Inference from Visible Information and Background Knowledge

Michael Benedikt, Pierre Bourhis, Balder Ten Cate, Gabriele Puppis, Michael Vanden Boom

INFERENCE FROM VISIBLE INFORMATION AND BACKGROUND KNOWLEDGE

MICHAEL BENEDIKT, PIERRE BOURHIS, BALDER TEN CATE, GABRIELE PUPPIS, AND MICHAEL VANDEN BOOM

Department of Computer Science, Oxford University, Parks Rd, Oxford OX1 3QD, UK

CNRS/CRIStAL, Parc scientifique de la Haute Borne 40, avenue Halley. Bat. B, Park Plaza 59650 Villeneuve d'Ascq

Google Inc., Mountainview CA

CNRS / LaBRI, 351 Cours de la Libération, Talence 33405, France

Department of Computer Science, Oxford University, Pars Rd, Oxford OX1 3QD, UK

Abstract. We provide a wide-ranging study of the scenario where a subset of the relations in a relational vocabulary are 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 — which 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 concered 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.

1. Introduction

This paper concerns a setting where there is a collection of relations, but a given user or class of users has access to only a subset of these relations. This could arise, for example in a database setting, where a data owner restricts access to a subset of the stored relations for privacy reasons. Another example comes from information integration, where the integrated schema exposed to users contains both stored relations and "virtual relations", whose contents are not directly accessible, but which have a meaning defined by logical relationships with stored relations. Both of these scenarios can be subsumed by considering a schema consisting of a set of relations that must satisfy a *background theory* (invariants specified by sentences in some logic) with only a subset of the relations visible. A basic computational problem is to determine what questions can be answered by means of reasoning with the background theory and access to the visible relations. Can someone use the content of the visible relations along with background knowledge to answer a given question about the invisible relations?

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.2168/LMCS-???

We study this scenario, where a set of semantically-related relations are hidden while for another set the complete contents are visible. We will 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"). The basic analysis problem will be as follows. We are given a relational vocabulary partitioned into visible and invisible relations, a logical sentence Q (the *query*, representing information whose inference we want to check), and a background theory Σ, again consisting of logical sentences. Our goal is to determine whether we can infer using the visible relations and the background theory some properties about the evaluation of Q. We will be considering variations of the problem in two dimensions:

Instance-level vs. Schema-level Can Q be inferred from Σ and the extensions of the visible relations in a particular instance, where the extension of visible relations are given as input to the problem? Can Q be inferred on *some* instance?

Positive vs. Negative Can it be inferred that Q is true? Can it be inferred that Q is false?

Example 1. Just in order to give intuition for the problems we study in the paper, we give an example from logical analysis of information disclosure, in the spirit of prior works such as [GB14].

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 d of the doctor. A dataowner makes available one projection of $\mathsf{Appointment}$ by creating a relation $\mathsf{Patient}(p)$, defined by the following logical sentences Σ:

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

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

The query $Q = \exists\ a\ \mathsf{Appointment}(\text{"Smith"}, a, \text{"Jones"})$ asking whether patient Smith made an appointment with Dr. Jones can not be inferred under this schema in one sense: an external user with access to $\mathsf{Patient}$ will never be sure that the query is true, in any instance. We say that there can be no Positive Query Implication on any instance for this query, schema, and background theory. 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 instances where the visible relation $\mathsf{Patient}$ does not contain the patient name Smith, an external user will know that the query is false. We say that there is a Negative Query Implication on the visible instances where $\mathsf{Patient}$ does not contain Smith.

Our results. As mentioned above, 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 very rich logical language for background theories, using the same technique: a reduction to the guarded negation fragment of first-order logic (see below). 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 background theories the negative query implication problems are well-behaved as the instance changes, namely, in polynomial time and definable within a well-behaved logic. For the same class of background theories, the corresponding positive query implication questions are hard even when the schema and query are fixed. Our most significant hardness result is that even for simple background theories the positive query inference problem is ExpTime-hard in *data complexity*: that is, when everything except the visible instance is fixed. This is a big jump in complexity for the complexity for special cases of the problem studied in the description logic [LSW13] and database community [AD98] in the past.

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 background knowledge 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 an instance with a query implication is lost.

Our techniques. We make use of 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 which 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 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. [GM14]), and for decidability of Datalog satisfiability [Shm93], 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 the paper we explain how the 4 inference problems we consider (combinations of positive/negative and instance-level/schema-level) differ from previously-studied problems, such as the "open world query answering problem". However, we also provide 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 whether we can infer the truth of a CQ or UCQ: the problems PQI and ∃PQI. Section 5 turns to the problems NQI and ∃NQI, concerning inferring the negation of a CQ or UCQ. Section 6 deals with some small extensions of the framework, and some special cases of the problems of particular interest. We close in Section 7 with conclusions.

Acknowledgements. This is a long version of the extended abstract that appeared in [BBPtC16]. We are quite grateful to the referees of LICS for their helpful comments.

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.

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 conjunctive queries) 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 [AD98] isolate the complexity of this problem in the case where views are defined by conjunctive queries; 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 background theory consists entirely of conjunctive query view definitions. Chirkova and Yu [CY14] extend to the case where conjunctive query 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 papers by Fan and Geerts [FG10a, FG10b] isolate the complexity for several variations of the problem, with particular attention to the case where the completeness assertions are via inclusion dependencies 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 (see in particular the results of Howe et al. in [KUB$^+$12]) while the "Negative Query Implication" (NQI) problem in the second half of the paper is examined in the view context by Mendelzon and Zhang [ZM05], under the name of "conditional emptiness". As in the other work mentioned above, the emphasis has been on view definitions rather than more general background knowledge which 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 positive query implication 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 open world query answering problem matches our notion of positive query implication, except that we restrict to the case where the open-world/visible 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 inclusion dependency from R' to R. As with the database community, the main distinction between our study of the Positive Query Implication problem and the prior work in the DL community concerns the classes of background theories considered. Lutz et al. [LSW12, LSW15, LSW13] 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 the tractable cases, the problem coincides with the classical open-world query answering problem. Franconi et al. [FIS11] show CO-NP-completeness for a disjunction-free description logic. Our results on the data complexity of PQI consider the same problem, but for background theories that are more expressive and, in particular, can handle relations of arbitrary arity, rather than arity at most 2 as in [LSW13, LSW15, FIS11].

In summary, both the database and DL communities considered the PQI questions addressed in this paper, but for background theories that are different from those we consider. The Negative Query Implication problems are not well-studied in the prior literature, and we know of no work at all dealing with the schema-level questions (asking for the existence of an instance with a query implication) in prior work. However, in this paper we show (see Subsection 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 [LW07, KLWW13].

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* [NSV10], which is closely related to the notion of *implicit definability* in classical logic [Bet53]. Determinacy has been extensively studied for both views [NSV10, GM15] and for background theories and visible relations [BtCT16, BtCLT16].

Another contrast is to the work of Miklau and Suciu [MS07] considers whether such an inference is valid probabilistically, looking asymptotically at the uniform distribution over models of increasing size.

Recently [BCK17] 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 [BCK17] builds on the techniques presented in this paper.

3. Definitions

We consider partitioned schemas (or simply, schemas) $\mathbf{S} = \mathbf{S}_h \cup \mathbf{S}_v$, where the partition elements $\mathbf{S}_h$ and $\mathbf{S}_v$ 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. Instances will be used as inputs to the computational problems that are the focus of this work – in this case the instances must be finite. Our computational problems also quantify over instances, and they are also well-defined when the quantification is over all (finite or infinite) instances. For simplicity, by default *instances are always finite*. However, as we will show, taking any of the quantification over all instances will never impact our results, and this will allow us to make use of infinite instances freely in our proofs. The *active domain* of an instance is the set of values occurring within the interpretation of some relation in the instance.

As a suggestive notation, we write $\mathcal{V}$ (Visible) for instances over $\mathbf{S}_v$ and $\mathcal{F}$ (Full) for instances over $\mathbf{S}$. Given an instance $\mathcal{F}$ for $\mathbf{S}$, its restriction to the $\mathbf{S}_v$ relations will be referred to as its *visible part*, denoted $\mathsf{Visible}(\mathcal{F})$.

We will look at background theories defined in a number of logics. One class of logical sentences that we will focus on are Tuple-generating Dependencies (TGDs)m 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 the universally quantified variables $\bar{x}$ appear in $\phi(\bar{x})$. 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 often omit the universal quantifiers, writing just $\phi(\bar{x})\ \rightarrow\ \exists \bar{y}\ \rho(\bar{x}, \bar{y})$. The main feature of TGDs we will exploit is the lack of disjunction, which will allow for cleaner characterizations of our query inference problems.

For TGDs we will be able to obtain clean semantic characterizations for our inference problems. But most inference problems involving TGDs are undecidable [AHV95], including all those we study here. Thus for our decidability and complexity results we will look at classes of TGDs that are computationally better behaved:

- *Linear TGDs*: those where ϕ consists of a single atom.
- *Inclusion Dependencies* (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.
- Many of our results on inclusion dependencies will hold for two more general classes. *Frontier-guarded TGDs* (FGTGDs) [BLMS09] are TGDs where one of the conjuncts of ϕ is an atom that includes every universally quantified variable x_i occurring in ρ. *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 FGTGDs.

- We allow disjunction, by considering *Disjunctive Frontier-guarded 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, 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.
- A key role will be played by an even richer logic, one containing Disjunctive FGTGDs , the *Guarded Negation Fragment*. GNFO is built up inductively according to the grammar:
$$\begin{aligned}\phi\ ::= {} & R(\bar{t}) \mid t_1 = t_2 \mid \exists x\ \phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \\ & R(\bar{t}, \bar{y}) \wedge \neg\phi(\bar{y})\end{aligned}$$
where R is either a relation symbol or the equality relation $x = y$, and the t_i represent either variables or constants. Notice that any use of negation must occur conjoined with an atomic relation that contains all the free variables of the negated

formula – such an atomic relation is a *guard* of the formula. 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 paper [BtCO12] gives both Relational algebra and SQL-based syntax for GNFO, and argues that it covers useful queries and database integrity constraints in practice.

For simplicity (so that all of our sentences are well-defined on instances) we will always assume that our GNFO formulas are domain-independent; to enforce this we can use the relational algebra syntax for capturing these queries, mentioned above.

For many of the results in the paper, 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 above. GNFO contains every positive existential formula, is closed under Boolean combinations of sentences, and it subsumes disjunctive frontier-guarded TGDs up to equivalence. That is, by simply writing out a disjunctive frontier-guarded TGD using $\exists, \neg, \wedge$, one sees that these are expressible in GNFO.

Secondly, we will use that GNFO is "tame", encapsulated in the following result from [BtCS11]:

Theorem 3.1 ([BtCS11])**.** Satisfiability for GNFO sentences can be tested effectively, and is 2ExpTime-complete. Furthermore, every satisfiable sentence has a finite satisfying model.

Note that GNFO does *not* subsume the theories corresponding to CQ view definitions (e.g. $A(x, y) \wedge B(y, z) \leftrightarrow V(x, z)$ cannot be expressed in GNFO). However we will cover this special class of theories in Section 6.

Finally, we will 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. As with TGDs, EGDs generalize some well-known relational database integrity constraints, such as functional dependencies and key constraints. *EGDs with constants* further allow equalities between variables and constants, e.g. $x_i = a$, in the right-hand side.

Our problems take as input a background theory and also a logical sentence whose inference we want to study, the query. In this work we will consider queries specified as *conjunctive queries* (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 *unions of CQs* (UCQs), which are disjunctions (relational algebra unions) of CQs. *Boolean UCQs* are simply UCQs with no free variables. Every CQ Q is associated with a *canonical instance* $\mathsf{CanonInst}(Q)$, where the domain consists of variables and constants of Q and the facts are the atoms of Q.

We will always 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) [AHV95]. While the presence or absence of constants will often make no difference in our results, there are several problems where their presence adds significant complications. In contrast, it is easy to show that the presence of constants without the UNA will never make any difference in any of our results. *Note that in our background theories and query languages above, with the exception of IDs, constants are allowed by default.* When we want to restrict to formulas without constants, we add the prefix NoConst; for example, NoConst-FGTGD denotes the frontier-guarded TGDs that do not contain constants.

The crucial definition for our work is the following:

Definition 3.2. Let Q be a Boolean UCQ over schema $\mathbf{S}$, Σ a logical theory over $\mathbf{S}$, and $\mathcal{V}$ an instance over a visible schema $\mathbf{S}_v \subseteq \mathbf{S}$.

- $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$ if for every finite instance $\mathcal{F}$ satisfying Σ, if $\mathcal{V} = \mathsf{Visible}(\mathcal{F})$ then $Q(\mathcal{F}) = \mathsf{true}$.
- $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$ if for every finite instance $\mathcal{F}$ satisfying Σ, if $\mathcal{V} = \mathsf{Visible}(\mathcal{F})$ then $Q(\mathcal{F}) = \mathsf{false}$.

We call an $\mathbf{S}_v$-instance $\mathcal{V}$ *realizable* w.r.t. Σ if there is an $\mathbf{S}$-instance $\mathcal{F}$ satisfying Σ such that $\mathcal{V} = \mathsf{Visible}(\mathcal{F})$. If an instance $\mathcal{V}$ is not realizable w.r.t. Σ, then, trivially, $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. In practice, realizable instances are the only $\mathbf{S}_v$-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 $\mathbf{S}_v$. But since our lower bound arguments will only involve realizable instances, an alternative definition that assumes realizable inputs yields the same complexity bounds.

$\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ states something about every 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. *For the logical sentences we deal with for our background theories, there will be no difference between these notions.* That is, we will show that the finite and unrestricted versions of PQI coincide for a given class of arguments $Q, \Sigma, \mathbf{S}, \mathcal{V}$. We express this by saying that "$\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ is *finitely controllable*", and similarly for 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 (resp. NQI) problem.

The PQI problem contrasts with the usual *Open-World Query Answering* or *Certain Answer* problem, denoted here $\mathsf{OWQ}(Q, \Sigma, \mathcal{F})$, 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 iff 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 2. Consider a scenario where the background theory consists of inclusion dependencies $F_1(x) \rightarrow \exists y\ U(x, y)$ 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.

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

Definition 3.3. For Q a Boolean conjunctive query over schema $\mathbf{S}$, and Σ a set of sentences over $\mathbf{S}$, we let:

- $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ if there is a realizable $\mathbf{S}_v$-instance $\mathcal{V}$ such that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$;
- $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ if there is a realizable $\mathbf{S}_v$-instance $\mathcal{V}$ such that $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

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. For a class of input $Q, \Sigma, \mathbf{S}$, we say that "$\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$ is *finitely controllable*" if in both quantifications, quantification over finite instances can be freely replaced with quantification over arbitrary instances without changing the truth value of the statement.

4. Positive Query Implication

4.1. **Instance-level problems.** Here we study the problem $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$. Recall that this asks whether $Q(\mathcal{F}) = \mathsf{true}$ for every full instance $\mathcal{F}$ satisfying Σ which agrees with $\mathcal{V}$ in the visible part. The section is organized in three parts: in the first part we prove upper bounds for the PQI problem, establishing a connection to the Guarded Negation Fragment. In the second part we present a technique tailored to background theories of Horn logic, showing that instances witnessing the failure of PQI can be taken to be tree-like. In the third part we use this technique to prove tight lower bounds for the instance-level PQI problem.

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. This is interesting first of all since 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 suffices to capture the fact that an instance has a particular restriction to the visible relations. This is exploited in in the following reduction to the satisfiability problem for GNFO:

Theorem 4.1. The problem $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$, as Q ranges over Boolean UCQs and Σ over GNFO sentences, is in 2ExpTime.
Furthermore, for such sentences the problem is finitely controllable, that is, $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$ iff for every instance $\mathcal{F}$ (of any size) satisfying Σ, if $\mathcal{V} = \mathsf{Visible}(\mathcal{F})$, then $Q(\mathcal{F}) = \mathsf{true}$.

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

$$\begin{aligned}\phi^{\mathsf{PQItoGNF}}_{Q,\Sigma,\mathbf{S},\mathcal{V}} &= \neg Q \ \wedge\ \Sigma \ \wedge \\ &\bigwedge_{R\in\mathbf{S}_v} \Big(\bigwedge_{R(\bar{a})\in\mathcal{V}} R(\bar{a}) \ \wedge\ \forall\bar{x}\ \big(R(\bar{x}) \rightarrow \bigvee_{R(\bar{a})\in\mathcal{V}} \bar{x} = \bar{a}\big)\Big)\end{aligned}$$

Intuitively, the formulas 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.

If the sentences in the background theory are in GNFO, 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 $\forall \bar{x}\ (R(\bar{x}) \rightarrow \dots)$, which translates to $\neg\exists \bar{x}\ (R(\bar{x}) \wedge \neg \dots)$, with the inner negation guarded by $R(\bar{x})$ and the outer negation involving no free variables.

The finite controllability of $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ comes from the finite controllability of GNFO formulas (Theorem 3.1). $\square$

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 must be tested for non-emptiness. By a finer analysis of this translation of GNFO formulas to automata, we can see that the *data complexity* of the problem is only singly-exponential.

Theorem 4.2. If Q is a Boolean UCQ and Σ is a conjunction of GNFO sentences over a schema $\mathbf{S}$, then the data complexity of $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ (that is, as $\mathcal{V}$ varies over instances) is in ExpTime.

