its control states, and $\lhd, \rhd$ are fresh symbols. To correctly instantiate the relation C, we create also copies of the relations $T_i, T_{i,\mathrm{zero}}, T_{i,\mathrm{one}}$, expanded with configuration identifiers, and enforce constraints analogous to the ones introduced in the sentences above. More precisely, we have the following hidden relations: C of arity $n + 2$, T_i^C of arity $i + 1$, for all $i = 0, \ldots, n$, $T_{i,\mathrm{zero}}^C$ and $T_{i,\mathrm{one}}^C$ of arity $i + 1$, for all $i = 1, \ldots, n$. We have the following sentences for all $i = 1, \ldots, n$:

$$(\forall j \leq i) \qquad T_i^C(x, y_1, \ldots, y_i) \;\rightarrow\; \mathsf{Bit}(y_j)$$

$$T_n^C(x, y_1, \ldots, y_n) \rightarrow \exists z \, C(x, y_1, \ldots, y_n, z) \qquad T_{i,\mathrm{zero}}^C(x, y_1, \ldots, y_i) \rightarrow \mathsf{Zero}(y_i)$$

$$C(x, y_1, \ldots, y_n, z) \rightarrow V(z) \qquad T_{i,\mathrm{one}}^C(x, y_1, \ldots, y_i) \rightarrow \mathsf{One}(y_i)$$

$$T_{i-1}^C(x, y_1, \ldots, y_{i-1}) \rightarrow \exists y_i \, T_{i,\mathrm{zero}}^C(x, y_1, \ldots, y_i) \qquad T_{i,\mathrm{zero}}^C(x, y_1, \ldots, y_i) \rightarrow T_i^C(x, y_1, \ldots, y_i)$$

$$T_{i-1}^C(x, y_1, \ldots, y_{i-1}) \rightarrow \exists y_i \, T_{i,\mathrm{one}}^C(x, y_1, \ldots, y_i) \qquad T_{i,\mathrm{one}}^C(x, y_1, \ldots, y_i) \rightarrow T_i^C(x, y_1, \ldots, y_i).$$

Note that the analog of the sentence $\mathsf{true} \rightarrow T_0()$ is missing here. This will be given later, when we will explain how new configurations are created to simulate a computation tree of M. For the moment, it suffices to observe that, in every instance that satisfies the above sentences, as soon as T_0^C contains a configuration identifier x, then T_n^C contains all tuples of the form $(x, y_1, \ldots, y_n)$, with $(y_1, \ldots, y_n) \in \mathsf{Bit}^n$, and C specifies at least one value z for each configuration identifier x and each cell $(y_1, \ldots, y_n)$.

We now turn toward the encoding of the computation tree of M. This is almost the same as in the proof of Theorem 4.5. We introduce a visible unary relation I, which contains the identifier x_0 of the initial existential configuration, and three hidden binary relations $S^\exists$, $S_1^\forall$, and $S_2^\forall$. A fact of the form $S^\exists(x, x')$ (respectively, $S_1^\forall(x, x_1)$, $S_2^\forall(x, x_1)$) represents a transition from an existential (respectively, universal) configuration x to a universal (respectively, existential) configuration x' (respectively, x_1, x_2). We then include the following sentences in the background theory:

$$
\begin{aligned}
I(x) &\to \exists\, x'\, S^{\exists}(x, x') & S^{\exists}(x, x') &\to T_0^C(x) \\
S^{\exists}(x, x') &\to \exists\, x_1\, S_1^{\forall}(x', x_1) & & \\
S^{\exists}(x, x') &\to \exists\, x_2\, S_2^{\forall}(x', x_2) & S_1^{\forall}(x, x_1) &\to T_0^C(x) \\
S_1^{\forall}(x, x_1) &\to \exists\, x'\, S^{\exists}(x_1, x') & & \\
S_2^{\forall}(x, x_2) &\to \exists\, x'\, S^{\exists}(x_2, x') & S_2^{\forall}(x, x_2) &\to T_0^C(x).
\end{aligned}
$$

Intuitively, the rules on the left enforce the existence of a transition graph where $x_0 \in I$ is the initial node and every node has one or two outgoing edges, depending on whether it is existential or universal. The rules on the right trigger the instantiation of the tables T_n^C and C, with the intended goal of representing the content of the tape associated with each node/configuration. As usual, the unfolding of the transition graph from the initial node yields a tree, which should represent a computation of M.

