ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Inconsistency-tolerant Query Answering for Existential Rules

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Inconsistency-tolerant Query Answering for Existential Rules / Thomas Lukasiewicz, Enrico Malizia, Maria Vanina Martinez, Cristian Molinaro, Andreas Pieris, Gerardo I. Simari. - In: ARTIFICIAL INTELLIGENCE. - ISSN 0004-3702. - STAMPA. - 307:(2022), pp. 103685.1-103685.39. [10.1016/j.artint.2022.103685]

Availability:
This version is available at: https://hdl.handle.net/11585/879287 since: 2022-03-23

Published:
DOI: http://doi.org/10.1016/j.artint.2022.103685

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

27 April 2024

This is the final peer-reviewed accepted manuscript of:

Lukasiewicz, T., Malizia, E., Martinez, M. V., Molinaro, C., Pieris, A., & Simari, G. I. (2022). Inconsistency-tolerant query answering for existential rules. Artificial Intelligence, 307

The final published version is available online at ***https://dx.doi.org/10.1016/j.artint.2022.103685***

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Inconsistency-tolerant Query Answering for Existential Rules*

Thomas Lukasiewicz[a], Enrico Malizia[b], Maria Vanina Martinez[c,d], Cristian Molinaro[e], Andreas Pieris[f,g], and Gerardo I. Simari[h,i]

[a]Department of Computer Science, University of Oxford, UK
[b]DISI, University of Bologna, Italy
[c]Department of Computer Science, Universidad de Buenos Aires, Argentina
[d]Institute for Computer Science Research (ICC UBA–CONICET), Argentina
[e]DIMES, University of Calabria, Italy
[f]School of Informatics, University of Edinburgh, UK
[g]Department of Computer Science, University of Cyprus, Cyprus
[h]Dept. of Computer Sci. and Engineering, Universidad Nacional del Sur (UNS), Argentina
[i]Institute for Computer Science and Engineering (UNS–CONICET), Argentina

Abstract

Querying inconsistent knowledge bases is an intriguing problem that gave rise to a flourishing research activity in the knowledge representation and reasoning community during the last years. It has been extensively studied in the context of description logics (DLs), and its computational complexity is rather well-understood. Although DLs are popular formalisms for modeling ontologies, it is generally agreed that rule-based ontologies are well-suited for data-intensive applications, since they allow us to conveniently deal with higher-arity relations, which naturally occur in standard relational databases. The goal of this work is to perform an in-depth complexity analysis of querying inconsistent knowledge bases in the case of the main decidable classes of existential rules, based on the notions of guardedness, linearity, acyclicity, and stickiness, enriched with negative (a.k.a. denial) constraints. Our investigation concentrates on three central inconsistency-tolerant semantics: the ABox repair (AR) semantics, considered as the standard one, and its main sound approximations, the intersection of repairs (IAR) semantics and the intersection of closed repairs (ICR) semantics.

1 Introduction

The purpose of an ontology is to provide an explicit specification via an unambiguous language, typically based on logic, of an abstract model of a domain of interest. A relatively recent, and admittedly quite successful, application of ontologies is ontology-based data access (OBDA) [43], which in turn has emerged as an exciting application of knowledge representation and reasoning technologies in information management systems. The goal of OBDA is to facilitate access to data by separating the user from the raw data sources. This is done using an ontology that provides a unified conceptual view of the data, and makes it accessible via queries solely formulated in the vocabulary of the ontology without any knowledge of the actual structure of the data. In addition, the ontology enriches the possibly incomplete data sources with domain knowledge, enabling more complete answers to queries, typically conjunctive queries.

In real-life OBDA scenarios, involving large amounts of data, it is likely that the raw data is inconsistent with the ontology. Since standard ontology languages adhere to the classical first-order logic semantics, inconsistencies are nothing else than logical contradictions. Therefore, the classical inference semantics fails when faced with an inconsistency, since everything follows from a logical contradiction. This demonstrates the need of developing alternative semantics.