Sketch. In the body of the paper, we provide a proof outline for this. What we omit is a fine-grained analysis of the translation of GNFO to automata, extending the translation to automata found in [BCtCV15]. The reader interested in this conversion can find the details in the appendix.

We start by stating a satisfiability result for GNFO formulas ϕ in a normal form, GN-*normal form*, similar to one introduced in [BCS15].

Formulas in GN-normal form can be generated using the following grammar:

$$\phi ::= \textstyle\bigvee_i \exists \vec{x}\ \bigwedge_j \psi_{ij}$$
$$\psi ::= \alpha \mid \alpha \wedge \phi, \mid \alpha \wedge \neg\phi$$

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 (thus allowing free negation of such formulas). The ϕ 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. They are clearly as expressive as general GNFO formulas.

Note that if ϕ_i for $i = 1 \dots n$ are *sentences* in normal form then their conjunction $\bigwedge_i \phi_i$ is also in normal form.

The *width* of ϕ, denoted width(ϕ), is the maximum number of free variables of any subformula of ϕ.

The *CQ-rank* of a formula ϕ in GN-normal form, denoted $\mathrm{rank}_{\mathrm{CQ}}(\phi)$, is the maximum number of conjuncts ψ_i in any CQ-shaped subformula $\exists \vec{x}\ \bigwedge_i \psi_i$ of ϕ for non-empty $\vec{x}$. For the purposes of CQ-rank, $\alpha(\vec{x}) \wedge \neg\phi(\vec{x})$ and $\alpha \wedge \phi$ are treated as subformulas with 1 conjunct.

Theorem 4.3. For every fixed numbers r, m, and w, there is an ExpTime algorithm that determines whether a GNFO formula ϕ in GN-normal form over a schema with relations of arity at most m, with $\mathrm{rank}_{\mathrm{CQ}}(\phi) \leq r$ and width$(\phi) \leq w$ is satisfiable.

Theorem 4.3 is proven by creating an alternating two-way parity automaton whose state set consists of a collection of formulas derived from ϕ. The automaton runs on a tree whose nodes represent collections of elements in a tree-like model. If the formula ϕ were in the guarded fragment, rather than GNFO, it would suffice to use the subformulas of ϕ as states, where the subformulas would have additional annotations associating variables with elements of a guarded set. The bound on the arity would suffice to keep the number of annotations low. In the presence of CQ-shaped formulas, the vertices 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. A further problem is that for CQ-shaped subformulas, one will have to throw in all subformulas, representing 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 $\mathrm{rank}_{\mathrm{CQ}}(\phi)$ guarantees that this need to throw in subformulas does not blow up the number of states.

It is 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.

Now fix a Boolean UCQ Q and a conjunction Σ of GNFO sentences over a schema $\mathbf{S}$. Without loss of generality, we can assume that the sentences in Σ are already in GN-normal form. Consider the formula $\phi^{\mathsf{PSBtoGNF}}_{Q,\Sigma,\mathbf{S},\mathcal{V}}$ in the proof of Theorem 4.1:

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

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 add equality guards on the formulas $x_i \neq a_i$, and guards of the form $R(\vec{x})$ on the disjunctions $\bigvee_i x_i \neq a_i$. With these changes, which do not impact the size of the formula, the conditions of GN-normal form are satisfied.

Thus the formula $\phi^{\mathsf{PSBtoGNF}}_{Q,\Sigma,\mathbf{S},\mathcal{V}}$ can be normalized in polynomial time, and the schema arity and $\mathrm{rank}_{\mathrm{CQ}}$ of $\phi^{\mathsf{PSBtoGNF}}_{Q,\Sigma,\mathbf{S},\mathcal{V}}$ are fixed when Q, Σ, and $\mathbf{S}$ are fixed. Applying Theorem 4.3 the bound claimed in Theorem 4.2 now follows. □

A characterization of PQI for Horn logics. We have shown above that PQI can be reduced to satisfiability of a GNFO formula, and it is known that a satisfiable GNFO formula can always be taken to be "tree-like" — indeed, this is what allows automata-theoretic techniques to be applied. We can give a more concrete algorithm in the special case of background theories in a certain family related to the Horn fragment of first order logic; specifically for EGDs and TGDs. This will not get us better worst-case upper bounds for the cases we consider in this work: indeed, for general TGDs and EGDs it is not even effective. But it will prove useful for showing stronger lower bounds on the combined and data complexity of PQI, since it will allow us to show them when the background theories are in these restricted classes. It will also be essential for the schema-level problems considered in Section 4.2.

We show that for TGDs and EGDs, the PQI problem can be characterized using a variant of the chase procedure [One13, FKMP05]. Our procedure receives as input a relational schema $\mathbf{S}$, a background theory Σ consisting of TGDs and EGDs, and an initial instance $\mathcal{F}_0$ for the schema $\mathbf{S}$, 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 [DNR08], which produces multiple instances.

We now describe in detail how the variant for the chase procedure works, since we will need it in the remainder of the paper. We start with an explanation in the case where Σ contains only TGDs, and later extend it to handle EGDs.

Recall that w.l.o.g. TGDs are assumed to have exactly one atom in the right-hand side. Due to the unique name assumption (UNA), functions between domain elements are tacitly assumed to preserve all the constants that appear in the sentences of the background theory and in the query (i.e. $h(a) = a$ whenever a appears as a constant in Σ or Q). As usual, such functions are homomorphically extended to relational instances (i.e. $h(R(x_1, \ldots, x_n)) = R(h(x_1), \ldots, h(x_n))$ for all relations R). The procedure builds a *chase tree* of instances, starting with the singleton tree consisting of the input $\mathbf{S}$-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. 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 variables $\bar{z}$ to some values in the active domain $\{a_1, \ldots, a_n\}$ of the visible instance $\mathsf{Visible}(\mathcal{F}_0)$. 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 papers on the chase [DNR08], this step would be a sequence of "disjunctive chase step", 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" in order 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 [CK90]. The limit instance $\lim_{n \in \mathbb{N}} K_n$ satisfies the background theory Σ. We denote by $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{F}_0)$ 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 $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{F}_0)$ as a single object.

We now indicate how we can modify the procedure for $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V})$ so as to take into account also the EGDs in Σ that can be triggered on the instances that emerge in the chase tree. Formally, chasing an EGD of the form $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \rightarrow x = x'$, where x, x' are two variables from $\bar{x}_1, \ldots, \bar{x}_m$, amounts at 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 $\mathcal{V}$.

It is clear that every instance in $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{F}_0)$, except the special "failure instance", satisfies the sentences in Σ and, in addition, agrees with $\mathcal{F}_0$ on the visible part of the schema. Below, we prove that $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{F}_0)$ satisfies the following property:

Lemma 4.4. Let Σ consist of EGDs and TGDs without constants. Let $\mathcal{F}_0$ be an instance of a schema $\mathbf{S}$ and let $\mathcal{F}$ be another instance over the same schema that contains $\mathcal{F}_0$, agrees with $\mathcal{F}_0$ on the visible part (i.e. $\mathsf{Visible}(\mathcal{F}) = \mathsf{Visible}(\mathcal{F}_0)$), and satisfies all sentences of Σ. Then, there exist an instance $K \in \mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{F}_0)$ and a homomorphism from K to $\mathcal{F}$.

Proof. For brevity we prove the result for TGDs only. We consider the chase tree for $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{F}_0)$ 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 $R_1\big(f(\bar{x}_1)\big)$, $\ldots$, $R_m\big(f(\bar{x}_m)\big)$ 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 $R_1\big(h_n(f(\bar{x}_1))\big)$, $\ldots$, $R_m\big(h_n(f(\bar{x}_m))\big)$, it must also contain a fact of the form $S\big(h'(f'(\bar{z}))\big)$, 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 $K' = K_n \cup \big\{S(f'(\bar{z}))\big\}$ 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 $K' = K_n \cup \big\{S(f'(\bar{z}))\big\}$ to $\mathcal{F}$. In particular, h' maps the variables $\bar{z}$ to some values in the active domain of the visible part $\mathsf{Visible}(\mathcal{F}_0)$ 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 $h'_i : K_i \rightarrow \mathcal{F}$, 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 instances of the PQI problem when the sentences in the background theory are TGDs without constants:

Proposition 4.5. If Q is a Boolean UCQ, Σ is a set of TGDs or EGDs without constants over a schema $\mathbf{S}$, and $\mathcal{V}$ is a visible instance, then $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$ iff every instance K in $\text{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V})$ satisfies Q.

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

Conversely, suppose that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$. This means that there is an $\mathbf{S}$-instance $\mathcal{F}$ that has $\mathcal{V}$ as visible part, satisfies the sentences in Σ, but not the query Q. By Lemma 4.4, letting $\mathcal{F}_0 = \mathcal{V}$, we get an instance $K \in \text{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V})$ and a homomorphism from K to $\mathcal{F}$. Since Q is preserved under homomorphisms, K does not satisfy Q. $\square$

Lower bounds. Below we show that the data complexity bound in Theorem 4.2 is tight even for inclusion dependencies (IDs). The proof proceeds by showing that a "universal machine" for alternating PSPACE can be constructed by fixing appropriate $Q, \Sigma, \mathbf{S}$ in a PQI problem.

Theorem 4.6. There are a Boolean CQ Q and a set Σ of IDs over a schema $\mathbf{S}$ for which the problem $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ is EXPTIME-hard in data complexity.

Proof. We first prove the hardness result using a UCQ Q; later, we show how to generalize this to a CQ. We reduce the acceptance problem for an alternating PSPACE Turing machine M to the negation of $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$.

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 2 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 (resp., universal) control state has exactly one (resp., 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 control state, and from there onwards 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 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 the 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 can *not* be computed easily from M. In summary, below we will be describe each part of the schema $\mathbf{S}$ 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 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, that are intended to distinguish the first and last cells of the tape. Given M, we will instantiate First (resp., Last) with the singleton consisting of the identifier of the first (resp., 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 $(a, \triangleleft)$ (resp., $(a, \triangleright)$), this means that its content is a and the cell is to the immediate left (resp., 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 $\mathsf{First}^C(x, y)$ 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 $S^{\exists}$, $S^{\forall}_1$, and $S^{\forall}_2$. 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 $S^{\forall}_1(x, x_1)$ and $S^{\forall}_2(x, x_2)$.

So far, we have introduced the visible relations T, First, Last, V, I, and the hidden relations C, T^C, First^C, $S^{\exists}$, $S^{\forall}_1$, $S^{\forall}_2$. These are sufficient to store an encoding of the computation tree of the machine. However, the background theories are only allowed to contain inclusion dependencies, 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 inclusion dependencies.

We now list some inclusion dependencies 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:
$$\begin{array}{rclcrcl} T^C(x, y, y') & \to & T(y, y') & \qquad & \mathsf{First}^C(x, y) & \to & \exists\, y'\ T^C(x, y, y') \\ \mathsf{First}^C(x, y) & \to & \mathsf{First}(y) & \qquad & T^C(x, y, y') & \to & \exists\, y''\ T^C(x, y', y'') \ . \end{array}$$
Note that, while we can easily enforce that T contains the projection of T^C onto the last two attributes, and similar for First and First^C. It is more difficult, instead, to enforce that T^C contains copies of T annotated with each configuration identifier. This is 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 $\mathsf{First}^C(x, y)$ for each configuration x will be enforced later.
- We proceed by enforcing the existence of values associated with each cell in each configuration:
$$T^C(x, y, y') \ \to\ \exists\, z\ C(x, y, z) \qquad\qquad C(x, y, z) \ \to\ V(z) \ .$$
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:
$$\begin{array}{rclcrcl} I(x) & \to & \exists\, x'\ S^{\exists}(x, x') & \qquad & S^{\exists}(x, x') & \to & \exists\, y\ \mathsf{First}^C(x, y) \\ S^{\exists}(x, x') & \to & \exists\, x_1\ S^{\forall}_1(x', x_1) & & & & \\ S^{\exists}(x, x') & \to & \exists\, x_2\ S^{\forall}_2(x', x_2) & & S^{\forall}_1(x, x_1) & \to & \exists\, y\ \mathsf{First}^C(x, y) \\ S^{\forall}_1(x, x_1) & \to & \exists\, x'\ S^{\exists}(x_1, x') & & & & \\ S^{\forall}_2(x, x_2) & \to & \exists\, x'\ S^{\exists}(x_2, x') & & S^{\forall}_2(x, x_2) & \to & \exists\, y\ \mathsf{First}^C(x, y) \ . \end{array}$$

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, $\mathsf{Err}_{I,\mathsf{first}}$, $\mathsf{Err}_{I,\mathsf{last}}$, $\mathsf{Err}_{I,\mathsf{adj}}$, $\mathsf{Err}_{C,\mathsf{adj}}$, $\mathsf{Err}_{S^\exists}$, $\mathsf{Err}_{S^\forall_1}$, and $\mathsf{Err}_{S^\forall_2}$, 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:
$$Q_C \;=\; \exists\, x\ y\ z\ z'\ C(x,y,z) \;\wedge\; C(x,y,z') \;\wedge\; \mathsf{Err}_C(z,z')\ .$$
- The relation $\mathsf{Err}_{I,\mathsf{first}}$ 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, $\mathsf{Err}_{I,\mathsf{first}}$ contains all the 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:
$$\begin{aligned} Q_{I,\mathsf{first}} \;=\; & \exists\, x\ y\ y'\ z\ z' \\ & I(x) \;\wedge\; \mathsf{First}(y) \;\wedge\; T(y,y') \;\wedge\; C(x,y,z) \;\wedge\; C(x,y',z') \;\wedge\; \mathsf{Err}_{I,\mathsf{first}}(z,z')\ . \end{aligned}$$
- Similarly, the relation $\mathsf{Err}_{I,\mathsf{last}}$ contains pairs of values that cannot be associated with the last two cells in the initial configuration, i.e., $\mathsf{Err}_{I,\mathsf{last}} = (V \times V) \setminus (z_1, z_{-1})$, 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
$$\begin{aligned} Q_{I,\mathsf{last}} \;=\; & \exists\, x\ y\ y'\ z\ z' \\ & I(x) \;\wedge\; T(y,y') \;\wedge\; \mathsf{Last}(y') \;\wedge\; C(x,y,z) \;\wedge\; C(x,y',z') \;\wedge\; \mathsf{Err}_{I,\mathsf{last}}(z,z')\ . \end{aligned}$$
- The relation $\mathsf{Err}_{I,\mathsf{adj}}$ contains pairs of values that cannot appear on any two consecutive cells of the initial configuration, namely, $\mathsf{Err}_{I,\mathsf{adj}}$ contains all the 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
$$Q_{I,\mathsf{adj}} \;=\; \exists\, x\ y\ y'\ z\ z'\ I(x) \;\wedge\; T(y,y') \;\wedge\; C(x,y,z) \;\wedge\; C(x,y',z') \;\wedge\; \mathsf{Err}_{I,\mathsf{adj}}(z,z')\ .$$
- In a similar way we can check violations of labellings of consecutive cells in every configuration. This is done with the binary visible relation $\mathsf{Err}_{C,\mathsf{adj}}$, instantiated with all pairs from $V \times V$ that cannot be adjacent in an arbitrary configuration (for example, the pair $\big((a,\lhd),(b,\rhd)\big)$), and the CQ
$$Q_{C,\mathsf{adj}} \;=\; \exists\, x\ y\ y'\ z\ z'\ T(y,y') \;\wedge\; C(x,y,z) \;\wedge\; C(x,y',z') \;\wedge\; \mathsf{Err}_{C,\mathsf{adj}}(z,z')\ .$$
- The relation $\mathsf{Err}_{S^\exists}$ 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 $\mathsf{Err}_{S_1^\forall}$ (resp., $\mathsf{Err}_{S_2^\forall}$) 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 (resp., second) successor configuration. The corresponding CQs $Q_{S_1^\forall}, Q_{S_2^\forall}$ are defined by

$$\begin{aligned} Q_{S_i^\forall} \;=\; & \exists\; x\; x'\; y\; y'\; y''\; z\; z'\; z''\; z''' \\ & S_i^\forall(x,x') \;\wedge\; T(y,y') \;\wedge\; T(y',y'') \;\wedge \\ & C(x,y,z) \;\wedge\; C(x,y',z') \;\wedge\; C(x,y'',z'') \;\wedge\; C(x_1,y',z''') \;\wedge \\ & \mathsf{Err}_{S_i^\forall}(z,z',z'',z''') \;. \end{aligned}$$

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

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

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

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

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, $\mathsf{Err}_{I,\mathsf{first}}$, $\mathsf{Err}_{I,\mathsf{last}}$, $\mathsf{Err}_{I,\mathsf{adj}}$, $\mathsf{Err}_{C,\mathsf{adj}}$, $\mathsf{Err}_{S^\exists}$, $\mathsf{Err}_{S_1^\forall}$, and $\mathsf{Err}_{S_2^\forall}$. We have described these semantics above, and argued why they can be created in polynomial time. Below, 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 $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_M) = \mathsf{false}$.

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, First^C, $S^\exists$, $S_1^\forall$, $S_2^\forall$, we can easily construct a full instance $\mathcal{F}$ that satisfies all the 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 $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_M) = \mathsf{false}$.