It remains to describe how we detect badly formed encodings of computations of M. For this, we introduce new visible relations Err_C, $\mathrm{Err}_{I,\mathrm{first}}$, $\mathrm{Err}_{I,\mathrm{last}}$, $\mathrm{Err}_{I,\mathrm{adj}}$, $\mathrm{Err}_{C,\mathrm{adj}}$, $\mathrm{Err}_{S^{\exists}}$, $\mathrm{Err}_{S_1^{\forall}}$, and $\mathrm{Err}_{S_2^{\forall}}$, whose instances are defined exactly as in the proof of Theorem 4.5:

- The relation Err_C is binary and contains all pairs of distinct values from $V \times V$. This is used to detect multiple values associated with the same cell:

$$
Q_C = \exists\, x\, \bar{y}\, z\, z'\ C(x, \bar{y}, z) \wedge C(x, \bar{y}, z') \wedge \mathrm{Err}_C(z, z').
$$

- The relation $\mathrm{Err}_{I,\mathrm{first}}$ contains all pairs in $V \times V$ but (z_0, z_1), where $z_0 = (\vdash, q_0)$ and $z_1 = (\sqcup, \rhd)$. This is used to detect wrong values associated with the first two cells of the initial configuration:

$$
\begin{aligned}
Q_{I,\mathrm{first}} = \exists\, x\, \bar{y}\, \bar{y}'\, z\, z' \\
I(x) \wedge \mathrm{First}(\bar{y}) \wedge \bigvee_{1 \le i \le n} Q_{\mathrm{adj},i}(\bar{y}, \bar{y}') \wedge \\
C(x, \bar{y}, z) \wedge C(x, \bar{y}', z') \wedge \mathrm{Err}_{I,\mathrm{first}}(z, z').
\end{aligned}
$$

  Note that, strictly speaking, the above query is not a UCQ but can be easily normalized into a UCQ of polynomial size. The same remark applies to all remaining queries.

- Similar visible relations $\mathrm{Err}_{I,\mathrm{last}}$, $\mathrm{Err}_{I,\mathrm{adj}}$, $\mathrm{Err}_{C,\mathrm{adj}}$ and UCQs $Q_{I,\mathrm{last}}$, $Q_{I,\mathrm{adj}}$, $Q_{C,\mathrm{adj}}$ are used to detect wrong values, respectively, for the last two cells of the initial configuration, for any two adjacent cells of the initial configuration, and for any two adjacent cells of an arbitrary configuration.

- To detect the violations that involve values associated with the same position of the tape but in two consecutive configurations, we use the following UCQs:

$$
\begin{aligned}
Q_{S^{\exists}} = \exists\, x\, x'\, \bar{y}\, \bar{y}'\, \bar{y}''\, z\, z'\, z''\, z''' \\
S^{\exists}(x, x') \wedge \bigvee_{1 \le i \le n} Q_{\mathrm{adj},i}(\bar{y}, \bar{y}') \wedge \bigvee_{1 \le i \le n} Q_{\mathrm{adj},i}(\bar{y}', \bar{y}'') \wedge \\
C(x, \bar{y}, z) \wedge C(x, \bar{y}', z') \wedge C(x, \bar{y}'', z'') \wedge C(x', \bar{y}', z''') \wedge \mathrm{Err}_{S^{\exists}}(z, z', z'', z''')
\end{aligned}
$$

$$
\begin{aligned}
Q_{S_1^{\forall}} = \exists\, x\, x_1\, \bar{y}\, \bar{y}'\, \bar{y}''\, z\, z'\, z''\, z''' \\
S_1^{\forall}(x, x_1) \wedge \bigvee_{1 \le i \le n} Q_{\mathrm{adj},i}(\bar{y}, \bar{y}') \wedge \bigvee_{1 \le i \le n} Q_{\mathrm{adj},i}(\bar{y}', \bar{y}'') \wedge \\
C(x, \bar{y}, z) \wedge C(x, \bar{y}', z') \wedge C(x, \bar{y}'', z'') \wedge C(x_1, \bar{y}', z''') \wedge \mathrm{Err}_{S_1^{\forall}}(z, z', z'', z''')
\end{aligned}
$$

