Querying Visible and Invisible Information

Michael Benedikt Pierre Bourhis Balder ten Cate Gabriele Puppis

Abstract

We provide a wide-ranging study of the scenario where a subset of the relations in the schema are visible — that is, their complete contents are known — while the remaining relations are invisible. We also have integrity constraints (invariants given by logical sentences) which may relate the visible relations to the invisible ones. We want to determine which information about a query (a positive existential sentence) can be inferred from the visible instance and the constraints. We consider both positive and negative query information, that is, whether the query or its negation holds. We consider the instance-level version of the problem, where both the query and the visible instance are given, as well as 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.

1. Introduction

There are many applications scenarios where there is a collection of relations representing information of interest to a set of users, but a given user or class of users has access to only a subset of these relations. For example a data owner may restrict access to a subset of the stored relations for privacy reasons. Another example comes from information integration where the schema exposed to users contains both stored relations and "virtual relations". The virtual relations can be referenced in user queries, but are defined by logical constraints relating them to stored relations. Both of these scenarios can be subsumed by considering a schema consisting of a set of relations that must satisfy a set of *integrity constraints* (invariants specified by sentences in some logic) with only a subset of the relations visible. A basic computational problem is what questions can be answered by means of reasoning with the constraints and access to the visible relations. Can someone use the content of the visible relations along with constraints to answer a given question about the invisible relations?

We study exactly this scenario, where a set of semantically-related relations are hidden while for another set the complete contents are visible. We will consider semantic relationships specified in a variety of logical languages that are rich enough to capture complex relationships between relations, including relationships that arise in information integration, as well as common integrity constraints within a single source. The basic analysis problem will be the following: given a schema and a query Q (also given as a logical formula; for simplicity we focus on sentences), can we infer using the visible data and schema information that the result of Q is true or that the result is false.

LICS '16, July 05-08, 2016, New York, NY, USA.

http://dx.doi.org/10.1145/http://dx.doi.org/10.1145/2933575.2935306

Example 1. *Consider a medical datasource with relation* $\mathsf{Appointment}(p, a, \ldots)$ *containing patient names p, appointment ids a, and other information about the appointment, such as the name of the doctor. A dataowner makes available one projection of* Appointment *by creating a relation* $\mathsf{Patient}(p)$ *with the constraints:*

$$\forall\ p\ \mathsf{Patient}(p) \ \rightarrow\ \exists\ a\ d\ \bar{y}\ \mathsf{Appointment}(p, a, d, \bar{y})$$
$$\forall\ p\ a\ d\ \bar{y}\ \mathsf{Appointment}(p, a, d, \bar{y}) \ \rightarrow\ \mathsf{Patient}(p)\ .$$

The query $Q = \exists\ a\ \bar{y}\ \mathsf{Appointment}(\text{"}Smith\text{"}, a, \text{"}Jones\text{"}, \bar{y})$ *asking whether patient Smith made an appointment with Dr. Jones will be secure under this schema in one sense: an external user with access to* Patient *will never be sure that the query is true, in any instance. We say that there can be no Positive Query Implication for this query and schema, on any instance. But suppose we consider whether a user can infer the query to be false? On many instances, such an inference is not possible. But on an instance where the visible relation* Patient *is empty, an external user will know that the query is false. We say that there is a Negative Query Implication on the visible instance where* Patient *is empty.*

Our results. We will consider the instance-based problems: given a query and instance, can a user determine that the query is true (Positive Query Implication) or that the query is false (Negative Query Implication)? We also look at the corresponding schema-level problem: given a query and a schema, is there some instance where a query implication of one of the above types occurs?

We start by observing that the instance-level problems, both positive and negative, are decidable for a broad class of constraints. However, when we analyze the complexity of the decision problem as the size of the instance increases, we see surprisingly different behavior between the positive and negative case. For very simple constraints the negative query implication problems are well-behaved as the instance changes, namely, in polynomial time and definable within a well-behaved query language. For the same class of constraints, the corresponding positive query implication questions are hard even when the schema and query are fixed.

When we turn to the schema-level problems, even decidability is not obvious. We prove a set of "critical instance" results, showing that whenever there is an instance where information about the query can be implied, the "obvious instance" witnesses this. Thus, schema-level problems reduce to special cases of the instance-level problems. Although we use this technique to obtain decidability and complexity results both for positive and for negative query implication, the classes of constraints to which they apply are different. We give undecidability results that show that when the classes are even slightly enlarged, decidability of the existence of a schema with a query implication is lost.

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

- **Embeddings in rich decidable logics.** 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 nega-

tion fragment. This exploits powerful prior decidability results "off-the-shelf". However, to get tight complexity bounds, we also need a new analysis of the complexity of this logic.

- **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 above, we show that whenever there is some instance, this can be taken to be the "simplest possible instance". While this idea has been used before to simplify analysis of undecidability (e.g. [16]), we give a broad result that allows the use 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 databases that lie within the active domain of the visible instance).
- **Relationships between problems.** We prove reductions relating the positive and negative implication problems, the schema- and instance-level problems, and the widely-studied "certain answer problem". We use these reductions to derive tight complexity bounds.
- **Coding techniques for lower bounds.** We introduce methods for coding computation in query implication problems, yielding both undecidability and complexity lower bounds.

Related work. Two different communities have studied the problem of determining the information that can be inferred from accessing data in a subset of the relations of a schema using constraints that relate the subset to the full vocabulary.

In the database community, the focus has been on views. The schema is divided into *base tables* and *view tables*, with the latter being defined by queries (typically *conjunctive queries*, abbreviated 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 Positive Query Implication (PQI) problem in the case where the constraints consist entirely of CQ view definitions. Another subcase of PQI that has received considerable attention is the case where the constraints consist *only* of completeness assertions between the invisible and visible portions of the schema (e.g. [14]).

The PQI problem is also related to works on instance-based determinacy (see in particular the results of Howe et al. in [19]), while the Negative Query Implication (NQI) problem is studied in the view context by Mendelzon and Zhang [24], under the name of *conditional emptiness*. In both cases, the emphasis has been on view definitions rather than more general constraints which may restrict both the visible and invisible instance. In contrast, our work deals with constraint classes that can restrict the visible and invisible data in ways incomparable to view definitions. However, our techniques do extend to solve the corresponding problems for view definitions; this extension is discussed in the full version.

The PQI problem also relates to works in description logics (e.g. [15, 21]) focusing on *hybrid closed and open world query answering* (*DBoxes*), where the schema is divided into closed-world and open-world relations. The problem in this setting is to find out if a Boolean CQ holds in all instances that can add facts to the open-world relations but do not change the closed-world relations. Thus, closed and open worlds match our notions of visible and invisible relations, and the hybrid closed and open world query answering problem matches our notion of Positive Query Implication, except that in our setting we restrict to the case where the open-world/invisible relations 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 inclusion dependency from R' to R. The main distinction between our study of the PQI problem and the prior work in the description logic community concerns the classes of constraints considered. Lutz et al. [20, 21] study the complexity of this problem for the constraint languages $\mathcal{EL}$ and DL-LITE. [21] gives a criteria identifying constraints for which all queries are first-order rewritable, while [20] carries out an analysis at the level of constraints and queries. Franconi et al. [15] show co-NP-completeness for a disjunction-free description logic. Our results on the data complexity of PQI consider the same problem, but for decidable constraint languages that are more expressive and, in particular, can handle relations of arbitrary arity, rather than arity at most 2 as in [15, 20, 21].

In summary, the database and description logic communities considered the Positive Query Implication problems addressed here, but for different classes of constraints. The Negative Query Implication problems are not well-studied in the prior literature, and we do not know any work dealing with the schema-level questions (existence of an instance with a query implication). Despite this, we show (Subsection 4.2) that there is a close relation between the schema-level questions and the work of Lutz et al. [22] concerning conservativity and modularity of constraints.

For space reasons, some proofs and further applications of the techniques we develop are deferred to the full version; they can also be found in the technical report [8].

2. Definitions

We consider schemas $\mathcal{S} = \mathcal{S}_h \uplus \mathcal{S}_v$, where the partition elements $\mathcal{S}_h$ and $\mathcal{S}_v$ are finite sets of relation names (or simply, relations), each with an associated arity. 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. The *active domain* of any instance $\mathcal{I}$, denoted $\mathsf{adom}(\mathcal{I})$, is the set of values occurring within the interpretations of the relations of $\mathcal{I}$. As a suggestive notation, we will use $\mathcal{F}$ for instances over $\mathcal{S}$ and $\mathcal{V}$ for instances over $\mathcal{S}_v$. Given an instance $\mathcal{F}$ for $\mathcal{S}$, its restriction to the $\mathcal{S}_v$ relations will be referred to as its *visible part*, denoted Visible($\mathcal{F}$).