© 2022. This manuscript version is made available under the CC BY-NC-ND 4.0 license. The formal publication of this manuscript is available via the DOI: 10.1016/j.artint.2022.103685.

*This paper is a substantially extended and revised version of the papers [34, 37, 39, 40].

There has been a significant effort on the development of inconsistency-tolerant semantics for query answering purposes. Consistent query answering, first developed for relational databases [1], and then generalized as the *ABox repair* (AR) semantics for several DLs [28], is the most widely accepted semantics for querying inconsistent knowledge bases. The AR semantics is based on the idea that an answer is considered to be valid if it can be inferred from each of the repairs of the extensional data set D, i.e., the $\subseteq$-maximal consistent subsets of D. However, obtaining the set of consistent answers under the AR semantics is known to be a hard problem, even for lightweight ontology languages [28]. For this reason, several other semantics have been developed with the aim of approximating the set of consistent answers [3, 9, 28, 38]; the list is by no means exhaustive, and we refer the reader to [4] for a comprehensive survey. We also refer the reader to the related work section (Section 9) for further details on inconsistency-tolerant semantics.

The two main approximations of the AR semantics that are of special interest for the present work are the following:

1. The *intersection of ABox repairs* (IAR) semantics: an answer must be inferred from the intersection of the repairs and the ontology [28].

2. The *intersection of closed repairs* (ICR) semantics: an answer must be inferred from the intersection of the *closure* of the repairs and the ontology [3].

Apart from being natural approximations of the AR semantics, the IAR and ICR semantics are coming with additional advantages of practical relevance. Recent work on explanation in the context of inconsistency-tolerant query answering shows that explanations are much easier to define and compute for the IAR and ICR semantics [7]. Moreover, the IAR and ICR semantics are amenable to preprocessing, since the intersection of the (closed) repairs can be computed offline, and then standard query answering algorithms can be employed online. Indeed, the latter approach has been adopted in the implementation of the IAR semantics [30], while for the ICR semantics, it has been already remarked in [5].

The complexity of query answering under different inconsistency-tolerant semantics, including the ones above, has been extensively studied for a wide spectrum of DLs; see, e.g., [3, 6, 8, 30, 44]. But what about rule-based ontology languages? It is agreed that rule-based ontologies are well-suited for data-intensive applications such as OBDA, since they allow us to conveniently deal with higher-arity relations, which naturally occur in standard relational databases. Therefore, analyzing and understanding the complexity of query answering under inconsistency-tolerant semantics in the presence of rule-based ontologies is a highly relevant task that deserves our attention. This is the main concern of the present work.

Towards this direction, we focus on ontologies modeled using existential rules (also called tuple-generating dependencies), i.e., first-order sentences of the form

$$\forall \bar{x} \forall \bar{y} \, (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \, \psi(\bar{x}, \bar{z})) \,,$$

with ϕ and ψ being conjunctions of atoms, and negative (a.k.a. denial) constraints, which are first-order senetences of the form

$$\forall \bar{x} \, (\phi(\bar{x}) \rightarrow \bot) \,,$$

where ϕ is a conjunction of atoms, and $\bot$ denotes the truth constant false. Note that rules of the above form are also known in the literature as Datalog$^{\pm}$ rules [12]. It is known, however, that query answering under arbitrary existential rules (even without negative constraints) is undecidable (see, e.g., [10]). This has led to an intensive research activity for identifying restrictions on existential rules that lead to decidability. The main decidable paradigms are (i) guardedness [10] (which includes linearity [11]) that is based on the relativization of quantifiers by atomic formulas, (ii) acyclicity [37] that forbids recursion, and (iii) stickiness [13] that forces variables that appear more than once in the left-hand side of an existential rule to be propagated to the right-hand side of the rule. The goal underlying stickiness is to express non-guarded statements without forbidding recursion. The formal definitions of the above paradigms are given in Section 3.