$$Q_{S_2^\forall} = \exists\, x\, x_2\, \bar{y}\, \bar{y}'\, \bar{y}''\, z\, z'\, z''\, z'''$$
$$S_2^\forall(x, x_2) \,\wedge\, \bigvee_{1 \le i \le n} Q_{\mathrm{adj}, i}(\bar{y}, \bar{y}') \,\wedge\, \bigvee_{1 \le i \le n} Q_{\mathrm{adj}, i}(\bar{y}', \bar{y}'') \,\wedge$$
$$C(x, \bar{y}, z) \,\wedge\, C(x, \bar{y}', z') \,\wedge\, C(x, \bar{y}'', z'') \,\wedge\, C(x_2, \bar{y}', z''') \,\wedge\, \mathrm{Err}_{S_2^\forall}(z, z', z'', z'''),$$

where $\mathrm{Err}_{S^\exists}$, $\mathrm{Err}_{S_1^\forall}$, and $\mathrm{Err}_{S_2^\forall}$ are defined exactly as in the proof of Theorem 4.5.

In addition, we check whether the Turing machine M reaches the rejecting state q_{rej} along some path in its computation tree. This is done with the CQ

$$Q_{\mathrm{rej}} = \exists\, x\, \bar{y}\, z\, C(x, \bar{y}, z) \,\wedge\, V_{\mathrm{rej}}(z),$$

where V_{rej} is the visible relation that contains all cell values of the form (a, q_{rej}), for some $a \in \Sigma$.

Let Q be the disjunction of all previous UCQs, and let $\mathcal{V}$ be the instance that captures the intended semantics of the visible relations Zero, One, Bit, V, Err_C, $\mathrm{Err}_{I,\mathrm{first}}$, $\mathrm{Err}_{I,\mathrm{last}}$, $\mathrm{Err}_{I,\mathrm{adj}}$, $\mathrm{Err}_{C,\mathrm{adj}}$, $\mathrm{Err}_{S^\exists}$, $\mathrm{Err}_{S_1^\forall}$, and $\mathrm{Err}_{S_2^\forall}$. We can argue as in the proof of Theorem 4.5 that M has a successful computation tree if and only if $\mathrm{PQI}(Q, \Sigma, S, \mathcal{V}) = \mathrm{false}$.   □

D UNDECIDABILITY OF SCHEMA-BASED NEGATIVE IMPLICATION FOR CQ VIEW DEFINITIONS: PROOF OF THEOREM 6.2

Recall the statement of Theorem 6.2:

The ∃NQI problem under background knowledge given as CQ-view definitions is undecidable.

This appendix will be devoted to the proof of the theorem.

As in earlier undecidability results, such as Theorem 4.16, we will give the proof for the unrestricted version of the problem, which asserts the existence of an instance with an NQI, finite or infinite.

We give a reduction from a tiling problem that is specified by a set of tiles T, an initial tile $t_\perp \in T$, and horizontal and vertical constraints $H, V \subseteq T \times T$. To match the unrestricted version of ∃NQI, we will deal with the infinite tiling variant, thus considering the problem of tiling the infinite grid $\mathbb{N} \times \mathbb{N}$.

As before, we will have visible relations E_H and E_V representing the horizontal and vertical edges of the grid. Recall that every visible relation must be associated with a CQ-view definition on a subset of hidden relations. In particular, for the relations E_H, E_V, it is sufficient to introduce hidden copies E_H', E_V' and enforce the trivial dependencies:

$$E_H(x, y) \iff E_H'(x, y)$$
$$E_V(x, y) \iff E_V'(x, y).$$

Note that these dependencies serve only to satisfy the requirement that every visible relation has an associated view definition on Σ. They play no further role in the reduction.

Similarly, each node of the grid has to be associated with a tile in T, and this will be represented by some visible unary relations U_t, together with the corresponding hidden copies U_t'. We have associated sentences in the background theory: $U_t(x) \iff U_t'(x)$, for all $t \in T$.