Conversely, suppose that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_M) = \mathsf{false}$. Let $\mathcal{F}$ be an $\mathbf{S}$-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 $S_2^\forall$. 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), $(a, \lhd)$, or $(a, \rhd)$. We observe this is well-defined (that is, every tape position y at every configuration x has exactly one associated value) thanks to the sentences $T^C(x, y, y') \to \exists\, z\ C(x, y, z)$ and $C(x, y, z) \to V(z)$, and thanks to the fact that the query Q_C is violated. Moreover, because the CQs $Q_{I,\mathsf{first}}$, $Q_{I,\mathsf{last}}$, and $Q_{I,\mathsf{adj}}$ are also violated, 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 relations $S^{\exists}$, $S^{\forall}_1(x, x_1)$, and $S^{\forall}_2(x, x_2)$ 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 chose 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, and there is no choice. Moreover, using the assumption that the CQs $Q_{C,\mathsf{adj}}$, $Q_{S^{\exists}}$, $Q_{S^{\forall}_1}$, and $Q_{S^{\forall}_2}$ 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.

We have just shown the ExpTime-hardness result for the data complexity of the PQI problem, using a UCQ as query. To finish the proof of Theorem 4.6, we show that PQI problems for UCQs can be reduced to analogous problems for CQs.

Lemma 4.7. Let $Q = \bigcup Q_i$ be a Boolean UCQ, let Σ be a set of 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 CQ Q', a set Σ' of 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 $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$ iff $\mathsf{PQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{true}$. Moreover, the transformation preserves all logical languages considered for background theories in our results (e.g., inclusion dependencies).

Proof. The general idea is as follows. For every visible (resp., hidden) relation R of $\mathbf{S}$ of arity k, we add to $\mathbf{S}'$ a corresponding visible (resp., 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, for example [GP03], and will also be used later in this paper. However, due to the nature of the PQI problem, we also need to add dummy facts $R'(\perp, \ldots, \perp, 0)$ in order to correctly transfer the validity from the UCQ Q to the CQ Q'. We give below the full details.

As mentioned, the new schema $\mathbf{S}'$ contains a copy R' of each relation R in $\mathbf{S}$, where R' is visible iff R is visible, and R' has arity $k + 1$ iff R has arity k. In addition, the schema $\mathbf{S}'$ contains the visible relations Or, Zero, One of arities 3, 0, 0, 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 $\perp$ that do not belong to the active domain of $\mathcal{V}$. First, we include in $\mathcal{V}'$ the facts $\mathsf{Or}(1, 1, 1)$, $\mathsf{Or}(1, 0, 1)$, $\mathsf{Or}(0, 1, 1)$, $\mathsf{Zero}(0)$, $\mathsf{One}(1)$, and $\mathsf{Bottom}_k(\perp, \ldots, \perp, 0)$ 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 $\mathbf{S}$, we introduce the ID

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

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

$$Q' \;=\; \exists\, b_1 \ldots b_n\ b'_0\ b'_1\ \ldots\ b'_n\ \bigwedge_i Q'_i(b_i) \;\wedge\; \mathsf{Zero}(b'_0) \;\wedge\; \mathsf{One}(b'_n) \;\wedge\; \bigwedge_i \mathsf{Or}(b'_{i-1}, b_i, b'_i)$$

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 $R'(\perp, \ldots, \perp, 0)$ 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 sub-query $\bigwedge_i Q'_i(b_i)$ holds at least with all 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 $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$ iff $\mathsf{PQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{true}$. Suppose that $\mathsf{PQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{true}$ and consider an $\mathbf{S}$-instance $\mathcal{F}$ that satisfies the sentences in Σ and such that $\mathsf{Visible}(\mathcal{F}) = \mathcal{V}$. Without loss of generality, we can assume that the active domain of $\mathcal{F}$ does not contain the values 0, 1, and $\perp$. We can easily transform $\mathcal{F}$ into an $\mathbf{S}'$-instance $\mathcal{F}'$ by expanding all facts with the additional attributed value 1 and by adding new facts of the form $R'(\perp, \ldots, \perp, 0)$, for all relations $R' \in \mathbf{S}'$, together with the visible facts $\mathsf{Or}(1, 1, 1)$, $\mathsf{Or}(1, 0, 1)$, $\mathsf{Or}(0, 1, 1)$, $\mathsf{Zero}(0)$, $\mathsf{One}(1)$, and $\mathsf{Bottom}_k(\perp, \ldots, \perp, 0)$ 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 $\mathsf{PQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{true}$, 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 $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$ and consider an $\mathbf{S}'$-instance $\mathcal{F}'$ that satisfies the sentences in Σ' and such that $\mathsf{Visible}(\mathcal{F}') = \mathcal{V}'$. By selecting from $\mathcal{F}'$ only the facts of the form $R'(\bar{a}, 1)$, with $R \in \mathbf{S}$, and by projecting away the last attribute, we obtain an $\mathbf{S}$-instance $\mathcal{F}$ that satisfies the sentences in Σ and such that $\mathsf{Visible}(\mathcal{F}) = \mathcal{V}$. Since $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$, 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 $R'(\perp, \ldots, \perp, 0)$ 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. □

Applying the lemma above, we have proven Theorem 4.6. □

We note that the above lower bound for data complexity makes use of a schema with arity above 2, even for CQs. 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. [FIS11], which show that the data complexity lies in CO-NP (and can be CO-NP-hard) for certain description logics over arity 2.

We now turn to the combined complexity and show that the 2EXPTIME upper bound of Theorem 4.1 is tight even for IDs.

Theorem 4.8. Checking $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$, where Q ranges over CQs and Σ over sets of inclusion dependencies, is 2EXPTIME-hard for combined complexity.

Proof. This proof builds up on ideas from the previous proof for Theorem 4.6. Specifically, we reduce the acceptance problem for an alternating EXPSPACE Turing machine M to the negation of $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$, where Q is a Boolean UCQ and Σ consists of inclusion dependencies. Note that to further reduce the problem to a Positive Query Implication problem with a Boolean CQ, one can exploit Lemma 4.7.

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 4.6, 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 $T_i, T_{i,\mathsf{zero}}, T_{i,\mathsf{one}}$ of arity i, for all $i = 1, \dots, n$, 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, while $T_{i,\mathsf{zero}}$ (resp., $T_{i,\mathsf{one}}$) is the restriction of T_i to the tuples ending with 0 (resp., 1). We enforce this semantics using a simple induction on $i = 1, \dots, n$ and the following inclusion dependencies:

$$\begin{array}{rrcl@{\qquad}rcl}
 & \mathsf{true} & \rightarrow & T_0() & T_{i,\mathsf{zero}}(y_1, \dots, y_i) & \rightarrow & \mathsf{Zero}(y_i) \\
(\forall j \leq i) & T_i(y_1, \dots, y_i) & \rightarrow & \mathsf{Bit}(y_j) & T_{i,\mathsf{one}}(y_1, \dots, y_i) & \rightarrow & \mathsf{One}(y_i)
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
T_{i-1}(y_1, \dots, y_{i-1}) & \rightarrow & \exists y_i \; T_{i,\mathsf{zero}}(y_1, \dots, y_i) & T_{i,\mathsf{zero}}(y_1, \dots, y_i) & \rightarrow & T_i(y_1, \dots, y_i) \\
T_{i-1}(y_1, \dots, y_{i-1}) & \rightarrow & \exists y_i \; T_{i,\mathsf{one}}(y_1, \dots, y_i) & T_{i,\mathsf{one}}(y_1, \dots, y_i) & \rightarrow & T_i(y_1, \dots, y_i) \; .
\end{array}$$

It is clear that every instance satisfying the above sentences will have $T_n = \mathsf{Bit}^n$, 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 $\bar{y} = (y_1, \dots, y_n)$ and $\bar{y}' = (y'_1, \dots, y'_n)$ 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 $i = 1, \dots, n$ of the following CQs:

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

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 $\bigvee_{1\le i\le n} Q_{\mathsf{adj},i}$ 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 $\{(0,\dots,0)\}$ and $\{(1,\dots,1)\}$, 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 4.6. 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 $C(x,y_1,\dots,y_n,z)$ denotes the existence of a configuration identified by x, where the tape cell represented by $\bar{y}=(y_1,\dots,y_n)$ carries the value z. As usual (cf. proof of Theorem 4.6), we define cell values as elements from a visible unary relation $V=\Sigma_Q \uplus \Sigma_{\triangleleft} \uplus \Sigma_{\triangleright}$, where Σ is the alphabet of the Turing machine, $\Sigma_Q=\Sigma\times Q$. $\Sigma_{\triangleleft}=\Sigma\times\{\triangleleft\}$, $\Sigma_{\triangleright}=\Sigma\times\{\triangleright\}$, Q is the set of its control states, and $\triangleleft,\triangleright$ are fresh symbols. To correctly instantiate the relation C, we create also copies of the relations $T_i, T_{i,\mathsf{zero}}, T_{i,\mathsf{one}}$, 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 $i=0,\dots,n$, $T_{i,\mathsf{zero}}^C$ and $T_{i,\mathsf{one}}^C$ of arity $i+1$, for all $i=1,\dots,n$. We have the following sentences for all $i=1,\dots,n$:

$$(\forall j\le i)\qquad T_i^C(x,y_1,\dots,y_i)\ \to\ \mathsf{Bit}(y_j)$$

$$\begin{aligned} &T_n^C(x,y_1,\dots,y_n)\to\exists z\ C(x,y_1,\dots,y_n,z) && T_{i,\mathsf{zero}}^C(x,y_1,\dots,y_i)\to\mathsf{Zero}(y_i)\\ &C(x,y_1,\dots,y_n,z)\to V(z) && T_{i,\mathsf{one}}^C(x,y_1,\dots,y_i)\to\mathsf{One}(y_i)\end{aligned}$$

$$\begin{aligned} &T_{i-1}^C(x,y_1,\dots,y_{i-1})\to\exists y_i\ T_{i,\mathsf{zero}}^C(x,y_1,\dots,y_i) && T_{i,\mathsf{zero}}^C(x,y_1,\dots,y_i)\to T_i^C(x,y_1,\dots,y_i)\\ &T_{i-1}^C(x,y_1,\dots,y_{i-1})\to\exists y_i\ T_{i,\mathsf{one}}^C(x,y_1,\dots,y_i) && T_{i,\mathsf{one}}^C(x,y_1,\dots,y_i)\to T_i^C(x,y_1,\dots,y_i).\end{aligned}$$

Note that the analog of the sentence $\mathsf{true}\to T_0()$ 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 $(x,y_1,\dots,y_n)$, with $(y_1,\dots,y_n)\in\mathsf{Bit}^n$, and C specifies at least one value z for each configuration identifier x and each cell $(y_1,\dots,y_n)$.

We now turn towards the encoding of the computation tree of M. This is almost the same as in the proof of Theorem 4.6. 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 $S^{\exists}(x,x')$ (resp., $S_1^{\forall}(x,x_1)$, $S_2^{\forall}(x,x_1)$) represents a transition from an existential (resp., universal) configuration x to a universal (resp., existential) configuration x' (resp., 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, $\mathsf{Err}_{I,\mathsf{first}}$, $\mathsf{Err}_{I,\mathsf{last}}$, $\mathsf{Err}_{I,\mathsf{adj}}$, $\mathsf{Err}_{C,\mathsf{adj}}$, $\mathsf{Err}_{S^\exists}$, $\mathsf{Err}_{S_1^\forall}$, and $\mathsf{Err}_{S_2^\forall}$, whose instances are defined exactly as in the proof of Theorem 4.6.

- 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:
$$Q_C \;=\; \exists\, x\, \bar{y}\, z\, z'\; C(x,\bar{y},z) \;\wedge\; C(x,\bar{y},z') \;\wedge\; \mathsf{Err}_C(z,z')\ .$$
- The relation $\mathsf{Err}_{I,\mathsf{first}}$ 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:
$$\begin{aligned} Q_{I,\mathsf{first}} \;=\; & \exists\, x\, \bar{y}\, \bar{y}'\, z\, z' \\ & I(x) \;\wedge\; \mathsf{First}(\bar{y}) \;\wedge\; \bigvee\nolimits_{1\le i\le n} Q_{\mathsf{adj},i}(\bar{y},\bar{y}') \;\wedge \\ & C(x,\bar{y},z) \;\wedge\; C(x,\bar{y}',z') \;\wedge\; \mathsf{Err}_{I,\mathsf{first}}(z,z')\ . \end{aligned}$$
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 $\mathsf{Err}_{I,\mathsf{last}}$, $\mathsf{Err}_{I,\mathsf{adj}}$, $\mathsf{Err}_{C,\mathsf{adj}}$ and UCQs $Q_{I,\mathsf{last}}$, $Q_{I,\mathsf{adj}}$, $Q_{C,\mathsf{adj}}$ 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:

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

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

$$\begin{aligned} Q_{S_2^\forall} \;=\; & \exists\, x\, x_2\, \bar{y}\, \bar{y}'\, \bar{y}''\, z\, z'\, z''\, z''' \\ & S_2^\forall(x,x_2) \;\wedge\; \bigvee\nolimits_{1\le i\le n} Q_{\mathsf{adj},i}(\bar{y},\bar{y}') \;\wedge\; \bigvee\nolimits_{1\le i\le n} Q_{\mathsf{adj},i}(\bar{y}',\bar{y}'') \;\wedge \\ & C(x,\bar{y},z) \;\wedge\; C(x,\bar{y}',z') \;\wedge\; C(x,\bar{y}'',z'') \;\wedge\; C(x_2,\bar{y}',z''') \;\wedge\; \mathsf{Err}_{S_2^\forall}(z,z',z'',z''') \end{aligned}$$

where $\mathsf{Err}_{S^\exists}$, $\mathsf{Err}_{S_1^\forall}$, and $\mathsf{Err}_{S_2^\forall}$ are defined exactly as in the proof of Theorem 4.6.

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

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

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 the previous UCQs and let $\mathcal{V}$ be the instance that captures the intended semantics of the visible relations Zero, One, Bit, V, Err_C, $\mathsf{Err}_{I,\mathsf{first}}$, $\mathsf{Err}_{I,\mathsf{last}}$, $\mathsf{Err}_{I,\mathsf{adj}}$, $\mathsf{Err}_{C,\mathsf{adj}}$, $\mathsf{Err}_{S^{\exists}}$, $\mathsf{Err}_{S_1^{\forall}}$, and $\mathsf{Err}_{S_2^{\forall}}$. We can argue along the same lines of the proof of Theorem 4.6 that M has a successful computation tree iff $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$. □

4.2. **Schema-level problem.** In this section we focus on the schema-level problem $\exists\mathsf{PQI}$, namely, the problem of deciding the existence of a instance $\mathcal{V}$ such that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

Let a be an arbitrary domain element. Further let $\mathcal{V}_{\{a\}}$ be a fixed instance for the visible part of a schema $\mathbf{S}$ whose domain contains the single value a and whose visible relations are singleton relations of the form $\{(a, \dots, a)\}$. We will show that, for certain languages for the background theories, if $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$, 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 to chase termination problems: Proposition 3.7 of Marnette and Geerts [MG10] states a related result for disjunctive TGDs in isolation; Gogacz and Marcincowski [GM14] call such an instance a "well of positivity". The following result shows that the technique applies to TGDs and EGDs without constants.

Theorem 4.9. For every Boolean UCQ Q without constants, and every set Σ of TGDs and EGDs without constants, $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ iff $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_{\{a\}}) = \mathsf{true}$.

First, we prove the theorem for background theories consisting only of TGDs without constants. Then we will show how to generalize the proof in the additional presence of EGDs without constants.

We recall that the visible instance $\mathcal{V}_{\{a\}}$ is constructed over a singleton active domain and the sentences in the background theory Σ have no constants. This implies that there are no disjunctive choices to perform while chasing with the dependencies starting from the initial instance $\mathcal{V}_{\{a\}}$. Moreover, it is easy to see that this chase always succeeds. That is, it returns a collection $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$ with exactly one instance — in particular, $\mathcal{V}_{\{a\}}$ is a realizable instance. By a slight abuse of notation, we denote by $\mathrm{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$ the unique instance in the collection $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$.

Lemma 4.10. If Σ is a set of TGDs without constants over a schema $\mathbf{S}$ and $\mathcal{V}$ is an instance of the visible part of $\mathbf{S}$, then every instance $K \in \mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V})$ maps homomorphically to $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$, that is, $h(K) \subseteq \mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$ for some homomorphism h.

Proof. Recall that the instances in $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V})$ are either leaves or limits of infinite paths of the chase tree. Below, we prove that every instance K in the chase tree for $\mathrm{Chases}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V})$ maps to $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$ 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 $h_0, h_1, \dots$ are homomorphisms mapping instances $K_0, K_1, \dots$ 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 sentences of the background theory, 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 $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$ via a homomorphism h. We also consider an instance K' that is a child of K and is obtained by chasing some dependency $R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \to \exists \bar{y}\ S(\bar{z})$, 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 $1 \leq j \leq m$;
(4) $K' = g\big(K \cup \big\{S(f(\bar{z}))\big\}\big)$.

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 $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$. Since $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$ 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 $K' = g\big(K \cup \big\{S(f(\bar{z}))\big\}\big)$ to $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$. If S is not visible, then we recall that g is the identity on $f(\bar{z})$, and hence h' already maps $K' = g\big(K \cup \big\{S(f(\bar{z}))\big\}\big) = K \cup \big\{S(f(\bar{z}))\big\}$ to $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$. 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 $K' = g\big(K \cup \big\{S(f(\bar{z}))\big\}\big)$ to $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$. □

Now that we established the key lemmas, we can easily reduce the existence problem to an instance-based problem (recall that for the moment we assume that the background theory consists only of TGDs):

Proof of Theorem 4.9 (with TGDs only). One of the two directions is trivial: if $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_{\{a\}}) = \mathsf{true}$, then clearly $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$.