Our main goal is to perform an in-depth complexity analysis of querying inconsistent knowledge bases under the AR, IAR, and ICR semantics, when the ontology is modeled using one of the main decidable classes of existential rules discussed above enriched with negative constraints. Note that we can easily inherit from DLs that our problem in the case of the AR semantics is intractable, even when the ontology and the query are fixed [28]. Thus, another objective of our work, apart from clarifying the complexity landscape, is to understand whether the IAR and ICR semantics reduce the complexity of the problem in question, especially when the ontology and the query are fixed. Our contributions are as follows:

- Before studying inconsistency-tolerant query answering, we first need to understand standard query answering under existential rules without inconsistencies. The latter is well-understood in the case of guardedness, linearity, and stickiness. Surprisingly, query answering under acyclic existential rules has not been explicitly studied before the conference paper [37], which is one of the works on which the present journal paper is based. We show that the problem is NExpTime-complete even if we bound the arity, NP-complete if we fix the ontology, and in AC_0 when both the ontology and the query are fixed; these results are summarized in Proposition 3.3.

- The complete picture concerning inconsistency-tolerant query answering in the case of the AR semantics is given by Theorem 5.1. The main outcome is that the problem is intractable, actually coNP-complete, even when the ontology and the query are fixed. This is not surprising in view of the fact that the problem is already coNP-hard for lightweight DLs such as DL-Lite [28].

- The complete picture for the IAR semantics is given by Theorem 6.1. It turned out that the IAR semantics does not reduce the complexity in the case of guardedness; the only difference is a minor decrease from Π_2^P to Θ_2^P when the ontology is fixed. However, in the case of linear, acyclic, and sticky existential rules, we observed some significant differences: the complexity decreases from Π_2^P to NP when the ontology is fixed, and from coNP to AC_0 when both the ontology and the query are fixed. This is due to a central property known as first-order rewritability. Note that all the complexity results, apart from AC_0, are completeness results.

- The complete picture for the ICR semantics is given by Theorem 7.1. The ICR semantics does not reduce the complexity of our problem, no matter which class of existential rules we consider. We only observed a minor decrease from Π_2^P-complete to Θ_2^P-complete when the ontology is fixed.

- Finally, for the sake of completeness, we consider the central class of full existential rules (i.e., rules without existentially quantified variables), as well as the main extensions of guarded, acyclic, and sticky existential rules that generalize full existential rules; the complete picture for all the inconsistency-tolerant semantics in question is given by Theorem 8.1. It turned out that the analysis performed for the less expressive classes allowed us to easily complete the picture for the more expressive classes of existential rules.

The rest of the paper is organized as follows. The basics on relational databases, conjunctive queries, existential rules, and negative constraints, as well as basic complexity classes, are recalled in Section 2. An overview of query answering under the main decidable classes of existential rules is given in Section 3. The main inconsistency-tolerant semantics for query answering under existential rules are introduced in Section 4. Our complexity results on inconsistency-tolerant query answering w.r.t. the AR, IAR, and ICR semantics are presented in Sections 5, 6, and 7, respectively. In Section 8, we consider the class of full existential rules, as well as the main extensions of guarded, acyclic, and sticky existential rules that generalize full existential rules. A rather comprehensive overview of the main inconsistency-tolerant semantics, which go beyond the AR, IAR, and ICR semantics, that have been proposed and studied during the last decade is given in Section 9. We finally conclude in Section 10 with a brief discussion and directions for future research.

2 Preliminaries

In this section, we recall the basics on relational databases, (unions of) conjunctive queries, ontological query answering (including tuple-generating dependencies[1] and negative constraints), the chase procedure, and the complexity classes encountered in this paper. Throughout the paper, we assume the disjoint countably infinite sets $\mathbf{C}$, $\mathbf{N}$, and $\mathbf{V}$ of *constants*, *(labeled) nulls*, and *variables*, respectively. We also refer to constants, nulls, and variables as *terms*.