As in earlier undecidability results, such as Theorem 6.1, the first goal is to ensure that for each node, there exists at most one predecessor and at most one successor for the relations E_H and E_V. We explain how to ensure this for the successor case and the relation E_H, but similar constructions work for the other cases. We introduce a hidden relation HorFuncChallenge of arity 4 and a visible relation ErrHorFun of arity 3 with the associated CQ-view definition

$$\mathrm{ErrHorFun}(x, y, x') \iff \mathrm{HorFuncChallenge}(x, y, x', y).$$

Our query Q will contain as a subquery the following UCQ:

$$Q_{\mathsf{HorFuncChallenge}} \quad = \quad \big(\exists\, x\, y\, y' \, \mathsf{ErrHorFun}(x, y, y') \big) \vee$$
$$\big(\exists\, x\, y\, y' \, \mathsf{HorFuncChallenge}(x, y, x, y') \wedge E_H(x, y) \wedge E_H(x, y') \big).$$

We explain how having an NQI for the subquery $Q_{\mathsf{HorFuncChallenge}}$ on a realizable visible instance is equivalent to the visible instance having the properties that ErrHorFun is empty and every element has at most one successor in the relation E_H.

Suppose that we have $\mathbf{S}_v$-instance $\mathcal{V}$ such that $\mathsf{NQI}(Q_{\mathsf{HorFuncChallenge}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. Note that any $\mathcal{V}$ for these constraints is visible, since at this point we just have trivial "renaming" constraints.

The visible relation ErrHorFun must be empty in $\mathcal{V}$, as otherwise the query $Q_{\mathsf{HorFuncChallenge}}$ would be satisfied in every full instance that agrees with $\mathcal{V}$ on the visible part. From the fact that ErrHorFun is empty in $\mathcal{V}$ and the view definition of ErrHorFun, we conclude that every full instance that satisfies the background theory and agrees with $\mathcal{V}$ does not contain a fact of the form $\mathsf{HorFuncChallenge}(x, y, x', y)$. Now, suppose, by way of contradiction, that there is an element x with two distinct E_H-successors y and y'. We can construct a full instance that extends $\mathcal{V}$ with the single fact $\mathsf{HorFuncChallenge}(x, y, x, y')$. This full instance satisfies all of the sentences in Σ and also the query $Q_{\mathsf{HorFuncChallenge}}$, thus contradicting that $\mathsf{NQI}(Q_{\mathsf{HorFuncChallenge}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

For the converse direction, we consider a visible instance $\mathcal{V}$ in which the relation E_H is a function and the relation ErrHorFun is empty. We claim that $\mathsf{NQI}(Q_{\mathsf{HorFuncChallenge}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. Consider an arbitrary full instance $\mathcal{F}$ that agrees with $\mathcal{V}$ on the visible part and satisfies the sentences in Σ, and suppose by way of contradiction that $Q_{\mathsf{HorFuncChallenge}}$ holds on $\mathcal{F}$. Then, $\mathcal{F}$ would contain the following facts, for a triple of nodes x, y, y': $\mathsf{HorFuncChallenge}(x, y, x, y')$, $E_H(x, y)$, $E_V(x, y')$. However, $\mathcal{F}$ cannot contain the fact $\mathsf{HorFuncChallenge}(x, y, x', y)$, as otherwise this would imply the presence of the visible fact $\mathsf{ErrHorFun}(x, y, x')$. From this, we conclude that $y \neq y'$, which contradicts the functionality of E_H.

Very similar constructions and arguments can be used to enforce single successors in E_V, single predecessors in E_H and E_V, as well as confluence of E_H and E_V.

We now explain how we enforce the existential properties of the grid, such as E_H being non-empty (and similarly for E_V). We introduce two nullary relations HorEmptyError and HorEmptyHiddenError, where the former is visible and the latter is hidden, and we constrain them via the CQ-view definition

$$\mathsf{HorEmptyError} \quad \Longleftrightarrow \quad \exists\, x\, y\, \big(E_H(x, y) \wedge \mathsf{HorEmptyHiddenError} \big).$$

We add as a subquery of our query the following UCQ:

$$Q_{\mathsf{HorEmptyError}} \quad = \quad \mathsf{HorEmptyError} \vee \mathsf{HorEmptyHiddenError}.$$

In the following, we show how this enforces non-emptiness of E_H, assuming that HorEmptyError is empty.