Instances will be used as inputs to the computational problems considered in this work – when used as inputs, the instances must be finite. We will also consider problems that quantify over instances – in this case the quantification can be either over finite instances or over all (finite or infinite) instances. We will deal with both possibilities, but *when we do not specify otherwise, instances are assumed to be unrestricted*.

We will consider formalisms based on first-order logic for expressing constraints and queries. Note that, in general, the evaluation of a first-order formula may depend on the underlying domain (not just on the instance and its active domain). We will always assume the formulas used in queries and constraints are domain-independent, which will allow us to evaluate queries and constraints over instances; this can be enforced, for example, by using relational algebra syntax.

We will mainly look at integrity constraints defined by *tuple-generating dependencies* (*TGD*s), which are first-order 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 containing variables and constants, and where all the

universally quantified variables $\bar{x}$ appear in $\phi(\bar{x})$. We will often omit the universal quantifiers, writing just $\phi(\bar{x}) \rightarrow \exists \bar{y}\ \rho(\bar{x}, \bar{y})$. Moreover, for all the problems considered in this work, one can take w.l.o.g. the right-hand side ρ to consist of a single atom, and we will assume this henceforth. We will consider the following classes of TGDs:

- *Linear TGDs*: those where ϕ consists of a single atom.
- *Inclusion dependencies* (IDs): where each of ϕ and ρ is a single atom having no constants and no repeated variables. In Example 1, both the constraints were IDs.
- *Frontier-guarded TGDs* (FGTGDs) [3]: these are TGDs where one of the conjuncts of ϕ is an atom that includes all the universally quantified variables occurring in ρ.
- *Connected TGDs*: TGDs such that the *co-occurrence graph* of ϕ is connected. The nodes of this graph are the universally quantified variables $\bar{x}$, and they 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. Many of our results apply to even richer constraints, which allow disjunction and (in the second case below) negation.

- *Disjunctive FGTGDs*: these are sentences of the form $\forall \bar{x}\ \phi(\bar{x}) \rightarrow \exists \bar{y}\ \bigvee_i \rho_i(\bar{x}, \bar{y})$, where, for each i, ρ_i is a conjunction of atoms and there is an atom in ϕ that includes all the variables x_j occurring in ρ_i.
- The *Guarded negation fragment* (GNFO): this is the fragment of first-order logic in which every negation is conjoined with an atom that contains all the free variables of the negated formula. Formally, GNFO sentences are built up according to the following grammar:

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

  where R is either a relation symbol or the equality relation $x = y$, and the t_i's are either variables or constants.

The reader only needs to know a few facts about GNFO. It is very expressive, so results about GNFO constraints immediately apply to many classes of constraints mentioned above. GNFO contains all positive existential formulas, is closed under Boolean combinations of sentences, and subsumes Disjunctive FGTGDs (up to equivalence), and hence also subsumes FGTGDs and IDs. Moreover, GNFO is "tame":

Theorem 1 ([7])**.** *Satisfiability of GNFO sentences can be tested effectively, and is* 2Exp*-complete. Furthermore, every satisfiable sentence has a finite satisfying model.*

Finally, we will also consider *equality-generating dependencies* (EGDs), of the form $\forall \bar{x}\ \phi(\bar{x}) \rightarrow x_i = x_j$, where ϕ is a conjunction of atoms and x_i, x_j are variables or constants. EGDs generalize well-known relational database constraints, such as functional dependencies and key constraints.

To express queries we use *conjunctive queries* (CQs), i.e. 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), as well as disjunctions of CQs, abbreviated UCQs. *Boolean UCQs (abbreviated BUCQs)* are UCQs without free variables and Boolean CQs (BCQs) are CQs without free variables. *Although for brevity we focus on Boolean queries in this work, the techniques extend to the non-Boolean case, as explained in [8].* Every CQ Q is associated with a *canonical database* $\mathsf{CanonDB}(Q)$, whose domain consists of variables and constants of Q and whose facts are the atoms of Q.

We remark that in our constraint and query languages above, with the exception of IDs, *constants are allowed by default*. We will mention explicitly when we want to restrict to formulas without constants. We will further assume that we have associated with each value a corresponding constant, and we will identify constants with their values. Thus distinct constants will always be forced to denote distinct domain elements – this is often called the "unique name assumption" [2]. There are several problems that we study where the presence of such constants adds significant complications. In contrast, it is easy to show that the presence of constants without the unique name assumption will never make any difference in any of our results. None of our lower bounds rely on the presence of constants, except when explicitly stated otherwise.

We now define the crucial problems of *Positive Query Implication* (PQI) and *Negative Query Implication* (NQI):

Definition 2. *Let Q be a BUCQ over schema $\mathcal{S}$, C a set of constraints over $\mathcal{S}$, and $\mathcal{V}$ an instance over a visible schema $\mathcal{S}_v \subseteq \mathcal{S}$.*

- $\mathrm{PQI}(Q, C, \mathcal{S}, \mathcal{V}) = \mathsf{true}$ *if for every instance $\mathcal{F}$ satisfying C, if $\mathcal{V} = \mathrm{Visible}(\mathcal{F})$ then $Q(\mathcal{F}) = \mathsf{true}$.*
- $\mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V}) = \mathsf{true}$ *if for every instance $\mathcal{F}$ satisfying C, if $\mathcal{V} = \mathrm{Visible}(\mathcal{F})$ then $Q(\mathcal{F}) = \mathsf{false}$.*

We say that an $\mathcal{S}_v$-instance $\mathcal{V}$ is *realizable* w.r.t. C if there is an $\mathcal{S}$-instance $\mathcal{F}$ satisfying C such that $\mathcal{V} = \mathrm{Visible}(\mathcal{F})$. If $\mathcal{V}$ is not realizable w.r.t. C, then, trivially, $\mathrm{PQI}(Q, C, \mathcal{S}, \mathcal{V}) = \mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V}) = \mathsf{true}$. In practice, realizable instances are the only $\mathcal{S}_v$-instances we should ever encounter. For simplicity we state our instance-level results for the PQI and NQI problems that take as input an arbitrary instance of $\mathcal{S}_v$. But since our lower bound arguments will only involve realizable instances, an alternative definition that assumes realizable inputs yields the same complexity bounds.

We also observe that the above problems quantify over all instances, finite and infinite in line with our default assumption. One can also consider versions of these problems where the quantification is over finite infinite instances. We will show that *taking the quantification over finite instances will not impact our results*. That is, we will show that the finite and unrestricted versions of PQI and NQI coincide for a large class of arguments $Q, C, \mathcal{S}, \mathcal{V}$. We express this by saying that $\mathrm{PQI}(Q, C, \mathcal{S}, \mathcal{V})$ is *finitely controllable*, and similarly for NQI. Finite controllability will also allow us to make use of infinite instances freely in our proofs.

We will also be interested in studying the behavior of the above problems when $Q, \mathcal{S}, C$ are fixed, i.e. analysing how the computation time varies in the size of $\mathcal{V}$ only. We refer to this as the *data complexity* of the PQI and NQI problems.

The PQI problem contrasts with the usual *Open-World Query Answering* or *Certain Answer* problem, denoted here $\mathrm{OWQ}(Q, C, \mathcal{F})$, which is studied extensively in databases and description logics. The latter problem takes as input a Boolean query Q, a finite instance $\mathcal{F}$, and a set of constraints C, and returns true iff the query holds in any finite instance $\mathcal{F}'$ containing all facts of $\mathcal{F}$. The difference of this problem from PQI (and NQI) is that in PQI 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 2. *Consider a schema with inclusion dependencies $V_1(x) \rightarrow \exists y\ H(x, y)$ and $H(x, y) \rightarrow V_2(y)$, where V_1, V_2 are visible and H is not. Let Q be the CQ $\exists x\ H(x, x)$ and $\mathcal{V}$ the instance consisting only of facts $V_1(a), V_2(a)$. Here we have a Positive Query Implication: the visible fact $V_1(a)$ and the first constraint imply the existence of a fact $H(a, c)$, for some value c, but the second constraint and the content of V_2 imply $c = a$, whence $\exists x\ H(x, x)$. In contrast, one can see that Q is not certain in the usual sense, where V_1 and V_2 can be freely extended with additional facts.*

Our schema-level problems concern determining if there is a *realizable* instance that admits a query implication:

Definition 3. *For Q a BCQ over schema $\mathcal{S}$, and C a set of constraints over $\mathcal{S}$, we let:*

- $\exists\text{PQI}(Q, C, \mathcal{S})$ = true *if there exists a realizable $\mathcal{S}_v$-instance $\mathcal{V}$ such that* $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$ = true*;*
- $\exists\text{NQI}(Q, C, \mathcal{S})$ = true *if there exists a realizable $\mathcal{S}_v$-instance $\mathcal{V}$ such that* $\text{NQI}(Q, C, \mathcal{S}, \mathcal{V})$ = true.