2.1 Relational databases and (unions of) conjunctive queries

A *(relational) schema* $\mathbf{S}$ is a finite set of relation symbols (or predicates) with associated arity. We write R/n to denote that the arity of the relation symbol R is $n \geq 0$. A *relational atom* (or simply *atom*) over $\mathbf{S}$ is an expression of the form $R(\bar{t})$, where R is an n-ary relation symbol from $\mathbf{S}$, and $\bar{t}$ is an n-tuple of

[1]Henceforth, as customary in the literature, we adopt the more traditional term tuple-generating dependency instead of existential rule.

terms. An atom is *ground* if it mentions only constants of $\mathbf{C}$. An *instance* over $\mathbf{S}$ is a (possibly infinite) set of atoms over $\mathbf{S}$ that contain constants and nulls, while a *database* over $\mathbf{S}$ is a finite set of ground atoms over $\mathbf{S}$, i.e., a finite instance without nulls. For an instance I, we write $\mathsf{dom}(I)$ for the set of all terms occurring in I.

Consider two sets of terms T and S. A *substitution* from T to S is a function $h\colon T \to S$. The restriction of h to $T' \subseteq T$, denoted $h_{|T'}$, is the function from T' to S such that, for every $t \in T'$, $h_{|T'}(t) = h(t)$. Consider now two sets of atoms A and B. A *homomorphism* from A to B is a substitution h from the set of terms in A to the set of terms in B, i.e., from $\mathsf{dom}(A)$ to $\mathsf{dom}(B)$, such that (i) $t \in \mathbf{C}$ implies $h(t) = t$, and (ii) $R(t_1, \dots, t_n) \in A$ implies $h(R(t_1, \dots, t_n)) = R(h(t_1), \dots, h(t_n)) \in B$.

A *conjunctive query* (CQ) over a schema $\mathbf{S}$ is a formula of the form

$$q(\bar{x}) \ := \ \exists \bar{y} \left(R_1(\bar{z}_1) \wedge \cdots \wedge R_m(\bar{z}_m) \right),$$

where each $R_i(\bar{z}_i)$, for $i \in \{1, \dots, m\}$, is an atom over $\mathbf{S}$ without nulls, each variable occurring in a tuple $\bar{z}_i$ appears either in $\bar{x}$ or $\bar{y}$, and $\bar{x}$ contains all the *free variables* of q. If $\bar{x}$ is empty, then q is a *Boolean conjunctive query* (BCQ). The evaluation of CQs is defined in terms of homomorphisms. Given an instance I, the *evaluation of* $q(\bar{x})$ *over* I, denoted $q(I)$, is the set of all tuples $\bar{t} \in \mathbf{C}^{|\bar{x}|}$ such that there exists a homomorphism h from $q(\bar{x})$ to I with $h(\bar{x}) = \bar{t}$. By abuse of notation, we sometimes treat a tuple of variables as a set of variables, and a conjunction of atoms as a set of atoms. Note that in the case of Boolean CQs, the only possible answer is the empty tuple.

A *union of conjunctive queries* (UCQ) over $\mathbf{S}$ is a formula of the form

$$q(\bar{x}) \ := \ q_1(\bar{x}) \vee \cdots \vee q_n(\bar{x}),$$

where each $q_i(\bar{x})$ is a CQ over $\mathbf{S}$. The evaluation of q over an instance I, denoted $q(I)$, is defined as the set of tuples $\bigcup_{i \in \{1,\dots,n\}} q_i(I)$. By abuse of notation, we may treat a UCQ $q(\bar{x})$ as the one above as the set of CQs $\{q_1(\bar{x}), \dots, q_n(\bar{x})\}$.