Suppose that $\mathcal{V}$ is an $\mathbf{S}_v$-instance such that $\mathsf{NQI}(Q_{\mathsf{HorEmptyError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. We show that in this case, the relation E_H is non-empty. First, note that the fact HorEmptyError must not appear in $\mathcal{V}$, since otherwise all full instances extending $\mathcal{V}$ would satisfy $Q_{\mathsf{HorEmptyError}}$ (as $\mathcal{V}$ is realizable, there is at least one such full instance). If E_H were empty, we could set HorEmptyHiddenError to non-empty and thus get a contradiction of $\mathsf{NQI}(Q_{\mathsf{HorEmptyError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

For the converse direction, we consider a visible instance $\mathcal{V}$ in which the relation E_H is non-empty and HorEmptyError is empty. In any full instance that agrees with $\mathcal{V}$ on the visible part, HorEmptyHiddenError must agree with HorEmptyError and hence must be empty. This implies that the query $Q_{\mathsf{HorEmptyError}}$ is violated, when $\mathsf{NQI}(Q_{\mathsf{HorEmptyError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

Besides requiring that E_H and E_V are non-empty, we must also guarantee that for every pair $(x, y) \in E_H$ (respectively, $(x, y) \in E_V$), there is a pair $(y, z) \in E_V$ (respectively, $(y, z) \in E_H$). Note that once we have guaranteed this, functionality and confluence will ensure that E_H and E_V correctly encode the horizontal and vertical edges of the grid. We explain how to enforce that every pair $(x, y) \in E_H$ has a successor pair $(y, z) \in E_V$—a similar construction can be given for the symmetric property. We add to our schema another visible relation HorSuccError of arity 0 and a hidden relation HorSuccHiddenError of arity 1. The associated CQ-view definition is

$$\text{HorSuccError} \leftrightarrow \exists x \, y \, z \, E_H(x, y) \wedge \text{HorSuccHiddenError}(y) \wedge E_V(y, z).$$

Moreover, we add as a subquery of our query the following UCQ:

$$Q_{\text{HorSuccError}} = \text{HorSuccError} \vee \left(\exists x \, y \, E_H(x, y) \wedge \text{HorSuccHiddenError}(y) \right).$$

We show how this enforces the desired property.

Suppose that there is a visible instance $\mathcal{V}$ such that $\text{NQI}(Q_{\text{HorSuccError}}, \Sigma, S, \mathcal{V}) = \text{true}$. First, observe that the visible relation HorSuccError must be empty, as otherwise all extensions of $\mathcal{V}$ would satisfy $Q_{\text{HorSuccError}}$. Now, suppose, by way of contradiction, that there is a pair $(x, y) \in E_H$ that has no successor pair $(y, z) \in E_V$. In this case, we can construct a full instance that extends $\mathcal{V}$ with the hidden fact HorLabelHiddenError(y). This full instance has $\mathcal{V}$ as visible part and satisfies the sentences in the background theory and the query $Q_{\text{HorSuccError}}$. As this contradicts the hypothesis $\text{NQI}(Q_{\text{HorSuccError}}, \Sigma, S, \mathcal{V}) = \text{true}$, we conclude that for every pair $(x, y) \in E_H$, there is a successor pair $(y, z) \in E_V$.

Conversely, consider a visible instance $\mathcal{V}$ that represents a correct encoding of the infinite grid and where the visible relation HorSuccError is empty. In any full instance that agrees with $\mathcal{V}$ on the visible part, HorSuccError must be the same as $\exists x \, y \, z \, E_H(x, y) \wedge \text{HorSuccHiddenError}(y) \wedge E_V(y, z)$. In particular, because every node has both a successor in E_H and a successor in E_V, this implies that the hidden relation HorSuccHiddenError cannot contain the node y, for any pair $(x, y) \in E_H$. Hence, the query $Q_{\text{HorSuccError}}$ is necessarily violated, and this proves that $\text{NQI}(Q_{\text{HorSuccError}}, \Sigma, S, \mathcal{V}) = \text{true}$.