In Example 1, $\exists\text{PQI}(Q, C, \mathcal{S})$ = false, because the user could never infer that the query was true. But $\exists\text{NQI}(Q, C, \mathcal{S})$ = true, due to the empty instance. Note that the schema-level problems quantify over instances twice. As before, we say the schema-level problems are finitely controllable if both quantifications over instances can be simultaneously replaced with quantifications over finite instances.

3. Positive Query Implication

3.1 Instance-level problem

We begin by showing that the instance-level problem PQI is decidable for constraints in the rich logic GNFO. The key observation is that PQI can be expressed as a GNFO unsatisfiability problem.

Theorem 4. $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$ *is in* 2Exp*, as Q ranges over BUCQs and C over GNFO constraints. Furthermore, for such constraints the problem is finitely controllable.*

Proof. $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$ = true translates to unsatisfiability of

$$\neg Q \ \wedge \ C \ \wedge \bigwedge_{R \in \mathcal{S}_v} \Big(\bigwedge_{R(\bar{a}) \in \mathcal{V}} R(\bar{a}) \ \wedge \ \forall \bar{x} \, (R(\bar{x}) \rightarrow \bigvee_{R(\bar{a}) \in \mathcal{V}} \bar{x} = \bar{a}) \Big) .$$

If the constraints are in GNFO, then the formula above is also in GNFO. Finite controllability and the 2Exp bound on GNFO satifiability (Theorem 1) imply the conclusion. □

Above we are using decidability of satisfiability of GNFO as a black-box. The decision procedure for GNFO works by translating a formula into a tree automaton, which is then tested for non-emptiness. In the full version, we do a finer analysis of the translation, using a method developed in [9]. The analysis shows that for each fixed formula and schema, we can generate a two-way alternating tree automaton of size polynomial in the instance. From this and standard results in automata theory, it follows that the data complexity of PQI is only singly-exponential.

Theorem 5. *Given a BUCQ Q and set C of GNFO constraints over schema $\mathcal{S}$, the data complexity of* $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$*, as $\mathcal{V}$ varies over instances, is in* Exp.

The above data complexity bound is tight even for inclusion dependencies:

Theorem 6. *There is a BCQ Q and a set C of IDs over a schema $\mathcal{S}$ for which the problem* $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$ *is* Exp*-hard in data complexity.*

Note that this contrasts sharply with the case of OWQ for GNFO constraints which has data complexity in co-NP [6]. The proof (given in the full version, as most of our lower bounds are) proceeds by showing that a universal machine for alternating PSpace can be constructed by fixing appropriate $Q, C, \mathcal{S}$ in a PQI problem. The input of the machine is represented in the visible instance $\mathcal{V}$, and an encoding of the computation is induced in some hidden relations by the constraints. Encodings of rejecting computations and badly-formed encodings are also allowed, but these particular cases can be detected by the query. In particular, we have that $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$ = false iff there exists an encoding of a successful computation of the alternating PSpace Turing machine. We also remark that the proof requires a schema with arity above 2. In fact, if we move up from IDs to linear TGDs, we can show Exp-hardness even for arity 2.

We conclude the analysis of the instance-level PQI problem by establishing that the 2Exp combined complexity upper bound is tight even for IDs.

Theorem 7. $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$ *is* 2Exp*-hard for combined complexity, as Q ranges over BCQs and C over sets of IDs.*

The proof of this lower bound is similar to that of Theorem 6; we reduce the acceptance problem for an alternating ExpSpace Turing machine M to the negation of $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$. The technical difficulty here is to encode a tape of exponential size, which cannot be done succinctly using a visible instance. We overcome this problem by representing the positions of the tape as leaves of a full binary tree of linear height (of course this makes the schema and the constraints depend on the input of the machine).

3.2 Schema-level problem

We now turn to the schema-level problem ∃PQI. Let $\mathcal{V}_{\{a\}}$ be the instance for the visible part of a schema $\mathcal{S}$ whose active 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 constraint languages, whenever $\exists\text{PQI}(Q, C, \mathcal{S})$ = 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 which has been applied previously in undecidability results. E.g., Gogacz and Marcinkowski [16] refer to this instance as a "well of positivity". The following result shows that for ∃PQI, this technique works with TGDs and EGDs *without constants.*

Theorem 8. *For every BUCQ Q without constants and every set C of TGDs and EGDs without constants,* $\exists\text{PQI}(Q, C, \mathcal{S})$ = true *iff* $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}_{\{a\}})$ = true*. In particular,* $\exists\text{PQI}(Q, C, \mathcal{S})$ = true *iff there is a finite realizable instance $\mathcal{V}$ such that* $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V})$ = true.

We will prove the theorem first for constraints consisting only of TGDs without constants; then we will show how to generalize it in the presence of EGDs without constants.

First of all, we note that, by introducing additional invisible relations, we can assume w.l.o.g. that all TGDs have exactly one atom in the right-hand side. Next, we introduce a variant of the "classical" chase procedure [2] that returns a collection of instances (not necessarily finite); the extension is along the lines of the "Disjunctive Chase" of [13]. As in the classical chase, the procedure receives as input a relational schema $\mathcal{S}$, some constraints C, and an initial instance $\mathcal{F}_0$ for the schema $\mathcal{S}$, which does not need to satisfy the constraints in C. The procedure chases the constraints starting from the instance $\mathcal{F}_0$, guaranteeing at the same time that the visible relations of the constructed instances agree with $\mathcal{F}_0$. This variant of the chase will be used to prove Theorem 8, as well as other results related to the ∃NQI problem.

Formally, the procedure builds a tree-shaped collection of instances starting from the singleton consisting of the input $\mathcal{S}$-instance $\mathcal{F}_0$. It extends the collection by repeatedly applying the following process. 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 C, where $\bar{z}$ is a sequence of (possibly repeated) variables chosen among $\bar{x}_1, \ldots, \bar{x}_m, \bar{y}$, and a homomorphism f that maps $R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)$ to some facts in K. The procedure constructs a new instance from K by adding the fact $S(f'(\bar{z}))$, where f' is an extension of f that injectively maps the existentially quantified variables in $\bar{y}$ to some fresh null values. When the relation S is visible, the procedure replaces the instance $K' = K \cup \{S(f'(\bar{z}))\}$ with copies of itself of the form $g(K')$ such that $\text{Visible}(g(K')) = \text{Visible}(\mathcal{F}_0)$, for all possible homomorphisms g from the variables $\bar{z}$ to values in the active domain of $\text{Visible}(\mathcal{F}_0)$; in particular, if the active domain is $\{a_1, \ldots, a_n\}$, the latter step can be seen as chasing a disjunctive EGD of the form

$S(\bar{z}) \rightarrow \bar{z}(i) = a_1 \vee \ldots \vee \bar{z}(i) = a_n$. The resulting instances $g(K')$ are then appended as new children of K. In the special case where there are no homomorphisms g such that $\text{Visible}(g(K')) = \text{Visible}(\mathcal{F}_0)$, we append a dummy instance $\perp$ as a child of K: this denotes 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.

In the limit, the process generates a possibly infinite tree-shaped collection of instances, a *chase tree*. We assume that the chase strategy is "fair": whenever a dependency was applicable in a node on a maximal path of the chase tree, then it was fired at some descendant of that node along the same maximal path (unless the path ends with $\perp$). It now remains to complete the collection with "limits" in order to guarantee that all constraints are satisfied. Consider any infinite path $\pi = K_0, K_1, \ldots$ in the chase tree. By construction, the instances along π 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 refer to [12] for further details about the construction of this limit. Here we only remark that $\lim_{n\in\mathbb{N}} K_n$ satisfies the constraints C. We denote by $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{F}_0)$ the collection of all non-dummy instances that occur as leaves of the chase tree and all the limits of the infinite paths in it. 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 $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{F}_0)$ as a single object.