For the converse direction, suppose that $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$. This implies the existence of a realizable instance $\mathcal{V}$ such that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. By Proposition 4.5, every instance in $\mathrm{Chases}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V})$ satisfies the query Q. Moreover, by Lemma 4.10, every instance in $\mathrm{Chases}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V})$ maps homomorphically to $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$. Hence the unique instance in $\mathrm{Chases}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$, i.e. $\mathrm{chase}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V}_{\{a\}})$, also satisfies Q. By applying Proposition 4.5 again, we conclude that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_{\{a\}}) = \mathsf{true}$.

Finally, the second statement of the theorem follows from the fact that the previous proofs are independent of the assumption that relational instances are finite. □

Now, we explain how to generalize the proof of Theorem 4.9 to combinations of TGDs and EGDs (still without constants). Recall that we can modify the procedure for $\mathrm{Chases}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V})$ so as to also take into account the EGDs in Σ that can be triggered on the instances that emerge in the chase tree. Using this extended definition of $\mathrm{Chases}_{\mathsf{vis}}(\mathbf{S}, \Sigma, \mathcal{V})$ at hand, the proof of Lemma 4.10 does not pose particular problems, as one just needs to handle the standard case of an EGD dependency. Finally, the proof of Theorem 4.9 directly uses Proposition 4.5 and Lemma 4.10 as black boxes, and so carries over without any modification.

It is worth remarking that, by pairing Theorem 4.9 with the upper bound and the finite controllability for instance-level problems (Theorem 4.1), one immediately obtains the following:

Corollary 4.11. $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$ with Q ranging over Boolean UCQs and Σ over sets of frontier-guarded TGDs without constants, is decidable in 2ExpTime, and is finitely controllable.

In contrast, we show that allowing disjunctions or constants in the background theory sentences leads to undecidability. We first prove this in the case where the sentences include disjunctions. This shows that the interaction of disjunctive linear TGDs and linear EGDs (implicit in the requirement that in a possible world for an instance $\mathcal{F}$, each fact of a visible relation R world must be one of the R-facts of $\mathcal{F}$) causes the "critical instance" reduction to fail.

Theorem 4.12. The problem $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$ is undecidable as Q ranges over Boolean UCQs and Σ over sets of disjunctive linear TGDs.

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 $\exists\mathsf{PQI}$ 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 labelling 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. The undecidability argument also applies to the "unrestricted version" of $\exists\mathsf{PQI}$, in which both quantifications over instances consider arbitrary instances. This will also be true for all other undecidability results in this work, which always concern the schema-level problems.

Proof of Theorem 4.12. For simplicity, we deal with the "unrestricted variant" of the problem, which asks if there is an arbitrary instance of the visible schema such that every superinstance satisfying the sentences in Σ also satisfies Q. Later we will show to modify the proof for dealing with finite instances.

We reduce the problem of tiling the infinite grid, which is known to be undecidable, to the problem $\exists\mathsf{PQI}$. 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 $\mathbf{S}$, a query Q, and a set of disjunctive linear TGDs over $\mathbf{S}$ such that $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ 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 $\exists\mathsf{PQI}$ 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 (resp., E_V), and that this latter node has an associated tile that satisfies the horizontal constraints H (resp., the vertical constraints V). To do so we use the following disjunctive linear TGDs:

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

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

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 in order 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 in order to challenge functionality. Formally, the horizontal challenge is captured by a hidden ternary relation $\mathsf{HChallenge}_{\mathsf{funct}}$, by the linear TGDs

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

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

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 setennces 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 in order 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 positive query implication for Q_C iff 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 the visible and hidden relations of the schema $\mathbf{S}$, and the associated sentences Σ, we define the query for the $\exists\mathsf{PQI}$ 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 $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ iff there is a correct tiling of the infinite grid, namely, a function $f : \mathbb{N} \times \mathbb{N} \rightarrow T$ that satisfies the conditions 1), 2), and 3) above.

Suppose there is a correct tiling $f : \mathbb{N} \times \mathbb{N} \rightarrow T$. 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: $E_H = \big\{\big((i,j),(i+1,j)\big) \bigm| i,j \in \mathbb{N}\big\}$, $E_V = \big\{\big((i,j),(i,j+1)\big) \bigm| i,j \in \mathbb{N}\big\}$, and $U_t = \big\{(i,j) \bigm| f(i,j) = t\big\}$ 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 the 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 $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ 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 $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ 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 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 $\mathsf{HChallenge}$ 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 the 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.

The variant for finite instances is done by observing that the same reduction produces a periodic grid, which can be represented as a finite instance. □

Perhaps even more surprisingly, we show 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" which has been used to eliminate the need for disjunction in hardness proofs in several past works (e.g. [GP03]). It is also similar to the proof idea used in Lemma 4.7 from earlier in this paper.

Proposition 4.13. There is a polynomial time reduction from $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$, where Q ranges over Boolean UCQs and Σ over sets of disjunctive linear TGDs, to $\exists\mathsf{PQI}(Q', \Sigma', \mathbf{S}')$, where Q' ranges over Boolean UCQs and Σ' over sets of linear TGDs (with constants).

Proof. We transform the schema $\mathbf{S}$ to a new schema $\mathbf{S}'$ as follows. For every visible (resp., hidden) relation R of $\mathbf{S}$ of arity k, we add to $\mathbf{S}'$ a corresponding visible (resp., 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 $\mathbf{S}'$. Then, for every 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 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

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

Note that if needed, we can even 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 $\exists b\ \mathsf{Check}(b) \wedge \mathsf{One}(b)$. Below, we prove that $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ iff $\exists\mathsf{PQI}(Q', \Sigma', \mathbf{S}') = \mathsf{true}$.

For the easier direction, we consider a realizable $\mathbf{S}_v$-instance $\mathcal{V}$ such that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. We can easily transform $\mathcal{V}$ into a realizable $\mathbf{S}'_v$-instance $\mathcal{V}'$ that satisfies $\mathsf{PQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{true}$. 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, $\mathsf{Or}(0,1)$, $\mathsf{Or}(1,0)$, and $\mathsf{Or}(1,1)$.

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

Thus it remains to show that for every fact $\mathsf{Or}(b_1, b_2)$ 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 $\mathsf{Or}(b_1, b_2)$, with $b_1 \neq 1$ and $b_2 \neq 1$. Since $\mathcal{V}'$ is realizable, there is a full $\mathbf{S}'$-instance $\mathcal{F}'$ such that $\mathcal{F}' \models \Sigma'$ and $\mathsf{Visible}(\mathcal{F}') = \mathcal{V}'$. Note that $\mathcal{F}'$ may satisfy Q' and, in particular, the conjunct $\mathsf{Check}(1)$. However, removing the single fact $\mathsf{Check}(1)$ 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 $\mathsf{PQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{true}$. □

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

Corollary 4.14. The problem $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$ is undecidable as Q ranges over Boolean CQs and Σ over sets of linear TGDs (with constants).

We now turn to analysing how the complexity scales with less powerful background theories, e.g. linear TGDs without constants. As before, we reduce $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$ to $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$. We can then reuse some ideas from [JK84] to solve the latter problem in polynomial space:

Theorem 4.15. The problem $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$ as Q ranges over Boolean UCQs and Σ over sets of linear TGDs without constants, is in PSPACE, and the same is true for $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$.

Proof. By Proposition 4.5, $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}_{\{a\}}) = \mathsf{true}$ is equivalent to checking that there is a homomorphism h from $\mathsf{CanonInst}(Q_i)$ of some CQ Q_i of Q to the instance $\mathrm{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$. We can easily guess in NP a CQ Q_i of Q, some homomorphism h from $\mathsf{CanonInst}(Q_i)$, and the corresponding image I of $\mathsf{CanonInst}(Q_i)$ under h. Then, it remains to decide whether I is contained in $\mathrm{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$. Below, we explain how to decide this in polynomial space.

Recall that the instance $\mathrm{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$ is obtained as the limit of a series of operations that consist of alternatively adding new facts according to the TGDs in Σ and identifying the values that appear in some visible relation with the constant a. Note that the second type of operation may also affect tuples that belong to hidden relations (this happens when the values are shared with facts in the visible instance). Also note that the affected tuples could have been inferred during previous steps of the chase. Nonetheless, at the exact moment when a new fact $R(b_1, \ldots, b_k)$ is inferred by chasing a linear TGD, we can detect whether a certain value b_i needs to be eventually identified with the constant a, and in this case we can safely replace the fact $R(b_1, \ldots, b_k)$ with $R(b_1, \ldots, b_{i-1}, a, b_{i+1}, \ldots, b_k)$. More precisely, to decide whether the i-th attribute of $R(\bar{b})$ needs to be instantiated with the constant a, we test whether Σ entails a dependency of the form $R(\bar{x}) \to \exists \bar{y}\ S(\bar{z})$, 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 $1 \leq j \leq |\bar{z}|$. Note that the above entailment can be rephrased as a containment problem between two CQs – i.e. $R(\bar{x})$ and $\exists \bar{y}\ S(\bar{z})$ – under a given set of linear TGDs Σ, and we know from [JK84] that the latter problem is in PSPACE. We also observe that, in order to discover all the values in $R(\bar{b})$ that need to be identified with the constant a, it is not sufficient to execute the above analysis only once on each position $1 \leq i \leq \mathrm{ar}(R)$, as identifying some values with the constant a may change the equality type of the fact and thus trigger new dependencies from Σ (notably, this may happen when the linear TGDs are not IDs). We thus repeat the above analysis on all positions of R and until the corresponding equality type stabilizes – this can be still be done in polynomial space. After this, we add the resulting fact to the chase.

What we have just described is an alternative construction of $\mathsf{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$ in which every chase step can be done using a PSPACE sub-procedure. We omit the routine details showing that this alternative construction gives the same result, in the limit, as the version of the chase that we introduced at the beginning of Section 4.2 (the arguments are similar to the proof of Lemma 4.4).

Below, we explain how to adapt the techniques from [JK84] to this alternative variant of the chase, in order to decide whether the homomorphic image I of some CQ of Q is contained in $\mathsf{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$. For this, it is convenient to think of $\mathsf{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$ as a directed graph, where the nodes represent the facts in $\mathsf{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$ and the edges describe the inference steps that derive new facts from existing facts and sentences in Σ. Note that, because the background theory sentences are linear TGDs, each inference step depends on at most one fact. In particular, the nodes of this graph that have no incoming edge (we call them *roots*) are precisely the facts from the instance $\mathcal{V}_{\{a\}}$, and all the other nodes are reachable from some root. Moreover, by the previous arguments, one can check in polynomial space whether an edge exists between two given nodes.

Now, we focus on the minimal set of edges that connects all the facts of I to some roots in the graph. The graph restricted to this set of edges is a forest, namely, every node in it has at most one incoming edge. Moreover, the height of this forest 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 Σ. The algorithm terminates successfully once it has visited all the facts in I, witnessing that I is contained in $\mathsf{chase}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathcal{V}_{\{a\}})$. Otherwise, the computation is rejected after seeing exponentially many levels. □

We can derive matching lower bounds by reducing Open-World Query Answering (OWQ) to $\exists\mathsf{PQI}$:

Proposition 4.16. For any class of sentences containing linear TGDs, OWQ reduces to $\exists\mathsf{PQI}$.

Proof. Let Q be a query, Σ a set of sentences over a schema $\mathbf{S}$, and $\mathcal{F}$ an instance of the schema $\mathbf{S}$. We show how to reduce the Open-World Query Answering problem for Q, Σ, $\mathbf{S}$, and $\mathcal{F}$ to a problem $\exists\mathsf{PQI}(Q', \Sigma', \mathbf{S}')$. 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 $\mathbf{S}'$ consist of all the relations in $\mathbf{S}$, 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 the sentences from Σ, plus the sentence $\mathsf{Good} \rightarrow \exists \bar{y}\ Q_{\mathcal{F}}$, 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 $\mathcal{F}$. Note that the visible instance $\mathcal{V}_{\mathsf{Good}}$ that contains the atom Good is realizable, since it can be completed (using the chase) to an $\mathbf{S}'$-instance $\mathcal{F}'$ that satisfies the sentences in Σ'. Let $Q' = Q \wedge \mathsf{Good}$. We claim that $\exists\mathsf{PQI}(Q', \Sigma', \mathbf{S}') = \mathsf{true}$ if and only if Q is certain with respect to Σ on $\mathcal{F}$. In one direction, suppose $\exists\mathsf{PQI}(Q', \Sigma', \mathbf{S}') = \mathsf{true}$ holds. The witness visible instance having PQI can only be the instance $\mathcal{V}_{\mathsf{Good}}$. 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 $\mathcal{V}_{\mathsf{Good}}$ 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 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 there is a PQI for $Q', \Sigma', \mathbf{S}$ on $\mathcal{V}_{\mathsf{Good}}$. 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 $\mathcal{V}_{\mathsf{Good}}$ as required.

Thus we have reduced the Open-World Query Answering problem for Q, Σ, and $\mathbf{S}$ to the problem $\exists\mathsf{PQI}(Q', \Sigma', \mathbf{S}')$. ☐

From this and existing lower bounds on the Open-World Query Answering ([CFP84] coupled with a reduction from implication to OWQ for linear TGDs, [CGK13] for FGTGDs), we see that the prior upper bounds from Theorem 4.15 and Corollary 4.11 are tight:

Corollary 4.17. The problem $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$, where Q ranges over CQs and Σ over sets of linear TGDs, is PSpace-hard.

Corollary 4.18. The problem $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$, where Q ranges over CQs and Σ over sets of FGTGDs without constants, is 2ExpTime-hard.

4.3. Summary for Positive Query Implication.

The main results on positive query implication are highlighted in the table below.

Background Theory Σ	PQI data complexity	PQI combined complexity	$\exists\mathsf{PQI}$
NoConst Linear TGD	ExpTime-cmp Thm. 4.2 / Thm 4.6	2ExpTime-cmp Thm. 4.1 / Thm 4.8	PSpace-cmp Thm. 4.15 / Cor 4.17
NoConst FGTGD	ExpTime-cmp Thm. 4.2 / Thm 4.6	2ExpTime-cmp Thm. 4.1 / Thm 4.8	2ExpTime-cmp Cor. 4.11/Cor 4.18
NoConst Disj. Linear TGD	ExpTime-cmp Thm. 4.2 / Thm 4.6	2ExpTime-cmp Thm. 4.1 / Thm 4.8	undecidable Thm. 4.12
Linear TGD & FGTGD & GNFO	ExpTime-cmp Thm. 4.2 / Thm. 4.6	2ExpTime-cmp Thm. 4.1 / Thm. 4.8	undecidable Cor. 4.14

5. Negative Query Implication

5.1. Instance-level problems.

Here we analyze the complexity of the problem $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$. As in the positive case, we begin with an upper bound that holds for a very rich class of background theories, which go far beyond referential constraints (and FGTGDs).

Theorem 5.1. The problem $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$, as Q ranges over Boolean UCQs and Σ over sets of GNFO sentences, has 2ExpTime combined complexity, ExpTime data complexity, and it 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:

$$\begin{aligned}\phi^{\mathsf{NQItoGNF}}_{Q,\Sigma,\mathbf{S},\mathcal{V}} &= Q \ \wedge\ \Sigma\ \wedge \\ &\bigwedge_{R\in\mathbf{S}_v}\Big(\bigwedge_{R(\bar a)\in\mathcal{V}} R(\bar a)\ \wedge\ \forall\bar x\ \big(R(\bar x)\to\bigvee_{R(\bar a)\in\mathcal{V}}\bar x=\bar a\big)\Big)\end{aligned}$$

The data complexity analysis is as in Theorem 4.2, since the formulas agree on the part that varies with the instance. ☐

We can show that this bound is tight if the class of background theories is rich enough. This will follow from our lower bounds for positive query implication problems, since we can show that NQI is at least as difficult as PQI for sentences in a powerful logical language.

Theorem 5.2. For any class of sentences that include connected FGTGDs and for any UCQ Q, $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ reduces in polynomial time to $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}')$. When $Q, \Sigma, \mathbf{S}$ are fixed in the input to this reduction, then $Q', \Sigma', \mathbf{S}'$ are fixed in the output.
Thus, for these background theories, the lower bounds for combined and data complexity given in Theorems 4.6 and 4.8 apply to negative query implications as well.

Proof. We first provide a reduction that works with any class of TGDs 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 $\mathbf{S}'$ is obtained by copying both the visible and the hidden relations from $\mathbf{S}$ 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 frontier-guarded TGD of the form

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

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: $Q' = \mathsf{Good}$ and $\mathcal{V}' = \mathcal{V}$ (in particular, we initialize the visible relation Error with the empty set).

We now verify that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$ iff $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{false}$. Suppose that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$, namely, that there is an $\mathbf{S}$-instance $\mathcal{F}$ such that $\mathcal{F} \not\models Q$, $\mathcal{F} \models \Sigma$, and $\mathsf{Visible}(\mathcal{F}) = \mathcal{V}$. Let $\mathcal{F}'$ be the $\mathbf{S}'$-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 $Q_i(\bar{y}) \wedge \mathsf{Good} \rightarrow \mathsf{Error}$ because $\mathcal{F}$ violates every disjunct $\exists \bar{y}\ Q_i$ of Q. Hence, we have $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{false}$. Conversely, suppose that $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \mathcal{V}') = \mathsf{false}$, namely, that there is an $\mathbf{S}'$-instance $\mathcal{F}'$ such that $\mathcal{F}' \models Q'$, $\mathcal{F}' \models \Sigma'$, and $\mathsf{Visible}(\mathcal{F}') = \mathcal{V}'$. By copying the content of $\mathcal{F}'$ for those relations belong to the schema $\mathbf{S}$, we obtain an $\mathbf{S}$-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 $\mathcal{F}$ does. This shows that $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$.