Now that we have enforced a grid-like structure on the relations E_H and E_V, we consider the relations U_t that encode a candidate tiling function. Using similar techniques, we can ensure that every node of the grid has an associated tile. More precisely, we enforce that, for every pair $(x, y) \in E_H$, the element x must also appear in U_t, for some tile $t \in T$. We add a visible relation HorLabelError$_t$ of arity 0 for each tile $t \in T$ and a hidden relation HorLabelHiddenError of arity 1. The associated CQ-view definitions are of the form

$$\text{HorLabelError}_t \iff \exists x \, y \, E_H(x, y) \wedge \text{HorLabelHiddenError}(x) \wedge U_t(x).$$

We add as a subquery of our query the following UCQ:

$$Q_{\text{HorLabelError}} = \bigvee_{t \in T} \exists x \, y \, \left(\text{HorLabelError}_t(x, y) \right) \vee \left(E_H(x, y) \wedge \text{HorLabelHiddenError}(x) \right).$$

We prove that the above definitions enforce that all nodes that appear in the first column of the relation E_H have at least one associated tile.

Consider a visible instance $\mathcal{V}$ such that $\text{NQI}(Q_{\text{HorLabelError}}, \Sigma, S, \mathcal{V}) = \text{true}$. For each tile t, the visible relation HorLabelError$_t$ must be empty, as otherwise all extensions of $\mathcal{V}$ would satisfy $Q_{\text{HorLabelError}}$. Suppose, by way of contradiction, that there is a node x that appears in the first column of the visible relation E_H but does not appear in any relation U_t, with $t \in T$. We can construct a full instance where the relation HorLabelHiddenError contains the element x. This instance would then satisfy the query $Q_{\text{HorLabelError}}$, thus contradicting that $\text{NQI}(Q_{\text{HorLabelError}}, \Sigma, S, \mathcal{V}) = \text{true}$.

For the converse, consider a visible instance $\mathcal{V}$ in which the relation E_H is non-empty (as enforced in the previous steps) and, for all pairs $(x, y) \in E_H$, there is a tile $t \in T$ such that $x \in U_t$. Furthermore, assume that all relations HorLabelError_t, with $t \in T$, in this visible instance are empty. In every full instance that agrees with $\mathcal{V}$ and satisfies the background theory, HorLabelError_t must be the same as $\exists x\, y\, \text{HorLabelHiddenError}(x) \wedge E_H(x, y) \wedge U_t(x)$. In particular, because every node is associated with some tile, this implies that the hidden relation $\text{HorLabelHiddenError}$ cannot contain the node x, for any pair $(x, y) \in E_H$. Hence, the query $Q_{\text{HorLabelError}}$ is necessarily violated, and this proves that $\text{NQI}(Q_{\text{HorLabelError}}, \Sigma, S, \mathcal{V}) = \text{true}$.

We also need to guarantee that each node has at most one associated tile. This property can be easily enforced by the subquery

$$Q_{\text{TwoLabelsError}} \ = \ \bigvee_{t \neq t'} \exists x\, U_t(x) \wedge U_{t'}(x).$$

Finally, we enforce that the encoded tiling function respects the horizontal and vertical constraints using the following UCQ:

$$
\begin{aligned}
Q_{\text{ConstraintError}} \ = \ &\bigvee_{(t, t') \notin H} \big(\exists x\, y\, E_H(x, y) \wedge U_t(x) \wedge U_{t'}(y) \big) \ \vee \\
&\bigvee_{(t, t') \notin V} \big(\exists x\, y\, E_V(x, y) \wedge U_t(x) \wedge U_{t'}(y) \big).
\end{aligned}
$$

Summing up, if we let Q be the disjunction of all previous queries Q_e for subqueries above corresponding to various grid errors e. The previous arguments show that $\exists \text{NQI}(Q, \Sigma, S) = \text{true}$ if and only if there exists a valid tiling of the infinite grid $\mathbb{N} \times \mathbb{N}$. Note that, by definition, $\text{NQI}(Q, \Sigma, S, \mathcal{V}) = \text{true}$ holds for a UCQ $Q = \bigcup_e Q_e$ if and only if $\text{NQI}(Q_e, \Sigma, S, \mathcal{V}) = \text{true}$ holds for all Q_e, and this property can be directly transferred to the schema-level problem $\exists \text{NQI}$.

This completes the proof of the theorem.