It is clear that every instance in $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{F}_0)$ satisfies the constraints in C and agrees with $\mathcal{F}_0$ on the visible part of the schema. One can also show that $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{F}_0)$ satisfies the following universal property:

Lemma 9. *Let $\mathcal{F}_0, \mathcal{F}$ be two instances of the same schema $\mathcal{S}$ such that $\mathcal{F}_0 \subseteq \mathcal{F}$, $\text{Visible}(\mathcal{F}_0) = \text{Visible}(\mathcal{F})$, and $\mathcal{F}$ satisfies a set C of TGDs without constants. There exist an instance $K \in \text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{F}_0)$ and a homomorphism from K to $\mathcal{F}$.*

An interesting use of the chase procedure is to characterize the instance-level PQI problem:

Proposition 10. *Let Q be a BUCQ without constants, C a set of TGDs without constants, $\mathcal{V}$ a visible instance. $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}) = \text{true}$ iff Q holds on all instances of $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{V})$.*

Proof. Suppose that $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}) = \text{true}$ and recall that every instance in $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{V})$ satisfies the constraints in C and agrees with $\mathcal{V}$ on the visible part. This means that Q holds on every instance of $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{V})$.

Conversely, if $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}) = \text{false}$, there is an $\mathcal{S}$-instance $\mathcal{F}$ that has $\mathcal{V}$ as visible part, satisfies the constraints in C, but not the query Q. By Lemma 9, letting $\mathcal{F}_0 = \mathcal{V}$, we get an instance $K \in \text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{V})$ and a homomorphism from K to $\mathcal{F}$. Since Q is preserved under homomorphisms, K does not satisfy Q. □

Next, we recall that the visible instance $\mathcal{V}_{\{a\}}$ is constructed over a singleton active domain and that the constraints C have no constants. This implies that there are no disjunctive choices to perform while chasing the constraints starting from $\mathcal{V}_{\{a\}}$. Moreover, it is easy to see that this chase always succeeds, that is, returns a collection $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$ with exactly one instance — this also shows that $\mathcal{V}_{\{a\}}$ is a realizable instance. By a slight abuse of notation, we denote by $\text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$ the unique instance in the collection $\text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$.

The last ingredient is given by the following lemma:

Lemma 11. *If C is a set of TGDs without constants over a schema $\mathcal{S}$ and $\mathcal{V}$ is an instance of the visible part of $\mathcal{S}$, then every instance $K \in \text{Chases}_{\text{vis}}(C, \mathcal{S}, \mathcal{V})$ maps homomorphically to $\text{chase}_{\text{vis}}(\mathcal{S}, C, \mathcal{V}_{\{a\}})$.*

Now that we established the key lemmas, we can reduce the schema-level problem to an instance-level problem:

Proof of Theorem 8. Recall that for the moment we assume that the constraints consist only of TGDs. Clearly, $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}_{\{a\}}) = \text{true}$ implies $\exists\text{PQI}(Q, C, \mathcal{S}) = \text{true}$. For the converse, suppose that $\exists\text{PQI}(Q, C, \mathcal{S}) = \text{true}$. This implies the existence of a realizable instance $\mathcal{V}$ such that $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}) = \text{true}$. By Proposition 10, every instance in $\text{Chases}_{\text{vis}}(\mathcal{S}, C, \mathcal{V})$ satisfies the query Q. Moreover, by Lemma 11, every instance in $\text{Chases}_{\text{vis}}(\mathcal{S}, C, \mathcal{V})$ maps homomorphically to $\text{chase}_{\text{vis}}(\mathcal{S}, C, \mathcal{V}_{\{a\}})$. Hence $\text{chase}_{\text{vis}}(\mathcal{S}, C, \mathcal{V}_{\{a\}})$ also satisfies Q. By applying Proposition 10 again, we conclude that $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}_{\{a\}}) = \text{true}$. □

Now, we explain how to generalize the above proof to deal with both TGDs and EGDs (still without constants). For this we modify the procedure for $\text{Chases}_{\text{vis}}(\mathcal{S}, C, \mathcal{V})$ to take into account the additional EGDs that can be triggered on instances of the chase tree. Formally, chasing an EGD of the form $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \rightarrow x = x'$ 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 leads 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 $\mathcal{V}$. With the new definition of $\text{Chases}_{\text{vis}}(\mathcal{S}, C, \mathcal{V})$ at hand, the proofs of Lemmas 9 and 11 do not pose particular problems, as we just need to handle the standard case of an EGD, and Proposition 10 and Theorem 8 carry over without modifications.

By pairing Theorem 8 with Theorem 4 we obtain:

Corollary 12. *$\exists\text{PQI}(Q, C, \mathcal{S})$ is in 2Exp and finitely controllable, where Q ranges over BUCQs without constants and C over sets of FGTGDs without constants.*

The uniqueness of the instance $\text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$ played a key role in the argument above. We show that adding disjunction to the constraints, thus causing this uniqueness to fail, leads to undecidability. Intuitively, this shows that the interaction of disjunctive linear TGDs and linear EGDs (implicit in the visibility assumption) breaks the critical instance approach.

Theorem 13. *The problem $\exists\text{PQI}(Q, C, \mathcal{S})$ is undecidable as Q ranges over BUCQs and C over sets of disjunctive linear TGDs.*

The proof uses a technique that will be exploited for many of our undecidability arguments. We will reduce the existence of a tiling to $\exists$PQI. 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. There will be challenges to the labelling of adjacent cells, to the initial tile, and to the 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 the UCQ Q will have disjuncts that hold exactly when the challenges are passed. Thus, a correct tiling corresponds to a visible instance where every extension satisfies Q. The undecidability argument also applies to the variant of $\exists$PQI that quantifies over finite instances.

Perhaps even more surprisingly, we show that *disjunction can be simulated using constants (under UNA)*, and thus even the addition of constants causes the critical instance technique to break. The proof adapts the technique of "coding Boolean operations and truth values in the schema", which has been used to eliminate the need for disjunction in several past works (e.g. [17]).

Proposition 14. *There is a polynomial-time reduction from* $\exists\text{PQI}(Q, C, \mathcal{S})$*, as* Q *ranges over BUCQs and* C *over sets of disjunctive linear TGDs, to* $\exists\text{PQI}(Q', C', \mathcal{S}')$*, as* Q' *ranges over BUCQs and* C' *over sets of linear TGDs (with constants).*

From the previous two results we immediately see that the addition of (distinct) constants leads to undecidability:

Corollary 15. $\exists\text{PQI}(Q, C, \mathcal{S})$ *is undecidable as* Q *ranges over BUCQs and* C *over sets of linear TGDs (with constants).*

We now turn to analysing how the complexity scales with less powerful constraints, e.g. linear TGDs without constants. We use the reduction of the schema-level problem $\exists\text{PQI}(Q, C, \mathcal{S})$ to the instance-level problem $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}_{\{a\}})$ given in Proposition 8, and use some ideas from Johnson and Klug's [18] to show that for linear TGDs the latter problem is in polynomial space:

Theorem 16. $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}_{\{a\}})$ *is in* PSpace*, as* Q *ranges over BUCQs without constants and* C *over sets of linear TGDs without constants. Thus by Proposition 8 the same holds for* $\exists\text{PQI}(Q, C, \mathcal{S})$*.*

Proof. By Proposition 10, $\text{PQI}(Q, C, \mathcal{S}, \mathcal{V}_{\{a\}}) = \mathsf{true}$ is equivalent to asking the existence of a CQ Q_i in Q and a homomorphism h from $\mathsf{CanonDB}(Q_i)$ to $\text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$. We can easily guess the CQ Q_i, the homomorphism h, and its image $I = h(\mathsf{CanonDB}(Q_i))$. Thus, it remains to show that $I \subseteq \text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$ can be tested in PSpace.

Recall that $\text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$ is obtained as the limit of a series of operations that consist of alternatively adding new facts, according to the TGDs in C, and identifying with the constant a those values that appear in some visible relation. Note that the second type of operation may also affect tuples that belong to hidden relations and that were inferred during previous steps of the chase. We show that, at the exact moment when a new fact $R(b_1, \ldots, b_k)$ is inferred by chasing a linear TGD, one can also detect whether a value b_i will be eventually identified with the constant a – in this case we can safely replace the fact $R(b_1, \ldots, b_k)$ with $R(b_1, \ldots, b_{i-1}, a, b_{i+1}, \ldots, b_k)$. To detect this we test whether C entails a dependency of the form $R(\bar{x}) \rightarrow \exists \bar{y}\, S(\bar{z})$, where $\bar{x}$ is a sequence of (possibly repeated) variables that has the same equality type as $\bar{b}$ (i.e. $\bar{x}(j) = \bar{x}(j')$ iff $\bar{b}(j) = \bar{b}(j')$), S is a *visible* relation, $\bar{z}$ is a sequence of variables among $\bar{x}, \bar{y}$, and $\bar{x}(i) = \bar{z}(j)$ for some $1 \leq j \leq |\bar{z}|$. The above entailment can be seen as a containment between two CQs under a given set of linear TGDs C, and we know from [18] that the latter problem is in PSpace. We have just described an alternative construction of $\text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$ that avoids substitutions of values with constants, and in which every step can be performed using a PSpace sub-procedure.