2.2 Ontological query answering

A *tuple-generating dependency* (TGD) σ is a (constant-free) sentence

$$\forall \bar{x} \forall \bar{y} \ (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \, \psi(\bar{x}, \bar{z})) \, ,$$

where $\bar{x}$, $\bar{y}$, and $\bar{z}$ are tuples of variables of $\mathbf{V}$, and ϕ and ψ are conjunctions of atoms. For brevity, we write σ as $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \, \psi(\bar{x}, \bar{z})$, and use comma for joining atoms. We refer to $\phi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ as the *body* and *head* of σ, denoted $\mathsf{body}(\sigma)$ and $\mathsf{head}(\sigma)$, respectively. An instance I satisfies a TGD σ as the one above, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\phi(\bar{x}, \bar{y})) \subseteq I$, then there exists an *extension* h' of $h_{|\bar{x}}$, i.e., $h' \supseteq h_{|\bar{x}}$, such that $h'(\psi(\bar{x}, \bar{z})) \subseteq I$. The instance I satisfies a set Σ of TGDs, written $I \models \Sigma$, if $I \models \sigma$ for each $\sigma \in \Sigma$. Let TGD be the class of (finite) sets of TGDs.

A *negative constraint* (NC) σ is a sentence of the form

$$\forall \bar{x} \ (\phi(\bar{x}) \rightarrow \bot) \, ,$$

where $\bar{x}$ is a tuple of variables of $\mathbf{V}$, ϕ is a conjunction of atoms, and $\bot$ denotes the truth constant false. For brevity, we write σ as $\phi(\bar{x}) \rightarrow \bot$, and use comma for joining atoms. We refer to $\phi(\bar{x})$ as the *body* of σ, denoted $\mathsf{body}(\sigma)$. An instance I satisfies an NC σ as the one above, written $I \models \sigma$, if there is *no* homomorphism h such that $h(\phi(\bar{x})) \subseteq I$ (observe the inversion: satisfies NC $\rightarrow$ no homomorphism exists). The instance I satisfies a set Σ of NCs, written $I \models \Sigma$, if $I \models \sigma$ for each $\sigma \in \Sigma$. Let NC be the class of (finite) sets of NCs.

Consider a database D and a set Σ of TGDs and NCs; henceforth, we denote by $\tau(\Sigma)$ and $\nu(\Sigma)$ the set of TGDs and NCs, respectively, occurring in Σ. A *model* of D and Σ is an instance $I \supseteq D$ such that $I \models \tau(\Sigma)$ and $I \models \nu(\Sigma)$. We write $\mathsf{mods}(D, \Sigma)$ for the set of models of D and Σ. The *certain answers* to a CQ q w.r.t. D and Σ is defined as the set of tuples

$$\mathsf{cert}(q, D, \Sigma) \ = \bigcap_{I \in \mathsf{mods}(D, \Sigma)} q(I).$$

A problem that is central for our work is to compute the certain answers to a CQ w.r.t. a database and a set of TGDs and NCs that falls in $\mathsf{C} \cup \mathsf{NC}$, where $\mathsf{C} \subseteq \mathsf{TGD}$ is a class of TGDs; concrete classes of TGDs are given below. For brevity, given a class C of TGDs, we write $\mathsf{C}_\bot$ as an abbreviation for $\mathsf{C} \cup \mathsf{NC}$. As is customary when studying the complexity of this problem, we focus on its decision version:

PROBLEM :	$\mathsf{QAns}(\mathsf{C}_\perp)$
INPUT :	A database D, a set $\Sigma \in \mathsf{C}_\perp$, a CQ $q(\bar{x})$, and $\bar{c} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$.
QUESTION :	Does $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$?

This general formulation refers to the *combined complexity* of the problem, that is, the database, the set of TGDs and NCs, the CQ, and the candidate tuple are considered part of the input. It is common, however, to study also refined measures that are more realistic in practice. Here, we consider the *bounded-arity combined complexity*, where the arity of the underlying schema is bounded by an integer, the *fixed-program combined complexity*,[2] where the set of TGDs and NCs is fixed, and the *data complexity*, where the CQ and the set of TGDs and NCs are fixed. Henceforth, for brevity, we write c-, ba-, fp-, and d-complexity for combined, bounded-arity combined, fixed-program combined, and data complexity, respectively.