ACKNOWLEDGMENTS

REFERENCES

[1] S. Abiteboul and O. Duschka. 1998. Complexity of answering queries using materialized views. In *Proceedings of PODS'98*.

[2] S. Abiteboul, R. Hull, and V. Vianu. 1995. *Foundations of Databases*. Addison-Wesley.

[3] Antoine Amarilli and Michael Benedikt. 2018. When Can We Answer Queries Using Result-Bounded Interfaces? Retrieved March 21, 2021 from https://arxiv.org/pdf/1706.07936.pdf.

[4] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. 2009. Extending decidable cases for rules with existential variables. In *Proceedings of IJCAI'09*.

[5] Vince Bárány, Balder Ten Cate, and Martin Otto. 2012. Queries with guarded negation. In *Proceedings of VLDB'12*.

[6] Vince Bárány, Balder Ten Cate, and Luc Segoufin. 2011. Guarded negation. In *Proceedings of ICALP'11*.

[7] Vince Bárány, Balder Ten Cate, and Luc Segoufin. 2015. Guarded negation. *J. ACM* 62, 3 (2015), Article 22.

[8] Vince Bárány, George Gottlob, and Martin Otto. 2010. Querying the guarded fragment. In *Proceedings of LICS'10*.

[9] C. Beeri and M. Y. Vardi. 1981. The implication problem for data dependencies. In *Proceedings of ICALP'81*.

[10] Michael Benedikt, Pierre Bourhis, Louis Jachiet, and Michaël Thomazo. 2019. Reasoning about disclosure in data integration in the presence of source constraint. In *Proceedings of IJCAI'19*. Long version available at arxiv.org/pdf/1906.00624.pdf.

[11] Michael Benedikt, Pierre Bourhis, Louis Jachiet, and Efthymia Tsamoura. 2020. Balancing expressiveness and inexpressiveness in view design. In *Proceedings of KR'20*.

[12] Michael Benedikt, Pierre Bourhis, Gabriele Puppis, and Balder ten Cate. 2016. Querying visible and invisible information. In *Proceedings of LICS'16*.

[13] Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. 2017. Characterizing definability in decidable fixpoint logics. In *Proceedings of ICALP'17*.

[14] Michael Benedikt, Thomas Colcombet, Balder ten Cate, and Michael Vanden Boom. 2015. The complexity of boundedness for guarded logics. In *Proceedings of LICS '15*.

[15] Michael Benedikt, Bernardo Cuenca Grau, and Egor V. Kostylev. 2017. Source information disclosure in ontology-based data integration. In *Proceedings of AAAI'17*.

[16] Michael Benedikt, Bernardo Cuenca Grau, and Egor V. Kostylev. 2018. Logical foundations of information disclosure in ontology-based data integration. *Artif. Intell.* 262 (2018), 52–95.

[17] Michael Benedikt, Boris Motik, and Efthymia Tsamoura. 2018. Goal-driven query answering for existential rules with equality. In *Proceedings of AAAI'18*.

[18] Michael Benedikt, Julien Leblay, and Efthymia Tsamoura. 2016. *Generating Plans from Proofs: The Interpolation-Based Approach to Query Reformulation*. Morgan Claypool.

[19] Michael Benedikt, Balder ten Cate, and Efi Tsamoura. 2016. Generating plans from proofs. *ACM TODS* 40, 4 (2016), Article 22.

[20] Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. 2014. Effective interpolation and preservation in guarded logics. In *Proceedings of CSL-LICS'14*.

[21] E. W. Beth. 1953. On Padoa's method in the theory of definitions. *Indagationes Mathematicae* 15 (1953), 330–339.

[22] Pierre Bourhis, Marco Manna, Michael Morak, and Andreas Pieris. 2016. Guarded-based disjunctive tuple-generating dependencies. *ACM Trans. Database Syst.* 41, 4 (2016), Article 27, 45 pages.

[23] Andrea Calì, George Gottlob, and Michael Kifer. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *JAIR* 48 (2013), 115–174.

[24] Marco A. Casanova, Ronald Fagin, and Christos Papadimitriou. 1984. Inclusion dependencies and their interaction with functional dependencies. *JCSS* 28, 1 (1984), 29–59.

[25] C. C. Chang and H. J. Keisler. 1990. *Model Theory*. North-Holland.