Below, we explain how to adapt the techniques from [18] to this alternative variant of the chase in order to decide whether an instance I is contained in $\text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$. For this, it is convenient to think of $\text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$ as a directed graph, whose nodes denote the facts in $\text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$ and the edges denote the inference steps deriving new facts from existing ones – because the constraints are linear TGDs, each inference step depends on at most one fact. By the previous arguments, one can check in PSpace whether an edge exists between two given nodes. We focus on the minimal set of edges that connects all the facts of I to the facts of $\mathcal{V}_{\{a\}}$ (the *roots* of the graph). The graph restricted to this set of edges is a forest. Its height is at most exponential in $|I|$, and each level in it contains at most $|I|$ nodes. Thus, the restricted graph can be explored by a non-deterministic polynomial-space algorithm that guesses the nodes at a level on the basis of the nodes at the previous level and the linear TGDs in C. The algorithm succeeds once it has visited all the facts in I, witnessing $I \subseteq \text{chase}_{\text{vis}}(C, \mathcal{S}, \mathcal{V}_{\{a\}})$. Otherwise, the computation is rejected after seeing exponentially many levels. □

Recall that Open-World Query Answering (OWQ) is the "classical" variant of the query implication problem: all relations can be freely extended, and all relations are allowed to be non-empty. It is easy to show that OWQ reduces to PQI; i.e. PQI is at least as hard as OWQ (and as our results have shown, it is sometimes harder). We can also reduce OWQ to $\exists$PQI, which will allow us to show that the upper bound for linear TGD in Theorem 16 is tight.

Proposition 17. *For any class of constraints containing linear TGDs,* OWQ *reduces to* $\exists$PQI*.*

Proof. Let Q be a query, C a set of constraints over a schema $\mathcal{S}$, and $\mathcal{F}$ an instance of the schema $\mathcal{S}$. We show how to reduce the Open-World Query Answering problem for Q, C, $\mathcal{S}$, and $\mathcal{F}$ to a problem $\exists\text{PQI}(Q', C', \mathcal{S}')$. The idea is to create a copy of the instance $\mathcal{F}$ in the hidden part of the schema, which can be then extended arbitrarily.

Formally, we let the transformed schema $\mathcal{S}'$ consist of all the relations in $\mathcal{S}$, 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 C' contain all the constraints from C, plus a constraint of the form $\mathsf{Good} \rightarrow \exists \bar{y}\, Q_{\mathcal{F}}$, 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 $\mathcal{F}$. The visible instance $\mathcal{V}'$ that contains the atom Good is realizable, since it can be completed (using the chase) to an $\mathcal{S}'$-instance $\mathcal{F}'$ that satisfies the constraints C'. Moreover, every $\mathcal{S}'$-instance $\mathcal{F}'$ that contains the atom Good and satisfies the constraints C', contains an isomorphic copy of $\mathcal{F}$. Thus, if we finally let $Q' = Q \wedge \mathsf{Good}$, we have that $\exists\text{PQI}(Q', C', \mathcal{S}') = \mathsf{true}$ iff every $\mathcal{S}$-instance $\mathcal{F}'$ that contains the fact Good and satisfies the constraints C', also satisfies the query Q. This reduces the Open-World Query Answering problem for Q, C, and $\mathcal{S}$ to the problem $\exists\text{PQI}(Q', C', \mathcal{S}')$. □

We recall that [11] showed that the *implication problem for IDs* (does one ID follow from a set of IDs) is PSpace-hard. There is also a simple reduction from the latter implication problem to OWQ: given the problem of deciding whether a set of IDs Σ implies an ID $\lambda \rightarrow \rho$, one can produce the OWQ problem with constraints Σ, instance given by the canonical database of λ, and query formed from ρ by changing the free variables to constants. Thus OWQ is PSpace-hard for IDs as well, and the PSpace-hardness of $\exists$PQI follows using the result above. Thus, applying the reduction of Proposition 17, we see that the prior upper bound from Theorem 16 is tight:

Corollary 18. $\exists\text{PQI}(Q, C, \mathcal{S})$ *is* PSpace*-hard, as* Q *ranges over BCQs and* C *over sets of linear TGDs.*

Similarly applying existing lower bounds on OWQ for FGTGDs [10] we see that the prior upper bound from Corollary 12 is tight:

Corollary 19. $\exists\text{PQI}(Q, C, \mathcal{S})$ *is* 2Exp*-hard, as* Q *ranges over BCQs and* C *over sets of FGTGDs without constants.*

The table below highlights the main results on PQI and $\exists$PQI.

	PQI Data	PQI Combined	$\exists$PQI
NoConst Linear TGD	Exp-complete Thm 5/Thm 6	2Exp-complete Thm 4/Thm 7	PSpace-complete Thm 16/Cor 18
NoConst FGTGD	Exp-complete Thm 5/Thm 6	2Exp-complete Thm 4/Thm 7	2Exp-complete Cor 12/Cor 19
NoConst Disj. Linear TGD	Exp-complete Thm 5/Thm 6	2Exp-complete Thm 4/Thm 7	undecidable Thm 13
Linear TGD & FGTGD & GNFO	Exp-complete Thm 5/Thm 6	2Exp-complete Thm 4/Thm 7	undecidable Cor 15

4. Negative Query Implication

4.1 Instance-level problem

We now consider the problem NQI$(Q, C, \mathcal{S}, \mathcal{V})$. As in the positive case, for GNFO constraints we get decidability via reduction to satisfiability.

Theorem 20. NQI$(Q, C, \mathcal{S}, \mathcal{V})$*, as* Q *ranges over BUCQs and* C *over GNFO constraints, is* 2Exp *in combined complexity,* Exp *in data complexity, and finitely controllable.*

Proof. As in the positive case, we reduce NQI$(Q, C, \mathcal{S}, \mathcal{V})$ to unsatisfiability of the GNFO formula

$$Q \ \wedge \ C \ \wedge \bigwedge_{R \in \mathcal{S}_v} \Big(\bigwedge_{R(\bar{a}) \in \mathcal{V}} R(\bar{a}) \ \wedge \ \forall \bar{x} \, (R(\bar{x}) \rightarrow \bigvee_{R(\bar{a}) \in \mathcal{V}} \bar{x} = \bar{a}) \Big) .$$

The data complexity analysis is as in Theorem 5, since the formulas agree on the part that varies with the instance. □

The fact that the above bounds are tight follows from the lower bounds for combined and data complexity of PQI given in Theorems 6 and 7, since PQI reduces to NQI for rich enough constraints:

Theorem 21. *For any class of constraints that includes connected FGTGDs,* PQI$(Q, C, \mathcal{S}, \mathcal{V})$ *reduces in polynomial time to* NQI$(Q', C', \mathcal{S}', \mathcal{V}')$*. Further,* $Q', C', \mathcal{S}'$ *depend only on* $Q, C, \mathcal{S}$*, and thus the reduction preserves data complexity.*

Proof. We first provide a reduction that works with any class of constraints allowing arbitrary conjunctions in the left-hand sides (e.g. frontier-guarded TGDs). Subsequently, we show how to modify the constructions in order to preserve connectedness.

The schema $\mathcal{S}'$ is obtained by copying both the visible and the hidden relations from $\mathcal{S}$ and by adding the following relations: a visible relation Error of arity 0 and a hidden relation Good of arity 0. The constraints C' will contain the same constraints from C, plus one frontier-guarded TGD of the form

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

for each CQ of Q of the form $\exists \bar{y} \ Q_i(\bar{y})$. Finally, the query and the visible instance are defined as follows: $Q' = \mathsf{Good}$ and $\mathcal{V}' = \mathcal{V}$.

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

We observe that the constraints in the above reduction use left-hand sides that are not connected. In order to preserve connectedness, it is sufficient to modify the above constructions by adding a dummy variable that is shared by all atoms. More precisely, we expand the relations of the schema $\mathcal{S}$ and the relation Good with a new position, 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 the values associated with the "dummy" position. Using the visible instance, we can guarantee that the relation Check contains exactly one value. The constraints are thus modified as follows. Every constraint $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \rightarrow \exists \bar{y} \ S(\bar{z})$ in C' 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, the constraint $Q_i(\bar{y}) \wedge \mathsf{Good} \rightarrow \mathsf{Error}$ becomes $Q_i(\bar{y}, w) \wedge \mathsf{Good}(w) \rightarrow \mathsf{Error}(w)$, which is now a connected frontier-guarded TGD. Furthermore, for every relation $R(\bar{x})$ in $\mathcal{S}$, we add the constraint

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

and we do the same for the relation Good:

$$\mathsf{Good}(w) \ \rightarrow \ \mathsf{Check}(w) \, .$$

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

From the above reduction and from Theorems 6 and 7, we get the following hardness results for instance-based NSB.