It should be clear that if the given database D and set Σ of TGDs and NCs are *inconsistent*, i.e., $\mathsf{mods}(D, \Sigma) = \varnothing$,[3] then the set of certain answers to a CQ $q(\bar{x})$ consists of all the tuples $\bar{c} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$. This is because the definition of certain answers relies on the classical first-order semantics. That is,

$$\mathsf{cert}(q, D, \Sigma) \ = \{\bar{c} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert} \mid D \wedge \Sigma \models_{\mathsf{FO}} q(\bar{c})\},$$

where $D \wedge \Sigma$ denotes the first-order sentence $\bigwedge_{\alpha \in D} \alpha \wedge \bigwedge_{\sigma \in \Sigma} \sigma$, $q(\bar{c})$ is the first-order sentence obtained by instantiating the free variables of q with $\bar{c}$, and $\models_{\mathsf{FO}}$ denotes the standard first-order entailment. Therefore, if D and Σ are inconsistent, this means that the sentence $D \wedge \Sigma$ does not admit a model, i.e., is a logical contradiction, and thus, everything is entailed from it:

$$\mathsf{mods}(D, \Sigma) = \varnothing \implies \mathsf{cert}(q, D, \Sigma) = \left\{\bar{c} \mid \bar{c} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}\right\}.$$

Another crucial observation is that, if $\mathsf{mods}(D, \Sigma) \neq \varnothing$ (i.e., D and Σ are consistent), then for computing the certain answers to q w.r.t. D and Σ, it suffices to focus on the TGDs of Σ. That is:

$$\mathsf{mods}(D, \Sigma) \neq \varnothing \implies \mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q, D, \tau(\Sigma)).$$

By exploiting the above two implications, we can easily show that:

$$\bar{c} \in \mathsf{cert}(q, D, \Sigma) \iff \mathsf{mods}(D, \Sigma) = \varnothing \quad \text{or} \quad \bar{c} \in \mathsf{cert}(q, D, \tau(\Sigma)). \tag{1}$$

It is not difficult to see that the problem of checking whether $\mathsf{mods}(D, \Sigma) = \varnothing$ boils down to the problem of checking whether the database and the set of TGDs entail at least one NC. Given an NC σ of the form $\phi(\bar{x}) \rightarrow \perp$, we write q_σ for the Boolean CQ $\exists \bar{x}\, \phi(\bar{x})$. Then:

$$\mathsf{mods}(D, \Sigma) = \varnothing \iff \text{there is } \sigma \in \nu(\Sigma) \text{ s.t. } \mathsf{cert}(q_\sigma, D, \tau(\Sigma)) \neq \varnothing.^4 \tag{2}$$

From (1) and (2), we immediately get the following folklore result:

Proposition 2.1. *Consider a database D, a set Σ of TGDs and NCs, a CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$. The following are equivalent:*

1. $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$.
2. *There is $\sigma \in \nu(\Sigma)$ such that* $\mathsf{cert}(q_\sigma, D, \tau(\Sigma)) \neq \varnothing$, *or* $\bar{c} \in \mathsf{cert}(q, D, \tau(\Sigma))$.

2.3 The chase procedure

The above proposition shows that for checking whether a tuple is a certain answer to a CQ w.r.t. a database and a set Σ of TGDs and NCs, we actually have to reason with the TGD component of Σ. The *chase procedure* is a useful algorithmic tool when reasoning with TGDs that takes as an input a database D and a set Σ of TGDs, and constructs a (possibly infinite) instance I such that $I \supseteq D$ and $I \models \Sigma$; see, e.g., [10, 21]. We start by defining the notion of chase step.

[2] A set of TGDs and NCs is sometimes called a program, and hence the term fixed-program.

[3] If Σ consists only of TGDs, then D and Σ are always consistent, i.e., $\mathsf{mods}(D, \Sigma) \neq \varnothing$. In this case, a model of D and Σ can be constructed via the chase procedure introduced in Section 2.3.