[26] R. Chirkova and T. Yu. 2014. Obtaining information about queries behind views and dependencies. arXiv:1403.5199

[27] Alin Deutsch, Alan Nash, and Jeff Remmel. 2008. The chase revisited. In *Proceedings of PODS'08*.

[28] Ronald Fagin, Phokion G. Kolaitis, Renee J. Miller, and Lucian Popa. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci.* 336, 1 (2005), 89–124.

[29] W. Fan and F. Geerts. 2010. Capturing missing tuples and missing values. In *Proceedings of PODS'10*.

[30] W. Fan and F. Geerts. 2010. Relative information completeness. *ACM TODS* 35, 4 (2010), 27.

[31] E. Franconi, Y. Ibáñez-García, and I. Seylan. 2011. Query answering with DBoxes is hard. *ENTCS* 278 (2011), 71–84.

[32] Tomasz Gogacz and Jerzy Marcinkowski. 2014. All-instances termination of chase is undecidable. In *Proceedings of ICALP'14*.

[33] Tomasz Gogacz and Jerzy Marcinkowski. 2015. The hunt for a red spider: Conjunctive query determinacy is undecidable. In *Proceedings of LICS'15*.

[34] George Gottlob and Christos Papadimitriou. 2003. On the complexity of single-rule datalog queries. *Inf. Comp.* 183 (2003), 104–122.

[35] B. Cuenca Grau, I. Horrocks, M. Krötzsch, C. Kupke, D. Magka, B. Motik, and Z. Wang. 2013. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Int. Res.* 47, 1 (2013), 741–808.

[36] M. Guarnieri and D. A. Basin. 2014. Optimal security-aware query processing. *PVLDB* 7, 12 (2014), 1307–1318.

[37] B. Konev, C. Lutz, D. Walther, and F. Wolter. 2013. Model-theoretic inseparability and modularity of description logic ontologies. *Artif. Intell.* 203 (2013), 66–103.

[38] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. 2012. Query-based data pricing. In *Proceedings of PODS'12*.

[39] C. Lutz, I. Seylan, and F. Wolter. 2012. Mixing open and closed world assumption in ontology-based data access: Non-uniform data complexity. In *Proceedings of the 2012 International Workshop on Description Logics*. http://ceur-ws.org/Vol-846/paper_17.pdf.

[40] C. Lutz, I. Seylan, and F. Wolter. 2013. Ontology-based data access with closed predicates is inherently intractable (sometimes). In *Proceedings of IJCAI'13*.

[41] Carsten Lutz, Inanc Seylan, and Frank Wolter. 2015. Ontology-mediated queries with closed predicates. In *Proceedings of IJCAI'15*.

[42] C. Lutz and F. Wolter. 2007. Conservative extensions in the lightweight description logic EL. In *Proceedings of CADE'07*.

[43] Bruno Marnette and Floris Geerts. 2010. Static analysis of schema-mappings ensuring oblivious termination. In *Proceedings of ICDT'10*.

[44] G. Miklau and D. Suciu. 2007. A formal analysis of information disclosure in data exchange. *JCSS* 73, 3 (2007), 507–534.

[45] Alan Nash and Alin Deutsch. 2006. Privacy in GLAV information integration. In *Proceedings of ICDT'06*.

[46] A. Nash, L. Segoufin, and V. Vianu. 2010. Views and queries: Determinacy and rewriting. *ACM TODS* 35, 3 (2010), Article 21.
[47] Nhung Ngo, Magdalena Ortiz, and Mantas Simkus. 2016. Closed predicates in description logics: Results on combined complexity. In *Proceedings of KR'16*.
[48] Adrian Onet. 2013. The chase procedure and its applications in data exchange. In *Data Exchange Intregation and Streams*. Schloss Dagstuhl, 1–37.
[49] Oded Shmueli. 1993. Equivalence of datalog queries is undecidable. *JLAP* 15, 3 (1993), 231–241.
[50] M. Y. Vardi. 1998. Reasoning about the past with two-way automata. In *Proceedings of ICALP'98*.
[51] Z. Zhang and A. O. Mendelzon. 2005. Authorization views and conditional query containment. In *Proceedings of ICDT'05*.