Corollary 22. *There are a BUCQ* Q *and a set* C *of connected FGTGDs over a schema* $\mathcal{S}$ *for which the problem* NQI$(Q, C, \mathcal{S}, \mathcal{V})$ *is* Exp*-hard in data complexity (that is, as* $\mathcal{V}$ *varies over instances).*

Corollary 23. *The problem* NQI$(Q, C, \mathcal{S}, \mathcal{V})$*, as* C *ranges over sets of connected FGTGDs,* $\mathcal{S}$ *over schemas,* Q *over BCQs and* $\mathcal{V}$ *over instances, is* 2Exp*-hard.*

Thus far, the complexity of NQI has been similar to that of PQI. We will now show a strong contrast in the case of linear TGDs. Recall from Theorem 6 that PQI was highly intractable even for fixed schema with linear TGDs and fixed query. We begin by showing that NQI$(Q, C, \mathcal{S}, \mathcal{V})$ can be solved easily by looking only at full instances that agree with $\mathcal{V}$ on the visible part and whose active domains are the same as that of $\mathcal{V}$:

Definition 24. NQI$(Q, C, \mathcal{S}, \mathcal{V})$ *is* active domain controllable *if it is equivalent to asking that for all instances* $\mathcal{F}$ *over the active domain* $\mathsf{adom}(\mathcal{V})$ *of* $\mathcal{V}$*, if* $\mathcal{F}$ *satisfies* C *and* Visible$(\mathcal{F}) = \mathcal{V}$*, then* $Q(\mathcal{F}) = \mathsf{false}$.

It is clear that the the problem NQI$(Q, C, \mathcal{S}, \mathcal{V})$ is simpler when it is active domain controllable, as in this case we could guess a full instance $\mathcal{F}$ over the active domain of $\mathcal{V}$ and reduce the problem to checking whether Q holds on $\mathcal{F}$.

We give a simple argument that NQI under IDs is active domain controllable. Let C be a set of IDs, Q a BUCQ, and $\mathcal{V}$ a visible instance such that NQI$(Q, C, \mathcal{S}, \mathcal{V}) = \mathsf{false}$. This means that there is a full instance $\mathcal{F}$ such that $\mathcal{F} \models C$, Visible$(\mathcal{F}) = \mathcal{V}$, and $\mathcal{F} \models Q$. Take any value $a \in \mathsf{adom}(\mathcal{V})$ and let h be the homomorphism that is the identity over $\mathsf{adom}(\mathcal{V})$ and maps any other value from $\mathsf{adom}(\mathcal{F}) \setminus \mathsf{adom}(\mathcal{V})$ to a. Since, C consists of IDs (in particular, since the left-hand side atoms do not have constants or repeated occurrences of the same variable), we know that $h(\mathcal{F}) \models C$. Similarly, we have $h(\mathcal{F}) \models Q$. Hence, $h(\mathcal{F})$ is an instance over the active domain of $\mathcal{V}$ that equally witnesses NQI$(Q, C, \mathcal{S}, \mathcal{V}) = \mathsf{false}$.

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

Example 3. *Let* $\mathcal{S}$ *be the schema with a hidden relation* H *of arity* 2 *and two visible relations* V_1, V_2 *of arities* $1, 0$*, respectively. Consider the linear TGDs*

$$H(x, y) \ \rightarrow \ V_1(x) \qquad\qquad H(x, x) \ \rightarrow \ V_2$$

the CQ $Q = \exists x \, y \, H(x, y)$*, and the visible instance* $\mathcal{V}$ *that consists of the single fact* $V_1(a)$*. Clearly, every full instance* $\mathcal{F}$ *over the active domain* $\{a\}$ *of* $\mathcal{V}$ *that satisfies the above constraints and CQ* Q *must also contain the facts* $H(a, a)$ *and* V_2*, and so it cannot agree with* $\mathcal{V}$ *on the visible part. On the other hand, the full instance that contains the facts* $V_1(a)$ *and* $H(a, b)$*, for a fresh value* b*, satisfies the constraints and the CQ* Q*, and furthermore agrees with* $\mathcal{V}$*. This shows that* NQI$(Q, C, \mathcal{S}, \mathcal{V})$ *is not active domain controllable.*

Despite the above example, we show that we can still transform any schema with linear TGDs (which may include constants) and any query for an NQI problem so as to enforce active domain controllability. Moreover, we can do so while preserving the visible instance of the problem:

Theorem 25. *Given a schema $\mathcal{S}$, a set C of linear TGDs (possibly including constants), and a BUCQ Q, one can construct in exponential time a new schema $\mathcal{S}'$, a set C' of linear TGDs (with constants), and a BUCQ Q' such that $\mathcal{S}_v = \mathcal{S}'_v$ and, for all instances $\mathcal{V}$ over $\mathcal{S}_v$:*

1. $\text{NQI}(Q, C, \mathcal{S}, \mathcal{V}) = \text{NQI}(Q', C', \mathcal{S}', \mathcal{V})$,
2. $\text{NQI}(Q', C', \mathcal{S}', \mathcal{V})$ *is active domain controllable.*

The idea for proving the above result is to project away from the tuples of the hidden relations of $\mathcal{S}$ those positions that store values outside the active domain of $\mathcal{V}$. In doing so, one needs to recall the equality relationships between pairs of removed positions and between those positions and the constants used in the constraints – this is important because different linear TGDs may be triggered on the basis of these equalities. For example, consider a hidden ternary relation $R = \{(a,a,c),(a,c,d),(b,c,c),(c,d,e),(c,d,c)\}$ for the original schema $\mathcal{S}$, where a, b are the only values of the visible active domain. In the modified schema, the relation R is represented by copies of it of the form $R_{I,\varphi}$, one for each set I of removed positions and for each equality type φ. Precisely, the copies of R in the modified instance will be: $R_{\emptyset,\top} = \emptyset$, $R_{\{z\},\top} = \{(a,a)\}$, $R_{\{y,z\},\top} = \{(a),(b)\}$, $R_{\{y,z\},y=z} = \{(b)\}$, $R_{\{x,y,z\},\top} = \{()\}$, $R_{\{x,y,z\},x=z} = \{()\}$, $R_{\{x,y,z\},x=y} = R_{\emptyset,y=z} = R_{\emptyset,x=y=z} = \emptyset$. With the new schema $\mathcal{S}'$ defined, we introduce copies of the original constrains on the basis of the intended semantics of the relations $R_{I,\varphi}$. The goal is to simulate the behaviour of the original tuples using only their projection on the visible active domain and the information about the equalities between the removed positions and the constants. For example, if

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

is a linear TGD in C, then we add to C' linear TGDs of the form

$$\begin{aligned} R_{\emptyset,\top}(x,y,z) &\;\rightarrow\; \exists w\, S_{\emptyset,\top}(w,w) \\ R_{\emptyset,\top}(x,y,z) &\;\rightarrow\; \exists w\, S_{\{w,w'\},\top}() \\ R_{\emptyset,\top}(x,y,z) &\;\rightarrow\; \exists w\, S_{\{w,w'\},w=w'}() \\ &\dots \end{aligned}$$

Finally, we process the UCQ Q in a similar way. A fully detailed construction of $\mathcal{S}'$, C', and Q', with the proof that they satisfy the desired properties, is given in the full version.

Now we show how to exploit active domain controllability to prove that NQI can be solved not only efficiently, but "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] – i.e. as finite sets of rules of the form $R(\bar{x}) \leftarrow Q(\bar{x})$ where the $\bar{x}_i$ are implicitly universally quantified and 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 $\mathcal{F}$ consisting of both extensional and intensional relations, returns the instance $\mathcal{F}'$ where the extensional relations are as in $\mathcal{F}$ and the tuples of each intensional relation R are those satisfying $Q(R)$, where Q is any query appearing on the right of a rule with R. The immediate consequence operator is monotone, and the semantics of the GFP-Datalog program on an extensional database instance $\mathcal{F}$ is defined as the greatest fixpoint of this operator starting at the database instance $\mathcal{F}_0$ that extends $\mathcal{F}$ by setting each intensional relation "maximally" – that is, to the tuples of values from the active domain of $\mathcal{F}$ plus the constants appearing in the GFP-Datalog program. A program may also include a distinguished intensional predicate, the *goal predicate* G – in this case the result is taken to be the projection onto G of the greatest fixpoint.