We observe that the sentences 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 among all atoms. More precisely, we expand the relations of the schema $\mathbf{S}$ 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 the 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 $Q_i(\bar{y}) \wedge \mathsf{Good} \rightarrow \mathsf{Error}$ 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 $\mathbf{S}$, we add the sentence

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

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 attribute and with the visible fact $\mathsf{Check}(a)$. ☐

As mentioned above, combining the above reduction with Theorems 4.6 and 4.8, we get the following hardness results for instance-based NQI.

Corollary 5.3. There are a Boolean UCQ Q and a set Σ of IDs over a schema $\mathbf{S}$ for which the problem $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ is ExpTime-hard in data complexity (that is, as $\mathcal{V}$ varies over instances).

Corollary 5.4. The problem $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$, as Σ ranges over sets of connected frontier-guarded TGDs, $\mathbf{S}$ over schemas, Q over conjunctive queries and $\mathcal{V}$ over instances, is 2ExpTime-hard.

Thus far, the negative query implication results have been 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 begin by showing that $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ 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 $\mathcal{V}$:

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

It is clear that the the problem $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ 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 then 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 Σ be a set of IDs over a schema $\mathbf{S}$, Q be a UCQ, and $\mathcal{V}$ be a visible instance such that $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$. Without loss of generality — that is, by adding a dummy visible fact over a visible relation that does not occur in the sentences of the background theory — we can assume that the active domain $\mathsf{adom}(\mathcal{V})$ of $\mathcal{V}$ contains at least one element. The fact that $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$ implies the existence of a full instance $\mathcal{F}$ such that $\mathcal{F} \models \Sigma$, $\mathsf{Visible}(\mathcal{F}) = \mathcal{V}$, and $\mathcal{F} \models Q$. Now take any element $a \in \mathsf{adom}(\mathcal{V})$ 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 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 $h(J) \models \Sigma$. Similarly, we have $h(J) \models Q$. Hence, $h(J)$ is an instance over the active domain of $\mathcal{V}$ that equally witnesses $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$.

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 3. Let $\mathbf{S}$ 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 conjunctive query 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. On the other hand, 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 $\mathsf{NQI}(Q,\Sigma,\mathbf{S},\mathcal{V})$ 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 5.6. For a number k, the problem $\mathsf{NQI}(Q,\Sigma,\mathbf{S},\mathcal{V})$ is said to be *active domain controllable modulo* k if it is equivalent to asking that for every instance $\mathcal{F}$ whose active domain contains at most k elements outside of the active domain of $\mathcal{V}$, if $\mathcal{F}$ satisfies Σ and $\mathcal{V} = \mathsf{Visible}(\mathcal{F})$, then $Q(\mathcal{F}) = \mathsf{false}$.

Theorem 5.7. For any collection Σ of Linear TGDs, the problem $\mathsf{NQI}(Q,\Sigma,\mathbf{S},\mathcal{V})$ is active domain controllable modulo k, where k is the maximal arity of any relation in the schema.

Proof. Let k be the maximal arity of any relation in the schema. 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, we say that two tuples $\vec{t}$ and $\vec{t'}$ of the same length are *equality equivalent* if: $t_i = t_j$ if and only if $t'_i = t'_j$ and for every schema constant c, $t_i = c$ if and only if $t'_i = c$.

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

We fix an extension $\mathbb{D}$ of the active domain of $\mathcal{V}$ that contains k additional fresh values. For each fact $R(\bar{a})$ in $\mathcal{F}$ and each tuple $\bar{b} \in \mathbb{D}^{\mathrm{ar}(R)}$, if $\bar{b}$ and $\bar{a}$ are equality-equivalent and agree on each position whose value is in the active domain of $\mathcal{V}$, 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 $\mathcal{V}$.

Below 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 $i \in \{1,\dots,\mathrm{ar}(R)\}$ such that $\bar{a}(i) \in \mathsf{adom}(\mathcal{V})$. 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 $\bar{u}$. 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 $\mathbb{D}^{\mathrm{ar}(R)}$ equality-equivalent to $\bar{b}$ 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 $\mathsf{NQI}(Q,\Sigma,\mathbf{S},\mathcal{V})$ is active domain controllable modulo k. □

Example 4. As an example of the prior argument, consider a TGD τ

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

and suppose the instance $\mathcal{F}$ has a tuple $R(a,b,b)$ where a_1 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 to prove that NQI problems 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 [AHV95], that is, as finite sets of rules of the form $U(\bar{x}) \leftarrow Q(\bar{x})$ where the variables in $\bar{x}$ are implicitly universally quantified and Q is a conjunctive query 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 $Q(M)$, where Q is any query appearing on the right of a rule with U. The immediate consequence operator is monotone, and 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 $\mathsf{NQI}(Q,\Sigma,\mathbf{S},\mathcal{V})$:

Theorem 5.8. If Q is a Boolean UCQ, Σ a set of linear TGDs (with constants), and $\mathsf{NQI}(Q,\Sigma,\mathbf{S},\mathcal{V})$ is active domain controllable, then $\neg\mathsf{NQI}(Q,\Sigma,\mathbf{S},\mathcal{V})$, 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 $\mathbf{S}$.

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

Below, we implement the function $\neg\mathsf{NQI}(Q,\Sigma,\mathbf{S},-)$ 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 $\mathcal{V}$. Below, we describe the GFP-Datalog program that computes $\mathcal{F}$ starting from $\mathcal{V}$.

The extensional relations are the ones in the visible part $\mathcal{V}$, while the intensional relations are the ones in the hidden part of the schema $\mathbf{S}$, plus an extra intensional relation A

that collects the values in the active domain of $\mathcal{V}$. For each extensional (i.e. visible) relation R and each position $i \in \{1, \ldots, \mathrm{ar}(R)\}$, we add the rule $A(x_i) \leftarrow R(\bar{x})$, which collects all the values of the active domain into the relation A. 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 $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$ — while guaranteeing that the linear TGDs remain valid.

We finally add the rule

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

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 $\neg\mathsf{NQI}(Q, \Sigma, \mathbf{S}, -)$ 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 iff $\mathcal{F}$ satisfies some disjunct $S_1(\bar{z}_1) \wedge \ldots \wedge S_n(\bar{z}_n)$ of the UCQ Q, namely, iff $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$. □

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 5.8 clearly holds:

Theorem 5.9. If Q is a Boolean UCQ, Σ a set of linear TGDs (with constants), and $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ is active domain controllable modulo k, then $\neg\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$, 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 $\mathbf{S}$.

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 5.8. We can likewise get bounds for linear TGDs using Theorem 5.7 and Theorem 5.9.

Corollary 5.10. When Σ ranges over sets of linear TGDs and Q over Boolean UCQs, $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ has data complexity in PTime and combined complexity in ExpTime.

Example 5. Returning to the medical example from the introduction, Example 1, we see that the GFP-Datalog program is quite intuitive: since we have a referential constraint from $\mathsf{Appointment}$ into $\mathsf{Patient}$ and the visible instance does not contain the fact $\mathsf{Patient}(\text{Smith})$, all tuples of the form (Smith, a, d) are removed from the relation $\mathsf{Appointment}$. 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 5.11. The combined complexity of $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$, where Q ranges over UCQs and Σ ranges over IDs, is ExpTime-hard.

Proof. We reduce the acceptance problem for an alternating PSpace Turing machine M to $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$. As in the proof of Theorem 4.6, we assume that the transition function of M maps each universal configuration to a set of exactly 2 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 4.6 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, $y_1 = \perp$, $y_2 = (\perp, q_0)$, $y_3 = \ldots = y_n = \perp$. 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 inclusion dependencies. 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 $S^{\exists}$, $S^{\forall}_1$, and $S^{\forall}_2$, all of arity $2n+2$. We can easily enforce that the first $n+1$ and the last $n+1$ attributes in every tuple of $S^{\exists}$, $S^{\forall}_1$, and $S^{\forall}_2$ describe configurations in $C^{\exists}$ and $C^{\forall}$:

$$\begin{array}{rclcrcl}
S^{\exists}(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\exists}(x, \bar{y}) & \qquad & S^{\exists}(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\exists}(x', \bar{y}') \\
S^{\forall}_1(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\forall}(x, \bar{y}) & & S^{\forall}_1(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\forall}(x', \bar{y}') \\
S^{\forall}_2(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\forall}(x, \bar{y}) & & S^{\forall}_2(x, \bar{y}, x', \bar{y}') & \rightarrow & C^{\forall}(x', \bar{y}') \ .
\end{array}$$

Similarly, we guarantee that every existential (resp., universal) configuration has one (resp., two) successor configuration(s) in $S^\exists$ (resp., $S^\forall_1$ and $S^\forall_2$):

$$\begin{array}{rcl} C^\exists(x,\bar{y}) & \to & \exists\, x'\ \bar{y}'\ S^\exists(x,\bar{y},x',\bar{y}') \\ C^\forall(x,\bar{y}) & \to & \exists\, x'\ \bar{y}'\ S^\forall_1(x,\bar{y},x',\bar{y}') \\ C^\forall(x,\bar{y}) & \to & \exists\, x'\ \bar{y}'\ S^\forall_2(x,\bar{y},x',\bar{y}')\ . \end{array}$$

We now turn to explaining how we can enforce the correctness of the transitions represented in the relations $S^\exists$, $S^\forall_1$, and $S^\forall_2$. Compared to the proof of Theorem 4.6, 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 $N^\exists$, $N^\forall_1$, and $N^\forall_2$ 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 $N^\forall_1$ all the tuples of the form $\big(a_{-1}, (a,q), a_{+1}, a'\big)$ or $\big(a_{-2}, a_{-1}, (a,q), (a_{-1}, q')\big)$, with $a_{-2}, a_{-1}, a_{+1} \in \Sigma$. Accordingly, we introduce the following IDs, for all $1 < i < n$:

$$\begin{array}{rcl} S^\exists(x,\bar{y},x',\bar{y}') & \to & N^\exists(y_{i-1}, y_i, y_{i+1}, y'_i) \\ S^\forall_1(x,\bar{y},x',\bar{y}') & \to & N^\forall_1(y_{i-1}, y_i, y_{i+1}, y'_i) \\ S^\forall_2(x,\bar{y},x',\bar{y}') & \to & N^\forall_2(y_{i-1}, y_i, y_{i+1}, y'_i)\ . \end{array}$$

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

$$\begin{array}{rclcrcl} S^\exists(x,\bar{y},x',\bar{y}') & \to & E(y_1,y'_1) & \qquad & S^\exists(x,\bar{y},x',\bar{y}') & \to & E(y_n,y'_n) \\ S^\forall_1(x,\bar{y},x',\bar{y}') & \to & E(y_1,y'_1) & & S^\forall_1(x,\bar{y},x',\bar{y}') & \to & E(y_n,y'_n) \\ S^\forall_2(x,\bar{y},x',\bar{y}') & \to & E(y_1,y'_1) & & S^\forall_2(x,\bar{y},x',\bar{y}') & \to & E(y_n,y'_n) \end{array}$$

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

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

Let $\mathcal{V}$ be the instance that captures the intended semantics of the visible relations V, C_0, $N^\exists$, $N^\forall_1$, $N^\forall_2$, E, and V_{rej}, The proof that $\mathsf{NQI}(Q,\Sigma,\mathbf{S},\mathcal{V}) = \mathsf{true}$ iff 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 4.6. □

5.2. **Existence problems.** Here we consider the complexity of the schema-level question, $\exists\mathsf{NQI}(Q,\Sigma,\mathbf{S})$. We first show that when the background theories are preserved under disjoint unions (e.g., connected frontier guarded TGDs), 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: the variant of the chase procedure that we introduced in Section 4.2 terminates immediately when initialized with the empty instance

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

Theorem 5.12. If the query Q is monotone and the background theory Σ is preserved under disjoint unions of instances, then $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ iff $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathsf{true}$.

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

Suppose that $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathsf{false}$, namely, that there is an $\mathbf{S}$-instance $\mathcal{F}$ satisfying Σ and Q and such that $\mathsf{Visible}(\mathcal{F}) = \emptyset$. We aim at proving that $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$ for all realizable visible instances $\mathcal{V}$. Let $\mathcal{V}$ be such a realizable instance and let $\mathcal{F}'$ be an $\mathbf{S}$-instance that satisfies Σ and such that $\mathsf{Visible}(\mathcal{F}') = \mathcal{V}$. 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. Since $\mathcal{V} = \mathsf{Visible}(\mathcal{F}') = \mathsf{Visible}(\mathcal{F}'')$, we have $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{false}$. Finally, since $\mathcal{V}$ was chosen in an arbitrary way, this proves that $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S}) = \mathsf{false}$. $\square$

Using the "negative critical instance" result above and Theorem 5.1, we immediately see that $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$ is decidable in 2ExpTime for GNFO sentences that are closed under disjoint unions, and in particular for connected frontier-guarded TGDs. Combining with Corollary 5.10 also gives an ExpTime bound for linear TGDs. In fact, we can improve this upper bound by observing that the NQI problem over the empty visible instance reduces to classical Open-World Query answering:

Proposition 5.13. For any Boolean CQ Q, $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \emptyset)$ holds iff $\mathsf{OWQ}(Q', \Sigma, \mathsf{CanonInst}(Q))$ holds, where

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

and $\mathsf{CanonInst}(Q)$ is the canonical instance of the CQ Q.

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

The proof that $\mathsf{OWQ}(Q', \Sigma, \mathsf{CanonInst}(Q)) = \mathsf{true}$ implies $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathsf{true}$ follows symmetric arguments. $\square$

We know from previous results [BGO10] that OWQ for Boolean UCQs and linear TGDs is in PSpace. From the above reduction, we immediately get that the problem $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \emptyset)$, and hence (by Theorem 5.12) the problem $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$, for a set of linear TGDs is also in PSpace.

Corollary 5.14. The problem $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$, as Q ranges over Boolean UCQ and Σ over sets of linear TGDs, is in PSpace.

Matching lower bounds for $\exists\mathsf{NQI}$ come by a converse reduction from Open-World Query answering.

To prove this reduction, we first provide a characterization of the NQI problem over the empty visible instance, which is based, like Proposition 4.5, on our chase procedure:

Proposition 5.15. If Q is a Boolean CQ and Σ is a set of TGDs and EGDs without constants over a schema $\mathbf{S}$, then $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathsf{true}$ iff either Q contains a visible atom, or it does not and in this case $\mathrm{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathsf{CanonInst}(Q)) = \emptyset$.

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

Conversely, suppose that $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \emptyset) = \mathsf{false}$. This means that there is an $\mathbf{S}$-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 4.4, letting $\mathcal{F}_0 = \mathsf{CanonInst}(Q)$, we get the existence of an instance K in $\text{Chases}_{\mathsf{vis}}(\Sigma, \mathbf{S}, \mathsf{CanonInst}(Q))$. □

As in the positive case, the upper bounds are tight:

Theorem 5.16. $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$ is 2EXPTIME-hard as Q ranges over Boolean CQs and Σ over sets of connected FGTGDs.

Theorem 5.17. $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$ is PSPACE-hard as Q ranges over Boolean CQs and Σ over sets of linear TGDs.

The first theorem will be proven by reducing the open-world query answering problem to $\exists\mathsf{NQI}$, and then applying a prior 2EXPTIME-hardness result from Calì et al. [CGK13]. The PSPACE lower bound will be shown by a reduction from the implication problem for IDs, shown PSPACE-hard by Casanova et al. [CFP84].

We begin with the reduction from Open-World Query answering:

Proposition 5.18. There is a polynomial time reduction from the Open-World Query answering problem over a set of connected FGTGDs without constants and a connected Boolean CQ to an $\exists\mathsf{NQI}$ problem over a set of connected FGTGDs without constants and a Boolean CQ.

Proof. Consider the Open-World Query answering problem over a schema $\mathbf{S}$, a set Σ of sentences without constants and closed under disjoint union, a Boolean CQ Q, and an $\mathbf{S}$-instance $\mathcal{F}$. We reduce this problem to an $\exists\mathsf{NQI}$ problem over a new schema $\mathbf{S}'$, a new set of sentences Σ', and a new Boolean CQ Q'. The schema $\mathbf{S}'$ is obtained from $\mathbf{S}$ by adding a relation Good of arity 0, which is assumed to be the only visible relation in $\mathbf{S}'$. 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 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 5.12, $\exists\mathsf{NQI}(Q', \Sigma', \mathbf{S}') = \mathsf{true}$ iff $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \emptyset) = \mathsf{true}$. Thus, it remains to show that $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \emptyset) = \mathsf{true}$ iff $\mathsf{OWQ}(Q, \Sigma, \mathcal{F}) = \mathsf{true}$.

By contraposition, suppose that $\mathsf{OWQ}(Q, \Sigma, \mathcal{F}) = \mathsf{false}$. This means that there is a $\mathbf{S}$-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 $\mathbf{S}'$, 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 $\mathbf{S}'$-instance $\mathcal{F}'$ thus witnesses the fact that $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \emptyset) = \mathsf{false}$.