Theorem 26. *If Q is a BUCQ, C a set of linear TGDs (with constants), and $\text{NQI}(Q, C, \mathcal{S}, \mathcal{V})$ is active domain controllable, then $\neg\text{NQI}(Q, C, \mathcal{S}, \mathcal{V})$, viewed as a query over the visible part $\mathcal{V}$, is definable by a GFP-Datalog program that can be constructed in polynomial time from Q, C, and $\mathcal{S}$.*

Proof. We need to describe by means of a GFP-Datalog program the function $\neg\text{NQI}(Q, C, \mathcal{S}, \mathcal{V})$ that maps an instance $\mathcal{V}$ to true or false depending on whether or not Q holds over some instance $\mathcal{F}$ that satisfies the constraints C and such that $\text{Visible}(\mathcal{F}) = \mathcal{V}$. Thanks to active domain controllability, it suffices to consider full instances constructed over the active domain of $\mathcal{V}$, and define a witness $\mathcal{F}$ as a greatest fixpoint.

The extensional relations are those in the visible part $\mathcal{V}$. The intensional relations are those in the hidden part of the schema $\mathcal{S}$, plus an extra intensional relation A that derives the values in the active domain of $\mathcal{V}$. For each extensional relation R and each position $1 \le i \le \text{arity}(R)$, we have the rule $A(x_i) \leftarrow R(\bar{x})$, which collects the values from the active domain into A. In addition, for each intensional relation R, we have the rule

$$R(\bar{x}) \;\leftarrow\; \bigwedge_i A(x_i) \;\wedge\; \bigwedge_{\substack{R(\bar{x}) \rightarrow \exists \bar{y}\, S(\bar{z}) \\ \text{linear TGD in } C}} S(\bar{z})\,.$$

Let $\mathcal{F}$ be the instance consisting of the visible part $\mathcal{V}$ and the intensional relations R computed by the above Datalog program under the greatest fixpoint semantics. It is easy to see that $\mathcal{F}$ satisfies the constraints, that is: if $R(\bar{x}) \rightarrow \exists \bar{y}\, S(\bar{z})$ is a linear TGD in C and $\mathcal{F}$ contains a fact of the form $R(h(\bar{x}))$, for some homomorphism h, then $\mathcal{F}$ contains also a fact $S(h'(\bar{x}))$, for some homomorphism h' that extends h. Finally, to compute $\neg\text{NQI}(Q, C, \mathcal{S}, \mathcal{V})$, we add to the above program the goal predicate G and a rule $G \leftarrow S_1(\bar{z}_1) \wedge \dots \wedge S_n(\bar{z}_n)$ for each CQ $\exists \bar{y}\, S_1(\bar{z}_1) \wedge \dots \wedge S_n(\bar{z}_n)$ of Q. □

We remark 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. Note also that the transformation of Theorem 25 does not change the visible relations, which determine the maximum arity of the intensional relations of the GFP-Datalog program. Thus the constructions of Theorems 25 and 26 give us:

Corollary 27. *When C ranges over sets of linear TGDs and Q over BUCQs $\text{NQI}(Q, C, \mathcal{S}, \mathcal{V})$ has data complexity in* P *and combined complexity in* Exp.

Example 4. *Returning to the medical example from the introduction, Example 1, we see that the GFP-Datalog program is quite intuitive: since* Patient *is empty in the instance and we have a referential constraint from* Appointment *into* Patient, Appointment *is removed as well, leaving the empty instance. The program then simply evaluates the query on the resulting instance, which returns false, indicating that an* NQI *does hold on the original instance.*

Note that this result is in contrast to the situation with the OWQ problem for linear TGDs, where the instances for which an implication holds are definable in ordinary Datalog [4]. We do not know whether the use of GFP-Datalog can be replaced by other logics, e.g. Datalog. However we can prove that to define $\neg\text{NQI}(Q, C, \mathcal{S}, \mathcal{V})$ from a given visible instance $\mathcal{V}$, it is necessary to go beyond first-order logic:

Proposition 28. $\mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V})$ *cannot be described by a first-order query over* $\mathcal{V}$*. More generally, there are BCQs* Q *and sets of IDs* C *such that* $\mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V})$ *is* P*-hard in data complexity.*

As for the combined complexity of NQI with linear TGDs, we can prove a tight Exp lower bound:

Theorem 29. $\mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V})$ *is* Exp*-hard for combined complexity, as* C *ranges over IDs and* Q *over BUCQs.*

The proof is based on a reduction from the acceptance problem for an alternating PSpace Turing machine to $\mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V})$. Compared to the proof of Theorem 6, the techniques are similar. The only difference is that now, thanks to the dependency of the constraints from the machine and its input, we can directly avoid badly-formed encodings of computations. So, the query will only detect rejecting computations and, dually, it will be violated on the encodings of successful computations. In particular, we have that $\mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V}) = \mathsf{true}$ iff all instances satisfying C encode a successful computation of the alternating PSpace Turing machine.

4.2 Schema-level problem

Here we consider the schema-level question, i.e. ∃NQI. We first show that when the constraints are preserved under disjoint unions (this holds, e.g., for connected FGTGDs), the existence of an NQI can be checked by considering a single "negative critical instance", namely the empty visible instance. For TGD constraints, this instance is easily seen to be realizable: the chase procedure that we introduced in Section 3.2 terminates immediately when initialized with the empty instance $\mathcal{F}_0 = \emptyset$ and returns the singleton collection $\mathrm{Chases}_{\mathrm{vis}}(C, \mathcal{S}, \emptyset)$ consisting of the empty $\mathcal{S}$-instance satisfying C.

Theorem 30. *If the constraints* C *consist of TGDs preserved under disjoint unions of instances, then* $\exists\mathrm{NQI}(Q, C, \mathcal{S}) = \mathsf{true}$ *iff* $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset) = \mathsf{true}$*.*

Proof. Clearly, $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset) = \mathsf{true}$ implies $\exists\mathrm{NQI}(Q, C, \mathcal{S}) = \mathsf{true}$. As for the converse, suppose that $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset) = \mathsf{false}$ and let $\mathcal{F}$ be an $\mathcal{S}$-instance satisfying C and Q and such that $\mathrm{Visible}(\mathcal{F}) = \emptyset$. We aim at proving that $\mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V}) = \mathsf{false}$ for all realizable visible instances $\mathcal{V}$. Let $\mathcal{V}$ be a realizable instance and $\mathcal{F}'$ an $\mathcal{S}$-instance that satisfies C 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 constraints C are preserved under disjoint unions, $\mathcal{F}''$ satisfies C. Moreover, by monotonicity, $\mathcal{F}''$ satisfies the UCQ Q. Since $\mathcal{V} = \mathrm{Visible}(\mathcal{F}') = \mathrm{Visible}(\mathcal{F}'')$, we have $\mathrm{NQI}(Q, C, \mathcal{S}, \mathcal{V}) = \mathsf{false}$, whence $\exists\mathrm{NQI}(Q, C, \mathcal{S}) = \mathsf{false}$. □

By the above result and Theorem 20, we get that $\exists\mathrm{NQI}(Q, C, \mathcal{S})$ is decidable in 2Exp for GNFO constraints that are closed under disjoint unions, and in particular the problem is decidable for *connected FGTGDs*. Combining with Corollary 27 we also get an Exp bound for linear TGDs. In fact, we can improve the latter bound by observing that NQI over the empty visible instance reduces to Open-World Query Answering:

Proposition 31. *For any BCQ* Q*, constraints* C*, and schema* $\mathcal{S}$*,* $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset) = \mathsf{true}$ *iff* $\mathrm{OWQ}(Q', C, \mathsf{CanonDB}(Q)) = \mathsf{true}$*, where* $Q' = \bigvee_{R \in \mathcal{S}_v} \exists \bar{x}\, R(\bar{x})$ *and* $\mathsf{CanonDB}(Q)$ *is the canonical database of the CQ* Q *(defined in Section 2).*

We know from previous results [5] that OWQ for CQs and linear TGDs is in PSpace. Thus the above reduction implies that $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset)$ (and hence $\exists\mathrm{NQI}(Q, C, \mathcal{S})$, by Theorem 30) is in PSpace when C is a set of linear TGDs:

Corollary 32. $\exists\mathrm{NQI}(Q, C, \mathcal{S})$ *is in* PSpace*, as* Q *ranges over BUCQs and* C *over sets of linear TGDs.*

There are matching lower bounds for ∃NQI. Recall that Theorem 30 and Theorem 20 implied a 2Exp bound for connected FGTGDs. To show a matching lower bound the key tool is the following reduction.

Proposition 33. *There is a polynomial time reduction from* OWQ *over a set of connected FGTGDs without constants and a connected BCQ to an* ∃NQI *problem over a set of connected FGTGDs without constants and a BCQ.*

The existence of such a reduction is surprising, since OWQ deals with deriving positive information while ∃NQI concerns negative information.

Towards proving this reduction, we first state a characterization of $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset)$. Recall that the latter problem is related to ∃NQI: indeed, Theorem 30 reduces $\exists\mathrm{NQI}(Q, C, \mathcal{S})$ to $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset)$ when the constraints C are preserved under disjoint unions. The proof of the characterization of $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset)$, given in the full version, exploits the universality of our variant of the chase (Lemma 9).

Proposition 34. *If* Q *is a BCQ and* C *is a set of TGDs without constants, then* $\mathrm{NQI}(Q, C, \mathcal{S}, \emptyset) = \mathsf{true}$ *iff either* Q *contains a visible atom, or it does not and in this case* $\mathrm{Chases}_{\mathrm{vis}}(C, \mathcal{S}, \mathsf{CanonDB}(Q)) = \emptyset$*.*

Using this we can prove the reduction from OWQ to ∃NQI:

Proof of Proposition 33. Consider the Open-World Query answering problem over a schema $\mathcal{S}$, a set C of constraints without constants and closed under disjoint union, a BCQ Q, and a $\mathcal{S}$-instance $\mathcal{F}$. We reduce this problem to an ∃NQI problem over a new schema $\mathcal{S}'$, a new set of constraints C', and a new BCQ Q'. The schema $\mathcal{S}'$ is obtained from $\mathcal{S}$ by adding a relation Good of arity 0, which is assumed to be the only visible relation in $\mathcal{S}'$. The set of constraints C' is equal to C unioned with the constraint

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

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 these variables. Note that $\mathsf{CanonDB}(Q')$ is isomorphic to the input instance $\mathcal{F}$.

Now, assume that the original constraints in C were connected FGTGDs and the CQ Q was also connected. By construction, the constraints in C' turn out to be also connected FGTGDs. In particular, the satisfiability of these constraints is preserved under disjoint unions, and hence from Theorem 30, $\exists\mathrm{NQI}(Q', C', \mathcal{S}') = \mathsf{true}$ iff $\mathrm{NQI}(Q', C', \mathcal{S}', \emptyset) = \mathsf{true}$. Thus, it remains to show that $\mathrm{NQI}(Q', C', \mathcal{S}', \emptyset) = \mathsf{true}$ iff $\mathrm{OWQ}(Q, C, \mathcal{F}) = \mathsf{true}$.

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

Conversely, suppose that $\mathrm{NQI}(Q', C', \mathcal{S}', \emptyset) = \mathsf{false}$. Recall that the constraints in C' do not use constants and Q' contains no visible facts. We can thus apply Proposition 34 and derive $\mathrm{Chases}_{\mathrm{vis}}(C', \mathcal{S}', \mathsf{CanonDB}(Q')) \neq \emptyset$. Note that $\mathsf{CanonDB}(Q')$ is clearly isomorphic to the original instance $\mathcal{F}$. In particular, there is an instance K in $\mathrm{Chases}_{\mathrm{vis}}(C', \mathcal{S}', \mathsf{CanonDB}(Q'))$ that contains the original instance $\mathcal{F}$, satisfies the constraints in C', 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 $\mathcal{S}$, witnesses the fact that $\mathrm{OWQ}(Q, C, \mathcal{F}) = \mathsf{false}$. □

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 1) these two variants agree. Hence we do not distinguish them. Similar remarks hold for other uses of OWQ within proofs in the paper.

From Proposition 33 and a prior 2Exp-hardness result [10], we get the following lower bound:

Theorem 35. *$\exists$NQI$(Q, C, \mathcal{S})$ is 2Exp-hard as Q ranges over BCQs and C over sets of connected FGTGDs.*

Using a reduction from the implication problem for IDs, shown PSpace-hard in [11], we get tightness of the bounds for linear TGDs:

Theorem 36. *$\exists$NQI$(Q, C, \mathcal{S})$ is PSpace-hard as Q ranges over BCQs and C over sets of linear TGDs.*

Finally, recall that our decidability result for $\exists$NQI applied only to connected FGTGDs. We can show that the connectedness property is critical for decidability.

Theorem 37. *$\exists$NQI$(Q, C, \mathcal{S})$ is undecidable, as Q ranges over BCQs and C over sets of FGTGDs.*

Proof. We give a reduction from the *model conservativity problem* for $\mathcal{EL}$ TBoxes, shown undecidable in [22]. Intuitively, $\mathcal{EL}$ is a logic that defines FGTGDs over relations of arity 2, called "TBoxes". Given TBoxes ϕ_1 and ϕ_2 over two schemas $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, with $\mathcal{S}_1 \subseteq \mathcal{S}_2$, we say that ϕ_2 is a *model conservative extension* of ϕ_1 if every $\mathcal{S}_1$-instance $\mathcal{V}$ that satisfies ϕ_1 can be extended to an $\mathcal{S}_2$-instance that satisfies ϕ_2 without changing the interpretation of the predicates in $\mathcal{S}_1$, that is, by only adding an interpretation for the relations that are in $\mathcal{S}_2$ but not in $\mathcal{S}_1$. The model conservativity problem consists of deciding whether ϕ_2 is a model conservative extension of ϕ_1. In [22] this problem is proved to be undecidable for both finite instances and arbitrary instances.

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

A summary of results on negative implication is below. We notice that the decidable cases are orthogonal to those for positive implications: the dividing line for the positive case concerned the presence of disjunction, while for the negative case it concerns connectedness. Note also that unlike in the positive cases, we have tractable cases for data complexity.

	NQI Data	NQI Combined	$\exists$NQI
Linear TGD	P-complete Cor 27/Prop 28	Exp-complete Cor 27/Thm 29	PSpace-complete Cor. 32/Thm 36
Conn. Disj. FGTGD	Exp-complete Thm 20/Thm 21	2Exp-complete Thm 20/Thm 21	2Exp-complete Thm 30/Thm 35
FGTGD & GNFO	Exp-complete Thm 20/Thm 21	2Exp-complete Thm 20/Thm 21	undecidable Thm 37

5. Conclusions

This work gives a detailed examination of implication of query results from schemas with hidden relations in the presence of constraints in expressive integrity constraint languages. In future work we will look at whether a query is implied over "typical models", in the spirit of Miklau and Suciu's [23].

Acknowledgements. Benedikt's work was sponsored by the Engineering and Physical Sciences Research Council of the United Kingdom, grants EP/M005852/1 and EP/L012138/1. Bourhis was supported by CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020 and ANR Aggreg project ANR-14-CE25-0017.

References

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

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

[3] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *IJCAI*, 2009.

[4] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI*, 2011.

[5] V. Bárány, G. Gottlob, and M. Otto. Querying the guarded fragment. In *LICS*, 2010.

[6] V. Bárány, B. ten Cate, and M. Otto. Queries with guarded negation. In *VLDB*, 2012.

[7] V. Bárány, B. ten Cate, and L. Segoufin. Guarded negation. In *ICALP*, 2011.

[8] M. Benedikt, P. Bourhis, B. ten Cate, and G. Puppis. Query visible and invisible tables in the presence of constraints, 2015. arxiv.org.

[9] M. Benedikt, T. Colcombet, B. ten Cate, and M. V. Boom. The complexity of boundedness for guarded logics, 2015.

[10] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *JAIR*, 48:115–174, 2013.

[11] M. Casanova, R. Fagin, and C. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. *JCSS*, 28(1):29–59, 1984.

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

[13] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *PODS*, 2008.

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

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

[16] T. Gogacz and J. Marcinkowski. All-instances termination of chase is undecidable. In *ICALP*, 2014.

[17] G. Gottlob and C. Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comp.*, 183, 2003.

[18] D. S. Johnson and A. C. Klug. Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies. *JCSS*, 28(1):167–189, 1984.

[19] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Query-based data pricing. In *PODS*, 2012.

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

[21] C. Lutz, I. Seylan, and F. Wolter. Ontology-mediated queries with closed predicates. In *IJCAI*, 2015.

[22] C. Lutz and F. Wolter. Conservative extensions in the lightweight description logic EL. In *CADE*, 2007.

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

[24] Z. Zhang and A. O. Mendelzon. Authorization views and conditional query containment. In *ICDT*, 2005.