Conversely, suppose that $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \emptyset) = \mathsf{false}$. Recall that the sentences in Σ' do not use constants and Q' contains no visible facts. We can thus apply Proposition 5.15 and derive $\mathrm{Chases}_{\mathsf{vis}}(\Sigma', \mathbf{S}', \mathsf{CanonInst}(Q')) \neq \emptyset$. Note that $\mathsf{CanonInst}(Q')$ is clearly isomorphic to the original instance $\mathcal{F}$. In particular, there is an instance K in $\mathrm{Chases}_{\mathsf{vis}}(\Sigma', \mathbf{S}', \mathsf{CanonInst}(Q'))$ 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 $\mathbf{S}$, witnesses the fact that $\mathsf{OWQ}(Q, \Sigma, \mathcal{F}) = \mathsf{false}$. □

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

We are now ready to prove Theorem 5.16, namely, the 2ExpTime-hardness of the problem $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$, where Q ranges over Boolean CQs and Σ ranges over sets of connected FGTGDs.

Proof of Theorem 5.16. Theorem 6.2 of Calì et al. [CGK13] shows 2ExpTime-hardness of open-world query answering for FGTGDs. An inspection of the proof shows that only connected FGTGDs are required. Thus, the theorem follows immediately from Proposition 5.18. □

We now turn towards proving Theorem 5.17, namely, the PSpace lower bound for $\exists\mathsf{NQI}$ under linear TGDs. Recall that the reduction in Proposition 5.18 does not preserve smaller classes of sentences, such as linear TGDs. We thus prove the theorem using a separate reduction.

Proof of Theorem 5.17. We reduce from the implication problem for inclusion dependencies (IDs), which is known to be PSpace-hard from Casanova et al. [CFP84]. 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 $\bar{x}_\star$ and $\bar{y}$. We denote by $F(\delta)$ the sequence of variables shared between $\bar{x}_\star$ and $\bar{z}_\star$ and m the length of this vector. Note that we annotated relations and variables in δ with the subscript $_\star$ in order to make it clear when refer later to these particular objects.

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

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

we introduce a corresponding 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 distinct from the variables in $\bar{x}$. We also add the sentences

$$\begin{aligned} S_\star(\bar{x}_\star) \;&\rightarrow\; S'_\star(\bar{x}_\star, F(\delta)) \\ T'_\star(\bar{z}_\star, F(\delta)) \;&\rightarrow\; \mathsf{Good} \end{aligned}$$

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 R to R' and duplicates the attributes is not an ID, but is still a linear TGD. The query of our $\exists\mathsf{NQI}$ problem is defined as

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

The sentences that we just defined are preserved under disjoint unions. Thus, by Theorem 5.12, we know that $\exists\mathsf{NQI}(Q', \Sigma', \mathbf{S}') = \mathsf{true}$ iff $\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \emptyset) = \mathsf{true}$. Below, we prove that the latter holds iff 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 $\mathbf{S}'$ 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 $\mathbf{S}'$-instance $\mathcal{F}'$ with 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 it did, 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. This however would contradict the hypothesis that $\mathcal{F}'$ has empty visible part.

In the other direction, suppose that the implication fails and consider a witness $\mathbf{S}$-instance $\mathcal{F}$ that contains the fact $S_\star(\bar{x}_\star)$ but not the corresponding $T_\star$ fact. We create a full $\mathbf{S}'$-instance $\mathcal{F}'$ with empty visible part where Q' holds, thus showing that $\exists\mathsf{NQI}(Q', \Sigma', \mathbf{S}', \emptyset) = \mathsf{false}$. 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 the 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). We do not know whether $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$ is PSPACE-hard even for background theories consisting of IDs.

We can show that the connectedness requirement is critical for decidability:

Theorem 5.19. The problem $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$ is undecidable as Q ranges over Boolean CQs and Σ over sets of FGTGDs.

Proof. We give a reduction from the *model conservativity problem* for $\mathcal{EL}$ TBoxes, which is shown undecidable in [LW07]. 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 [LW07] shows that this problem is undecidable for both finite instances and arbitrary instances.

We reduce the above problem to the complement of $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$, 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 $\Sigma = \{\phi_1, \mathsf{Good} \rightarrow \phi_2\}$, 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 dependency of ϕ_2 (note that this makes the dependency unconnected). Finally, consider the query $Q = \mathsf{Good}$. We have that $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ iff there is an $\mathbf{S}_1$-instance $\mathcal{V}$ satisfying ϕ_1, none of whose $\mathbf{S}_2$-expansions satisfies ϕ_2. □

5.3. **Summary for Negative Query Implication.** A summary of results on negative implication is below. We notice that the decidable cases are orthogonal to those for positive implications. Note also that unlike in the positive cases, we have tractable cases for data complexity.

background theory Σ	NQI data complexity	NQI combined complexity	$\exists$NQI
Linear TGD	In PTime Cor. 5.10	ExpTime-cmp Cor. 5.10 / Thm. 5.11	PSpace-cmp Cor. 5.14 / Thm. 5.17
Conn. Disj. FGTGD	ExpTime-cmp Thm. 5.1 / Thm. 5.2	2ExpTime-cmp Thm. 5.1 / Thm. 5.2	2ExpTime-cmp Thm. 5.12/Thm. 5.16
FGTGD & GNFO	ExpTime-cmp Thm. 5.1 / Thm. 5.2	2ExpTime-cmp Thm. 5.1 / Thm. 5.2	undecidable Thm. 5.19

6. Extensions and special cases

We present some results concerning natural extensions of the framework.

Non-Boolean queries. Throughout this work we have restricted to queries to be given as sentence. 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. E.g. $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ 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 $\mathbf{S}$ satisfying the background theory Σ and having visible part $\mathcal{V}$. As usual, the schema-level problem $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$ (resp. $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$) for a non-Boolean query Q amounts at deciding whether there is a realizable visible instance $\mathcal{V}$ witnessing $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ (resp. $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$).

We show that *all of our results carry over to the non-Boolean case.* Since the lower-bounds for Boolean problems are clearly inherited by the non-Boolean ones, we focus on arguing that the upper bounds carry over.

All the complexity upper bounds for the instance-level problem carry over in a rather simple way. For example, given $\mathbf{S}$, Σ, $\mathcal{V}$ 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 $\mathsf{PQI}(Q_{\bar{t}}, \Sigma, \mathbf{S}, \mathcal{V})$, 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, while 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 5.7 extends with the same statement and proof, while the argument in Theorem 5.8 is easily extended to show that there is a GFP-Datalog program that returns the complement of $\mathsf{NQI}(Q, \Sigma, \mathbf{S})$ within the active domain.

The complexity results for $\exists$PQI also generalize to the non-Boolean case: we can revise Theorem 4.9 to state $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S}) = \mathsf{true}$ iff there is a positive query implication for the tuple $(a, \dots, a)$ and the instance $\mathcal{V}_{\{a\}}$. For $\exists$NQI, we can extend Theorem 5.12 to show that for logical sentences preserved under disjoint union, if there is a positive query implication involving some visible instance $\mathcal{V}$ and a tuple $\bar{t}$, then there is one involving the empty instance and the same tuple $\bar{t}$. From this it follows that the complexity bounds for $\exists\mathsf{NQI}$ carry over to the non-Boolean case.

Beyond unions of conjunctive queries. 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 $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$, as Q ranges over BCCQs, subsumes both $\mathsf{PQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ and $\mathsf{NQI}(Q, \Sigma, \mathbf{S}, \mathcal{V})$ 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 $\exists\mathsf{PQI}$ and $\exists\mathsf{NQI}$ we immediately run into problems:

Theorem 6.1. The problem $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$ for a Boolean combination Q of Boolean CQs is undecidable, even when the sentences in the background theory are IDs. The same holds for $\exists\mathsf{NQI}(Q, \Sigma, \mathbf{S})$.

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 $\exists\mathsf{PQI}(Q, \Sigma, \mathbf{S})$. Again, for convenience we deal with the infinite variant of the problem. The idea will be that the visible instance witnessing $\exists\mathsf{PQI}$ 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 $\mathsf{HorChallenge}(x, y)$, with an associated sentence in the background theory

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

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

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

Note that this can only happen if the relation $\mathsf{HorChallenge}$ 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 $\mathsf{VertChallenge}$ 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 $\mathsf{HorFuncChallenge}(x, y, y')$ and a background theory sentence

$$\begin{aligned} \mathsf{HorFuncChallenge}(x, y, y') &\ \rightarrow \ E_H(x, y) \\ \mathsf{HorFuncChallenge}(x, y, y') &\ \rightarrow \ E_H(x, y') \ . \end{aligned}$$

We also add to the query Q the conjunct:

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

We claim that if there is a visible instance witnessing $\exists\mathsf{PQI}$, 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 $\mathsf{HorFuncChallenge}$, 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 $\mathsf{HorFuncChallenge}$, the conjunct above is clearly satisfied by its first disjunct. If there is some tuple (x, y, y') in $\mathsf{HorFuncChallenge}$, 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 $\mathsf{ConfChallenge}(x, y, z, u, v)$ with the following sentences in the background theory:

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

A potential tuple in $\mathsf{ConfChallenge}(x, y, z, u, v)$ 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

$$\big(\ \neg\exists\ x\ y\ z\ u\ v\ \mathsf{ConfChallenge}(x, y, z, u, v)\ \big)\ \vee\ \big(\ \exists\ x\ y\ z\ u\ \mathsf{ConfChallenge}(x, y, z, u, u)\ \big)$$

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. □

The case of conjunctive query 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 instance-level problems are still well-behaved, 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 $V(\vec{x}) \rightarrow \bigvee_{\vec{a} \in V(\mathcal{V})} \vec{x} = \vec{a}$. Thus the "disjunctive chase" of $\mathcal{V}$ with these dependencies will terminate, since after the first round (where we fire Σ_1 dependencies), no new elements will be created.

The decidability of the $\exists\mathsf{PQI}$ problem follows immediately from these observations and Theorem 4.9, which applies to background theories capturing CQ-view definitions. In contrast, for the $\exists\mathsf{NQI}$ problem we prove that

Theorem 6.2. The $\exists\mathsf{NQI}$ problem under background knowledge given as CQ-view definitions is undecidable.

Proof. As in earlier undecidability results, such as Theorem 4.12, we will give the proof for the unrestricted version of the problem, which asserts the existence of an instance with a 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 $H, V \subseteq T \times T$. In order to match the unrestricted version of $\exists\mathsf{NQI}$, 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:

$$\begin{aligned} E_H(x,y) \quad &\iff \quad E'_H(x,y) \\ E_V(x,y) \quad &\iff \quad E'_V(x,y) \ . \end{aligned}$$

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 $\mathsf{HorFuncChallenge}$ of arity 4, and a visible relation $\mathsf{ErrHorFun}$ of arity 3 with the associated CQ-view definition

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

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

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

We explain how the subquery $Q_{\mathsf{HorFuncChallenge}}$ enforces that every element has at most one successor in the relation E_H.

Suppose that $\exists\mathsf{NQI}(Q_{\mathsf{HorFuncChallenge}}, \Sigma, \mathbf{S}) = \mathsf{true}$, namely, that there exists an $\mathbf{S}_v$-instance $\mathcal{V}$ such that $\mathsf{NQI}(Q_{\mathsf{HorFuncChallenge}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. The visible relation $\mathsf{ErrHorFun}$ 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 (note that $\mathcal{V}$ is clearly realizable). Moreover, as $\mathsf{ErrHorFun}$ is empty in $\mathcal{V}$, 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 the sentences in Σ and also the query $Q_{\mathsf{HorFuncChallenge}}$, thus contradicting $\mathsf{NQI}(Q_{\mathsf{HorFuncChallenge}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

For the converse direction, we aim at proving that there is a negative query implication on $Q_{\mathsf{HorFuncChallenge}}$ for those instances that encode valid tilings and are realizable. More precisely, we consider a visible instance $\mathcal{V}$ in which the relation E_H is a function and the relation $\mathsf{ErrHorFun}$ is empty (note that the latter condition on $\mathsf{ErrHorFun}$ is safe, in the sense that the considered instance $\mathcal{V}$ could be obtained from a valid tiling and, being realizable, could be used to witness a negative query implication). 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': $\mathsf{HorFuncChallenge}(x, y, x, y')$, $E_H(x, y)$, $E_V(x, y')$. On the other hand, $\mathcal{F}$ 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. We introduce two nullary relations $\mathsf{HorEmptyError}$ and $\mathsf{HorEmptyHiddenError}$, where the former is visible and the latter is hidden, and we constrain them via the CQ-view definition

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

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

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

Below, we show how this enforces non-emptiness of E_H.

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 $\mathsf{HorEmptyError}$ 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 $\mathsf{HorEmptyHiddenError}$ 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 $\mathsf{HorEmptyError}$ is empty (again, such an instance can be obtained from a valid tiling of the infinite grid and thus can be used to witness a negative query implication). In any full instance that agrees with $\mathcal{V}$ on the visible part, $\mathsf{HorEmptyHiddenError}$ must agree with $\mathsf{HorEmptyError}$, and hence must be empty. This implies that the query $Q_{\mathsf{HorEmptyError}}$ is violated, whence $\mathsf{NQI}(Q_{\mathsf{HorEmptyError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

Besides requiring that E_H and E_V are non-empty, we must also guarantee that for every pair $(x, y) \in E_H$ (resp., $(x, y) \in E_V$), there is a pair $(y, z) \in E_V$ (resp., $(y, z) \in E_H$). Note that once we have performed 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 $\mathsf{HorSuccError}$ of arity 0, and a hidden relation $\mathsf{HorSuccHiddenError}$ of arity 1. The associated CQ-view definition is

$$\mathsf{HorSuccError} \;\rightarrow\; \exists\, x\, y\, z\; E_H(x, y) \;\wedge\; \mathsf{HorSuccHiddenError}(y) \;\wedge\; E_V(y, z) \;.$$

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

$$Q_{\mathsf{HorSuccError}} \;=\; \mathsf{HorSuccError} \;\vee\; \big(\, \exists\, x\, y\; E_H(x, y) \;\wedge\; \mathsf{HorSuccHiddenError}(y)\,\big) \;.$$

We show how this enforces the desired property.

Suppose that there is a visible instance $\mathcal{V}$ such that $\mathsf{NQI}(Q_{\mathsf{HorSuccError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. First, observe that the visible relation $\mathsf{HorSuccError}$ must be empty, as otherwise all extensions of $\mathcal{V}$ would satisfy $Q_{\mathsf{HorSuccError}}$. 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 $\mathsf{HorLabelHiddenError}(y)$. This full instance has $\mathcal{V}$ as visible part and satisfies the sentences in the background theory and the query $Q_{\mathsf{HorSuccError}}$. As this contradicts the hypothesis $\mathsf{NQI}(Q_{\mathsf{HorSuccError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$, 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 $\mathsf{HorSuccError}$ is empty. In any full instance that agrees with $\mathcal{V}$ on the visible part, $\mathsf{HorSuccError}$ must be the same as $\exists\, x\; y\; z\; E_H(x, y) \;\wedge\; \mathsf{HorSuccHiddenError}(y) \;\wedge\; E_V(y, z)$. In particular, because every node has both a successor in E_H and a successor in E_V, this implies that the hidden relation $\mathsf{HorSuccHiddenError}$ cannot contain the node y, for any pair $(x, y) \in E_H$. Hence the query $Q_{\mathsf{HorSuccError}}$ is necessarily violated, and this proves that $\mathsf{NQI}(Q_{\mathsf{HorSuccError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

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 $\mathsf{HorLabelError}_t$ of arity 0 for each tile $t \in T$ and a hidden relation $\mathsf{HorLabelHiddenError}$ of arity 1. The associated CQ-view definitions are of the form

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

We add as conjunct of our query the following UCQ:

$$Q_{\mathsf{HorLabelError}} \;=\; \bigvee_{t \in T} \exists\, x\; y\; \big(\mathsf{HorLabelError}_t(x, y)\big) \;\vee\; \big(E_H(x, y) \wedge \mathsf{HorLabelHiddenError}(x)\big)\ .$$

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 $\mathsf{NQI}(Q_{\mathsf{HorLabelError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$. For each tile t, the visible relation $\mathsf{HorLabelError}_t$ must be empty, as otherwise all extensions of $\mathcal{V}$ would satisfy $Q_{\mathsf{HorLabelError}}$. 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 $\mathsf{HorLabelHiddenError}$ contains the element x. This instance would then satisfy the query $Q_{\mathsf{HorLabelError}}$, thus contradicting $\mathsf{NQI}(Q_{\mathsf{HorLabelError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

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 the relations $\mathsf{HorLabelError}_t$, with $t \in T$, in this visible instance are empty. Note that such an instance $\mathcal{V}$ is realizable and hence can be obtained from a valid tiling (if there is any) and used as a witness of a negative query implication. In every full instance that agrees with $\mathcal{V}$ and satisfies the background theory, $\mathsf{HorLabelError}_t$ must be the same as $\exists\, x\; y\; \mathsf{HorLabelHiddenError}(x) \wedge E_H(x, y) \wedge U_t(x)$. In particular, because every node is associated with some tile, this implies that the hidden relation $\mathsf{HorLabelHiddenError}$ cannot contain the node x, for any pair $(x, y) \in E_H$. Hence the query $Q_{\mathsf{HorLabelError}}$ is necessarily violated, and this proves that $\mathsf{NQI}(Q_{\mathsf{HorLabelError}}, \Sigma, \mathbf{S}, \mathcal{V}) = \mathsf{true}$.

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

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

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

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

Summing up, if we let Q be the disjunction of all previous queries, we know that $\exists\mathsf{NQI}(Q,\Sigma,\mathbf{S}) = \mathsf{true}$ if and only if there exists a valid tiling of the infinite grid $\mathbb{N}\times\mathbb{N}$. $\square$

7. Conclusions

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 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 works such as [BtCLT16, BtCT16].

References

[AD98] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *PODS*, 1998.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[BBPtC16] M. Benedikt, P. Bourhis, G. Puppis, and B. ten Cate. Querying visible and invisible information. In *LICS*, 2016.

[BBV17] Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. Characterizing definability in decidable fixpoint logics. In *ICALP*, 2017.

[BCK17] Michael Benedikt, Bernardo Cuenca Grau, and Egor V. Kostylev. Source information disclosure in ontology-based data integration. In *AAAI*, 2017.

[BCS15] Vince Bárány, Balder Ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3), 2015.

[BCtCV15] Michael Benedikt, Thomas Colcombet, Balder ten Cate, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *LICS*, 2015.

[Bet53] E. W. Beth. On Padoa's method in the theory of definitions. *Indagationes Mathematicae*, 15, 1953.

[BGO10] V. Bárány, G. Gottlob, and M. Otto. Querying the guarded fragment. In *LICS*, 2010.

[BLMS09] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *IJCAI*, 2009.

[BtCLT16] Michael Benedikt, Balden ten Cate, Julien Leblay, and Efthymia Tsamoura. *Generating plans from proofs: the interpolation-based approach to query reformulation.* Morgan Claypool, 2016.

[BtCO12] V. Bárány, B. ten Cate, and M. Otto. Queries with guarded negation. In *VLDB*, 2012.

[BtCS11] V. Bárány, B. ten Cate, and L. Segoufin. Guarded negation. In *ICALP*, 2011.

[BtCT16] Michael Benedikt, Balder ten Cate, and Efi Tsamoura. Generating plans from proofs. In *TODS*, 2016.

[BtCV14] Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. Effective interpolation and preservation in guarded logics. In *CSL-LICS*, 2014.

[CFP84] M. Casanova, R. Fagin, and C. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. *JCSS*, 28(1):29–59, 1984.

[CGK13] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *JAIR*, 48:115–174, 2013.

[CK90] C.C. Chang and H.J. Keisler. *Model Theory.* North-Holland, 1990.

[CY14] R. Chirkova and T. Yu. Obtaining information about queries behind views and dependencies. *CoRR*, abs/1403.5199, 2014.

[DNR08] Alin Deutsch, Alan Nash, and Jeff Remmel. The chase revisited. In *PODS*, 2008.
[FG10a] W. Fan and F. Geerts. Capturing missing tuples and missing values. In *PODS*, 2010.
[FG10b] W. Fan and F. Geerts. Relative information completeness. *ACM TODS*, 35(4):27, 2010.
[FIS11] E. Franconi, Y. Ibáñez-García, and I. Seylan. Query answering with DBoxes is hard. *ENTCS*, 278:71–84, 2011.
[FKMP05] Ronald Fagin, Phokion G. Kolaitis, Renee J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.
[GB14] M. Guarnieri and D. A. Basin. Optimal security-aware query processing. *PVLDB*, 7(12), 2014.
[GM14] T. Gogacz and J. Marcinkowski. All-instances termination of chase is undecidable. In *ICALP*, 2014.
[GM15] Tomasz Gogacz and Jerzy Marcinkowski. The hunt for a red spider: Conjunctive query determinacy is undecidable. In *LICS*, 2015.
[GP03] G. Gottlob and C. Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comp.*, 183, 2003.
[JK84] D. S. Johnson and A. C. Klug. Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies. *JCSS*, 28(1):167–189, 1984.
[KLWW13] B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artif. Intell.*, 203:66–103, 2013.
[KUB$^+$12] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Query-based data pricing. In *PODS*, 2012.
[LSW12] C. Lutz, I. Seylan, and F. Wolter. Mixing open and closed world assumption in ontology-based data access: Non-uniform data complexity. In *Description Logics*, 2012.
[LSW13] C. Lutz, I. Seylan, and F. Wolter. Ontology-based data access with closed predicates is inherently intractable (sometimes). In *IJCAI*, 2013.
[LSW15] C. Lutz, I. Seylan, and F. Wolter. Ontology-mediated queries with closed predicates. In *IJCAI*, 2015.
[LW07] C. Lutz and F. Wolter. Conservative extensions in the lightweight description logic EL. In *CADE*, 2007.
[MG10] B. Marnette and F. Geerts. Static analysis of schema-mappings ensuring oblivious termination. In *ICDT*, 2010.
[MS07] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. *JCSS*, 73(3):507–534, 2007.
[NSV10] A. Nash, L. Segoufin, and V. Vianu. Views and queries: Determinacy and rewriting. *TODS*, 35(3), 2010.
[One13] Adrian Onet. The chase procedure and its applications in data exchange. In *Data Exchange Intregation and Streams*, 2013.
[Shm93] Oded Shmueli. Equivalence of datalog queries is undecidable. *The Journal of Logic Programming*, 15(3):231 – 241, 1993.
[Var98] M. Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, 1998.
[ZM05] Z. Zhang and A. O. Mendelzon. Authorization views and conditional query containment. In *ICDT*, 2005.

Appendix 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 arity of the relations used in the sentence is fixed, and further certain parameters of the sentence are fixed: the *width*, and the *CQ-rank*.

A doubly-exponential bound on satisfiability of GNFO was proven in the papers where GNFO was introduced [BtCS11, BCS15]. However the argument was by reduction to satisfiability of the guarded fragment. Conversions of GNFO formulas to automata, and comments about what controls their complexity, are implicit in a number of other works [BCtCV15, BtCV14, BBV17]. But the conversions are performed for richer logics than GNFO. This means firstly that they introduce many complications that are unnecessary for GNFO, and secondly 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 arity of the relations, the width, and the CQ-rank is fixed.

We will start in Subsection A.1 explaining the tree-like model property, and in Subsection A.2 giving background on the automaton formalism we use. In Subsection A.3 we show decidability for the case of GNFO without equality and constants, restricting to sentences of a special kind ("normal form"). Finally in Subsection A.4 we extend to full GNFO, with equalities and constants. We close in Subsection A.5 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": that is a structure that 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 Subsection A.4.

We start by describing what these tree codes look like in detail.

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

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

- a unary predicate D_a for all $a \in N_k$
- unary predicates $R_{\vec{a}}$ for all $R \in \sigma$ of arity j and all $\vec{a} \in N_k^j$

Informally, $D_a(v)$ indicates that a is a name in the node v in the tree code, while $R_{\vec{a}}(v)$ indicates that R holds for the elements represented by the names $\vec{a}$ 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 $\Sigma^{\text{code}}_{\sigma,k}$ 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 $\Sigma^{code}_{\sigma,k}$-tree* is a $\Sigma^{\text{code}}_{\sigma,k}$-tree such that every node v satisfies

- $|\text{names}(v)| \leq k$
- for all $R_{\vec{a}} \in \Sigma^{\text{code}}_{\sigma,k}$, if $R_{\vec{a}}(v)$ then $\vec{a} \subseteq \text{names}(v)$;

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 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 $i \in \{0, \ldots, j\}$. We write $[v, a]$ for the equivalence class of a-connected nodes of v. For $\vec{a} = a_1 \ldots a_n$, we often abuse notation and write $[v, \vec{a}]$ for the tuple $[v, a_1], \ldots, [v, a_n]$

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

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

such that for each relation R, we have $R^{\mathrm{decode}(\mathcal{T})}([v_1, a_1], \ldots, [v_j, a_j])$ iff there is $w \in \mathsf{dom}(\mathcal{T})$ such that $R_{\vec{a}}(w)$ holds and $[w, a_i] = [v_i, a_i]$ for all i.

We are now ready to state the result that satisfiable GNFO sentences have k-tree-like models. The original papers on GNFO [BtCS11, BCS15] show that every satisfiable GNFO sentence (even with equality and constants) has a satisfying model with a *tree decomposition* in which each vertex of the tree is associated with k elements of the model. We will not need the definition of tree decomposition here, but it is easy to see (and explained in other works, such as [BBV17]) that structures with such a decomposition have codes of the type given above. Hence we have:

Proposition A.1. [BCS15] Suppose φ is a GNFO sentence without equality and constants 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 [CK90]. Using this fact, one can refine the proof of Proposition A.1 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 T where each node has at most countably many children can be converted to a binary tree code 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 T. This ensures that each node of T now has infinitely many (but still countably many) children. Second, we convert T into a full binary tree: starting from the root, each node u with children $(v_i)_{i \in \mathbb{N}}$ is replaced by the subtree consisting of $v_1, v_2, \ldots$ and new nodes $u_1, u_2, \ldots$ such that the label at each u_i is the same as the label at u, the left child of u_i is v_i and the right child of u_i is u_{i+1}. In other words, instead of having a node u with infinitely many children $(v_i)_{i \in \mathbb{N}}$, we create an infinite spine of nodes with the same label as u, and attach each v_i to a different copy u_i of u on this spine.

A.2. **Automata background.**

A.2.1. *Alternating Büchi automata.* We will consider infinite full binary trees: that is infinite trees in which the outdegree of every vertex is two. We assume a set of unary predicates $A_1 \ldots A_n$ for such 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 $\mathsf{Direction}_2$ 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 of elements from 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, while a conjunction holds if all of its conjuncts hold. We will be interested in positive Boolean combinations over $\mathsf{Direction}_2 \times Q$; these formulas will be used to describe possible moves of the automaton.

We will translate GNFO sentences to a *two-way alternating automaton over infinite trees.* Such an automaton is specified as $(Q, \Sigma, q_0, \delta, \Omega)$, where

- Q is a finite set of states
- Σ is as above
- $q_0 \in Q$ is the *initial state*
- $\delta \in Q \times \mathcal{P}(\Sigma) \to B^+(\mathsf{Direction}_2 \times Q)$ is the *transition relation*

- Ω is an acceptance condition, which we discuss below.

A *run* of the automaton starting at vertex v of a tree $\mathcal{T}$, is another tree t' whose labelling function $\lambda_{t'}$ labels vertices n with a vertex of $\mathcal{T}$ and a state $q \in Q$. We now describe further properties that are required for the run to be *accepting*.

First we require that the root of t' is assigned to (v, q_0). That is, the computation starts at the initial state with the specified vertex v.

Second, we require that the relationship between parent and children labels in t' be consistent with the transition function δ. Suppose a vertex n' of t' is associated by $\lambda_{t'}$ to a vertex n of $\mathcal{T}$ whose predicates correspond to subset S_n, and also to a state q', and let $C_{n'}$ be the children of n' in t'. Then we require that $\lambda_{t'}$ associate each $c' \in C_{n'}$ with a vertex of $\mathcal{T}$ that is either n, a parent of n, or child of n.

Given the above requirement we can associate each child $c'_i \in C_{n'}$ with a direction $d'_i \in \mathsf{Direction}_2$ as well as a state $q'_i \in Q$. Let $P_{n'}$ be the set of pairs (d'_i, q'_i) associated with some child of n'. We require that $P_{n'} \models \delta(q, S)$.

Finally, we require that every branch of t' obeys the acceptance condition Ω. There are a number of different acceptance conditions defined for automata over infinite trees. 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 t', there is a state in Ω that occurs infinitely often. Let $\mathsf{2ABT}^\omega$ denote 2-way alternating tree automata equipped with this Büchi acceptance condition.

Given an automaton $\mathcal{A}$ the *language of* $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of trees $\mathcal{T}$ such $\mathcal{A}$ has an accepting run 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 [Var98] showed that non-emptiness is decidable for $\mathsf{2ABT}^\omega$ in ExpTime (in fact, this was shown for parity automata, which includes Büchi automata).

Theorem A.2 ([Var98])**.** If $\mathcal{A}$ is a $\mathsf{2ABT}^\omega$ automaton $\mathcal{A}$, then it is decidable in ExpTime if $L(\mathcal{A}) \neq \emptyset$. More specifically, the running time is $f(|\mathcal{A}|)^{f(s)}$ where s is the number of states of $\mathcal{A}$ and f is a polynomial independent of $\mathcal{A}$.

Thus if we can convert our satisfiability problem into an emptiness check for a $\mathsf{2ABT}^\omega$ automaton 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 $\mathsf{2ABT}^\omega$ automaton 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 start by working with GNFO formulas φ in a normal form, GN-*normal form*, similar to one introduced in [BCS15]. Throughout this section, we also assume that the formulas *do not use equality or constants.*

A.3.1. *Normal form for GNF.*

We present the normal form that we use.

Formulas φ in GN-normal form can be generated using the following grammar:

$$\begin{aligned}\varphi &::= \bigvee_i \exists\vec{x}\ \bigwedge_j \psi_{ij}\\ \psi &::= \alpha \mid \alpha\wedge\varphi \mid \alpha\wedge\neg\varphi \mid\\ &\qquad \varphi \text{ if } \varphi \text{ has at most one free variable } \mid\\ &\qquad \neg\varphi \text{ if } \varphi \text{ has at most one free variable}\end{aligned}$$

where α is an atomic formula, and in the case of $\alpha\wedge\neg\varphi$ and $\alpha\wedge\varphi$, $\mathsf{Free}(\alpha)\supseteq\mathsf{Free}(\varphi)$. The φ are referred to as *UCQ-shaped formulas*, with each of the disjuncts being a *CQ-shaped formula.*

Note that if φ_i for $i=1\dots n$ are *sentences* in normal form then their conjunction $\bigwedge_i \varphi_i$ is also 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 the free variables of χ. The idea of the normal form is that the grammar generates UCQ-shaped formulas where each conjunct is an answer-guarded subformula.

Later we will see that we can convert arbitrary GNFO formulas to this normal form.

The *width* of a GNFO formula φ in the normal form above is the maximum number of free variables in any subformula.

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

A.3.2. *Automata for GNFO.* We now explain how to construct an automaton for a GNFO sentence φ in normal form without equality.

Specializations. The rough idea will be that an automaton has states for all subformulas of φ – the "subformula closure of φ". The automaton being in a state corresponding to subformula ψ at a vertex v of a tree $\mathcal{T}$ will indicate 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 relevant closure, we need to think more carefully about CQ-shaped formulas, and how they can be satisfied in a tree-like structure. For this, we need to describe *specializations.*

Consider a CQ-shaped normal form GNFO formula

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

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

- select a subset $\vec{y}_0$ of $\vec{y}$ (call variables from $\vec{x}\cup\vec{y}_0$ the *inside variables* and variables from $\vec{y}\setminus\vec{y}_0$ the *outside variables*);
- select a partition $\vec{y}_1,\dots,\vec{y}_s$ 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 the partition element $\vec{y}_j$;
- let χ_0 be the conjunction of the ψ_j using only inside variables, and let χ_i for $i\in\{1,\dots,s\}$ be the conjunction of the ψ_j using outside variables and satisfying $\mathsf{Free}(\psi_j)\subseteq\vec{x}\cup\vec{y}_0\cup\vec{y}_i$;

- set $\rho'(\vec{x},\vec{y}_0)$ to be
$$\chi_0(\vec{x},\vec{y}_0) \wedge \bigwedge_{i\in\{1,\dots,s\}} \exists\vec{y}_i\ \chi_i(\vec{x},\vec{y}_0,\vec{y}_i).$$

Roughly speaking, each specialization of ρ describes a different way that a CQ-shaped formula could be satisfied by elements $\vec{x}$ represented in a node of a tree code. 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 shape formula must be realized in a single node, the atoms must be "homogeneous" with respect to the partition, 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 all the existential quantifiers:

Lemma A.3. Let $\rho(\vec{x}) \in$ GNFO be a CQ-shaped formula $\exists\vec{y}\ \bigwedge_j \psi_j(\vec{x},\vec{y})$. For all structures M and for all specializations $\rho'(\vec{x},\vec{y}_0)$ of ρ, if $M \models \rho'(\vec{a},\vec{b})$, then $M \models \rho(\vec{a})$.

Since a formula is vacuously a specialization of itself, the converse direction is vacuously true. What is more useful is that whenever a formula is realized, it is realized by a specialization that is "simpler" than the original formula it specializes. We say a specialization is *non-trivial* if either χ_0 is non-empty or the partition of the outside variables is non-trivial ($s > 1$). The following result captures the idea that in realizing a formula we need to realize some simpler specialization:

Lemma A.4. Let $\rho(\vec{x}) \in$ GNFO be a CQ-shaped formula $\exists\vec{y}\ \bigwedge_j \psi_j(\vec{x},\vec{y})$. Given a structure M and its tree code $\mathcal{T}$, if there exists a vertex $v \in \mathcal{T}$ that includes names $\vec{a}$ and $M \models \rho([v,\vec{a}])$, then there is a non-trivial specialization $\rho'(\vec{x},\vec{y}_0)$ of ρ and a vertex $w \in \mathcal{T}$ with $\vec{a}$ and additional names $\vec{b}_0$ in its domain such that $[w,\vec{a}] = [v,\vec{a}]$ and $M \models \rho'([w,\vec{a}],[w,\vec{b}_0])$.

The idea behind the lemma is that if the formula holds at a node with certain witnesses for the free variables, we can traverse the nodes of the tree codes preserving all those witnesses, until we arrive at a node w where either some of the witnesses are found locally in w or the witnesses are found in different directions from w. In the first case we have realized a specialization in which χ_0 is non-empty, and in the second case we have realized a specialization in which the partition of the outside variables is non-trivial.

Let $\exists\vec{y}\ \eta(\vec{x},\vec{y})$ be any CQ-shaped GN-normal form formula and $\exists\vec{y}\ \eta(\vec{a},\vec{y})$ be formed by substituting elements a_i from N_k for each free variable x_i in $\exists\vec{y}\ \eta(\vec{x},\vec{y})$. We will write $\mathsf{Spec}(\exists\vec{y}\ \eta(\vec{a},\vec{y}),N_k)$ for the set of all specializations of $\exists\vec{y}\ \eta(\vec{a},\vec{y})$ with elements from N_k substituted for any new inside variables. For convenience in the construction below, each specialization $S \in \mathsf{Spec}(\exists\vec{y}\ \bigwedge_j \psi_j(\vec{a},\vec{y}),N_k)$ will be represented as a set: that is, the specialization $\chi_0(\vec{a},\vec{b}_0) \wedge \bigwedge_{i\in\{1,\dots,s\}} \exists\vec{y}_i\ \chi_i(\vec{a},\vec{b}_0,\vec{y}_i)$ of $\exists\vec{y}\ \bigwedge_{j\in\{1,\dots,r\}} \psi_j(\vec{a},\vec{y})$ is represented as the set:
$$\{\psi_j(\vec{a},\vec{b}_0) : j \in \{1,\dots,r\}\,, \psi_j(\vec{a},\vec{b}_0) \text{ in } \chi_0\} \cup \{\exists\vec{y}_i\ \chi_i(\vec{a},\vec{b}_0,\vec{y}_i) : i \in \{1,\dots,s\}\}.$$
Again, each formula in the set describes how a piece of the CQ-shaped formula is satisfied.

We are now ready to define the notion of subformula we are interested in. Fix some GNFO sentence φ in normal form. The closure $\mathsf{cl}_{\mathsf{GN}}(\varphi,N_k)$ 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, elements of $\mathsf{cl}_{\mathsf{GN}}(\varphi,N_k)$ will be written in the form $\langle\psi,p\rangle$ where ψ is a formula and $p \in \{+,-\}$ is a polarity to indicate whether ψ comes from a positive or negative part of φ (that is, a part under an even or odd number of negations).

Let $\mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$ be the smallest set of formulas containing $\langle \varphi, + \rangle$, $\langle \mathsf{true}, + \rangle$, $\langle \mathsf{true}, - \rangle$, $\langle \mathsf{false}, + \rangle$, $\langle \mathsf{false}, - \rangle$ and satisfying the following closure conditions:

- if $\langle \alpha \wedge \neg\psi, + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \alpha, + \rangle, \langle \psi, - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$;
- if $\langle \alpha \wedge \neg\psi, - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \alpha, - \rangle, \langle \psi, + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$;
- if $\langle \neg\psi, + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \psi, - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$;
- if $\langle \neg\psi, - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \psi, + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$;
- if $\langle \alpha \wedge \psi, + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \alpha, + \rangle, \langle \psi, + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$;
- if $\langle \alpha \wedge \psi, - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \alpha, - \rangle, \langle \psi, - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$;
- if $\langle \bigvee_i \psi_i, + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \psi_i, + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$ for all i;
- if $\langle \bigvee_i \psi_i, - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \psi_i, - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$ for all i;
- if $\langle \exists \vec{y}\ \eta(\vec{a}, \vec{y}), + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \psi', + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$ for all $S \in \mathsf{Spec}(\exists \vec{y}\ \eta(\vec{a}, \vec{y}), N_k)$ and $\psi' \in S$;
- if $\langle \exists \vec{y}\ \eta(\vec{a}, \vec{y}), - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, then $\langle \psi', - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$ for all $S \in \mathsf{Spec}(\exists \vec{y}\ \eta(\vec{a}, \vec{y}), N_k)$ and $\psi' \in S$.

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

Proposition A.5. For every GNFO sentence φ in GN-normal form, signature σ containing relations of φ, and $k \in \mathbb{N}$, there is a $\mathsf{2ABT}^\omega$ $\mathcal{A}_\varphi$ on $\Sigma^{\mathrm{code}}_{\sigma,k}$-trees such that $\mathcal{A}_\varphi$ accepts a consistent $\Sigma^{\mathrm{code}}_{\sigma,k}$-tree $\mathcal{T}$ iff the decoding decode($\mathcal{T}$) satisfies φ. Moreover, the number of states of the automaton is bounded by the size of $\mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, while the overall size and the time needed to construct the automaton is at most $f(|\varphi| \cdot |\mathcal{P}(\Sigma^{\mathrm{code}}_{\sigma,k})|) \cdot |N_k|^{f(\mathrm{width}(\varphi)\,\mathrm{rank}_{\mathrm{CQ}}(\varphi))}$ for some polynomial f independent of φ and k.

The $\mathsf{2ABT}^\omega$ automaton $\mathcal{A}_\varphi$ for φ is defined as follows:

- The state set is $\mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$.
- The initial state is $\langle \varphi, + \rangle$.
- The transition function δ is defined below.
- The set of accepting states consists of all states of the form $\langle \mathsf{true}, + \rangle$, $\langle \mathsf{false}, - \rangle$, $\langle R(\vec{a}), - \rangle$, or $\langle \exists \vec{y}\ \eta(\vec{a}, \vec{y}), - \rangle$.

We now describe the transition function. For τ a collection of symbols in $\Sigma^{\mathrm{code}}_{\sigma,k}$ and $\vec{a}$ a collection of names in N_k, we say that $\vec{a}$ is *represented in* τ if τ includes D_{a_i} for each a_i in $\vec{a}$; thus a vertex v labelled with τ that represents $\vec{a}$ has each a_i in $\vec{a}$ as one of its local names.

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

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

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

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

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

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

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

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

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

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

$$\delta(\langle \neg\psi,+\rangle,\tau) := (\mathsf{Stay},\langle \psi,-\rangle)$$

$$\delta(\langle \neg\psi,-\rangle,\tau) := (\mathsf{Stay},\langle \psi,+\rangle)$$

$$\delta(\langle \alpha\wedge\psi,+\rangle,\tau) := (\mathsf{Stay},\langle \alpha,+\rangle)\wedge(\mathsf{Stay},\langle \psi,+\rangle)$$

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

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

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

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

Lemma A.6. For each $\langle\psi(\vec{a}),+\rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi,N_k)$, $\psi(\vec{x})$ holds in decode($\mathcal{T}$) with valuation $[v,\vec{a}]$ for $\vec{x}$ if and only if the automaton above accepts when launched in $\mathcal{T}$ from vertex v with initial state $\langle\psi(\vec{a}),+\rangle$.

Likewise, for each $\langle\psi(\vec{a}),-\rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi,N_k)$, $\psi(\vec{x})$ does not hold in decode($\mathcal{T}$) with valuation $[v,\vec{a}]$ for $\vec{x}$ if and only if the automaton above accepts when launched in $\mathcal{T}$ from vertex v with initial state $\langle\psi(\vec{a}),-\rangle$.

Proof. The lemma is proven by structural induction. The base cases are simple to observe by construction. Lemmas A.3 and A.4 are utilized in the inductive case for CQ-shaped formulas. ☐

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

Lemma A.7. Let $\varphi \in$ GNFO in normal form, and let $k \in \mathbb{N}$. Then $|\mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)| \leq f(|\varphi|) \cdot |N_k|^{f(\text{width}(\varphi)\,\text{rank}_{\mathsf{CQ}}(\varphi))}$ for some polynomial function f independent of φ and k.

Proof. Let $w = \text{width}(\varphi)$ and $r = \text{rank}_{\mathsf{CQ}}(\varphi)$.

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. a CQ-shaped formula without projection) only contribute actual subformulas of φ to the closure set. However, the specializations of a CQ-shaped subformula η with existential quantification 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, this means that the overall size of the closure set is at most $|\varphi| \cdot 2^r \cdot |N_k|^w$. □

Let $w = \text{width}(\varphi)$ and $r = \text{rank}_{\mathsf{CQ}}(\varphi)$. Since the width and CQ-rank are bound by the size of the formula, this means that the size of the closure set $\mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, and hence the number of states of the automaton $\mathcal{A}_\varphi$, is at most exponential in the size of the formula. But it is polynomial when the maximal arity, width, and CQ-rank are fixed.

The size of $\mathcal{P}(\Sigma^{\text{code}}_{\sigma,k})$ is at most $2^{|\sigma| \cdot |N_k|^{\text{ar}\,\sigma}}$, 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 $2^w \cdot |N_k|^w \cdot w^w \cdot |\mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)|$. In particular, note that the transition function formula for a CQ-shaped formula ψ respects this bound since $|\mathsf{Spec}(\psi, N_k)|$ 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 $S \in \mathsf{Spec}(\psi, N_k)$ is of size at most $|\mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)|$.

This means that the size of the transition function is linear in $|Q| \cdot |\mathcal{P}(\Sigma^{\text{code}}_{\sigma,k})| \cdot 2^w \cdot |N_k|^w \cdot w^w \cdot |Q|$. 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 is of size at most doubly exponential in the size of φ, but singly exponential when the maximal arity, width, and CQ-rank is fixed.

From an automaton to decidability. We are now almost done with our satisfiability procedure. Combining Proposition A.5 with Proposition A.1, we see that φ is satisfiable if and only if there is a consistent k-tree code that satisfies $\mathcal{A}_\varphi$, where $k = \text{width}(\varphi)$.

Recall that a consistent $\Sigma^{\text{code}}_{\sigma,k}$-tree is just an arbitrary $\Sigma^{\text{code}}_{\sigma,k}$-tree such that every node v satisfies $|\text{names}(v)| \leq k$ and for all $R_{\vec{a}} \in \Sigma^{\text{code}}_{\sigma,k}$, if $R_{\vec{a}}(v)$ then $\vec{a} \subseteq \text{names}(v)$. It is straightforward to see that there is a $\mathsf{2ABT}^\omega$ automaton $\mathcal{A}_{\text{consistent}}$ that accepts exactly the trees that are consistent in the above sense. The size of $\mathcal{A}_{\text{consistent}}$ 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 two — an initial state and a "rejection" state representing a violation of consistency.

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

Theorem A.8. 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.

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

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

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

Constants. To handle constants requires more effort, since constants may have non-trivial equalities. One route to decidability, taken in [BCS15], is to reduce satisfiability of GNFO with equality and constants to satisfiability without constants. The idea of the reduction in [BCS15] 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 $\mathsf{Const}(\sigma)$ 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 $R(\dots c \dots)$ holding in a vertex, where c is a constant, and we also have an equality fact $c = d$ then we have the fact $R(\dots d \dots)$.

We can extend $\mathcal{A}_{\text{consistent}}$ 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 $\mathsf{Const}(\sigma)$, we let $\mathsf{Const}(\sigma)_{=,\mathcal{T}}$ be the equivalence classes of constants under the equality relation in $\mathcal{T}$. The decoding decode($\mathcal{T}$) is now the σ-structure with universe

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

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

We further claim the following extension of Proposition A.1

Proposition A.9. If a GNFO sentence φ of width k, possibly using equality and constants, is satisfiable, then φ 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 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 $R(x_1 \dots x_n)$ 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.1 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 $N_k \cup \mathsf{Const}(\sigma)$ 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 $\langle \mathsf{false}, + \rangle$, 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 $\langle \mathsf{true}, + \rangle$.
- 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 $\langle \mathsf{true}, + \rangle$. Otherwise it ensures rejection by switching to state $\langle \mathsf{false}, + \rangle$.

That is, for a name $a \in N_k$ and for constants $c, d \in \mathsf{Const}(\sigma)$, we have transitions:

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

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

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

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

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

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

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

Using these modifications, we can now extend Lemma A.6:

Lemma A.10. For each $\langle \psi(\vec{a}, \vec{c}), + \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, $\psi(\vec{x}, \vec{y})$ holds in decode($\mathcal{T}$) at vertex v with valuation $[v, \vec{a}]$ for $\vec{x}$ and constants $c_1 \dots c_l$ for $\vec{y}$ if and only if the automaton above accepts when launched in $\mathcal{T}$ from vertex v with initial state $\langle \psi(\vec{a}, \vec{c}), + \rangle$.

Likewise, for each $\langle \psi(\vec{a},\vec{c}), - \rangle \in \mathsf{cl}_{\mathsf{GN}}(\varphi, N_k)$, $\psi(\vec{x},\vec{y})$ does not hold in 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 $\langle \psi(\vec{a},\vec{c}), - \rangle$.

Recall that the proof of Lemma A.6 worked by induction on ψ. In the proof we first need to consider base cases for equality. For example, suppose $x_1 = x_2$ holds in 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 $\langle \mathsf{true}, + \rangle$, and will accept. The converse direction is similar.

On the other hand, suppose $x_1 = x_2$ holds in 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 $R(x_1, \ldots x_j, y_1 \ldots y_l)$ holds in decode($\mathcal{T}$) with valuation $x_i = [v, a_i], y_i = [c_i]_{=,\mathcal{T}}\}]$ for local names $\vec{a}$ and constants $\vec{c}$. By definition of our decoding, along with the congruence closure of the codes, this means that we must have a fact $R([v, \vec{a}], \vec{c})$ 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.11. 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 $\mathcal{A}_{\text{consistent}}$ for checking consistency, forming an automaton $\mathcal{A}'_\varphi$ accepting the intersection of $\mathcal{A}_{\text{consistent}}$ with $\mathcal{A}_\varphi$, and checking non-emptiness of $\mathcal{A}'_\varphi$.

Thus we obtain the main result, which immediately implies Theorem 4.3 n the body of the paper:

Theorem A.12. 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. Additional remarks: relationship to bounds for general GNFO.

We note that the previous result to allows us to re-prove the bounds for satisfiability of GNFO sentences that are not in normal form from [BtCS11, BCS15]. We include this only because it might be useful to have a self-contained presentation of the GNFO to automate 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.

Proposition A.13. Let ψ be a GNFO formula with $m = |\psi|$. We can construct a sentence $\mathsf{convert}(\psi)$ in GN-normal form equivalent to ψ such that

- $|\mathsf{convert}(\psi)| \leq 2^{f(m)}$,
- $\mathrm{width}(\mathsf{convert}(\psi)) \leq m$,
- $\mathrm{rank}_{\mathsf{CQ}}(\mathsf{convert}(\psi)) \leq m$,
- $|\mathsf{cl}_{\mathsf{GN}}(\mathsf{convert}(\psi), N_m)| \leq 2^{f(m)}$.

where f is a polynomial function independent of ψ.

Proof. We proceed by induction on ψ. The output $\mathsf{convert}(\psi)$ is a UCQ-shaped formula in normal form, with the same free variables as ψ.

- If ψ is atomic, then $\mathsf{convert}(\psi) := \psi$.

- Suppose $\psi = \alpha \wedge \neg\psi'$ where α is a guard for $\mathsf{Free}(\psi')$. Then $\mathsf{convert}(\psi) := \alpha \wedge \neg\mathsf{convert}(\psi')$.

  Similarly for the case of $\neg\psi$ where ψ has at most one free variable.
- Suppose $\psi = \exists y\ \psi'$. If $\mathsf{convert}(\psi')$ is a UCQ-shaped formula $\bigvee_i \exists\vec{z}_i\ \bigwedge_j \psi_{ij}$, then $\mathsf{convert}(\psi) := \bigvee_i \exists y\vec{z}_i\ \bigwedge_j \psi_{ij}$.
- Suppose $\psi = \psi_1 \vee \psi_2$. Then $\mathsf{convert}(\psi)$ is the UCQ-shaped formula $\mathsf{convert}(\psi_1) \vee \mathsf{convert}(\psi_2)$.
- Suppose $\psi = \psi_1 \wedge \psi_2$. If ψ_1 and ψ_2 are answer-guarded, e.g., $\psi_1 = \alpha_1 \wedge \psi_1'$ and $\psi_2 = \alpha_2 \wedge \psi_2'$ with $\mathsf{Free}(\alpha_1) \supseteq \mathsf{Free}(\psi_1')$ and $\mathsf{Free}(\alpha_2) \supseteq \mathsf{Free}(\psi_2')$, then $\mathsf{convert}(\psi) = (\alpha_1 \wedge \mathsf{convert}(\psi_1')) \wedge (\alpha_2 \wedge \mathsf{convert}(\psi_2'))$. The other cases where ψ_1 or ψ_2 have at most one free variable are handled similarly.

  Otherwise, let $\mathsf{convert}(\psi) := \bigvee_{i,i'} \exists\vec{y}_i\vec{y}'_{i'}\ (\chi_i[\vec{y}_i/\vec{x}_i] \wedge \chi'_{i'}[\vec{y}'_{i'}/\vec{x}'_{i'}])$ where $\mathsf{convert}(\psi_1) = \bigvee_i \exists\vec{x}_i\ \chi_i$, $\mathsf{convert}(\psi_2) = \bigvee_{i'} \exists\vec{x}'_{i'}\ \chi'_{i'}$, and the variables in every $\vec{y}_i$ and $\vec{y}'_{i'}$ are fresh.

By Lemma A.7, the size of $\mathsf{cl}_{\mathsf{GN}}(\mathsf{convert}(\psi), N_m)$ is exponential in the size m of ψ. □

Now when we apply the automaton construction of Proposition A.5 to the output, we will get an automaton with state set $\mathsf{cl}_{\mathsf{GN}}(\mathsf{convert}(\psi), N_{|\psi|})$. 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.2 to get a doubly-exponential algorithm for testing satisfiability:

Corollary A.14. [BCS15] There is a 2ExpTime satisfiability testing algorithm for GNFO sentences without equality.