[4] Notice that q_σ is Boolean, and by $\mathsf{cert}(q_\sigma, D, \tau(\Sigma)) \neq \varnothing$, we simply mean that the only possible answer (i.e., the empty tuple) is a certain answer.

Consider an instance I and a TGD σ of the form $\phi(\bar{x},\bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x},\bar{z})$. We say that σ is *applicable* w.r.t. I if there exists a homomorphism h from $\phi(\bar{x},\bar{y}))$ to I. In this case, *the result of applying σ over I with h* is the instance $J = I \cup h'(\psi(\bar{x},\bar{z}))$, where h' is an extension of $h_{|\bar{x}}$ that maps each variable of $\bar{z}$ to a distinct null not in I, and for each pair (z,w) of distinct variables of $\bar{z}$, $h'(z) \neq h'(w)$. For such a single chase step, we write $I\langle\sigma,h\rangle J$.

The main idea of the chase is, starting from a database D, to exhaustively apply TGDs over the instance constructed so far. This simple idea is formalized via the notion of chase sequence. We distinguish the two cases where a chase sequence is finite or infinite. Consider a set Σ of TGDs:

- A finite sequence $I_0, I_1, \ldots, I_n$ of instances, where $n \geq 0$, is a *chase sequence for I_0 under* Σ if: (i) for each $0 \leq i < n$, $I_i\langle\sigma,h\rangle I_{i+1}$ for some $\sigma \in \Sigma$ and homomorphism h from $\mathsf{body}(\sigma)$ to I_i, (ii) for each $0 \leq i < j < n$, assuming that $I_i\langle\sigma_i,h_i\rangle I_{i+1}$ and $I_j\langle\sigma_j,h_j\rangle I_{j+1}$, $\sigma_i = \sigma_j$ implies $h_i \neq h_j$, i.e., h_i and h_j are different homomorphisms, and (iii) there is no TGD of Σ that is applicable w.r.t. I_n. The result of the chase is the (finite) instance I_n.

- An infinite sequence $I_0, I_1, \ldots$ of instances is a *chase sequence for I_0 under* Σ if: (i) for each $i \geq 0$, $I_i\langle\sigma,h\rangle I_{i+1}$ for some $\sigma \in \Sigma$ and homomorphism h from $\mathsf{body}(\sigma)$ to I_i, (ii) for each $i,j > 0$ such that $i \neq j$, assuming that $I_i\langle\sigma_i,h_i\rangle I_{i+1}$ and $I_j\langle\sigma_j,h_j\rangle I_{j+1}$, $\sigma_i = \sigma_j$ implies $h_i \neq h_j$, and (iii) for each $i \geq 0$, and for every $\sigma \in \Sigma$ that is applicable w.r.t. I_i due to a homomorphism h, there exists $j \geq i$ such that $I_j\langle\sigma,h\rangle I_{j+1}$. The latter is known as the *fairness condition*, and guarantees that all the applicable TGDs eventually will be applied. The result of the chase is $\bigcup_{i\geq 0} I_i$.

Since we consider the *oblivious* version of the chase, i.e., a TGD is applied whenever its body is satisfied no matter whether its head is satisfied, every chase sequence for I under Σ leads to the same result (up to isomorphism). Thus, we can refer to *the* result of the chase for I under Σ, denoted $\mathsf{chase}(I,\Sigma)$.

The following is a well-known result, which exposes the usefulness of the chase in relation with ontological query answering. The key reason why this result holds is because, given a database D and a set Σ of TGDs, the instance $\mathsf{chase}(D,\Sigma)$ is not only a model, but is a *universal model* of D and Σ, which means that $\mathsf{chase}(D,\Sigma)$ can be homomorphically mapped to every instance $I \in \mathsf{mods}(D,\Sigma)$.

Proposition 2.2 (see, e.g., [19, 21])**.** *Consider a database D, a set Σ of TGDs , and a CQ q. It holds that* $\mathsf{cert}(q,D,\Sigma) = q(\mathsf{chase}(D,\Sigma))$.

2.4 Complexity classes

We assume that the reader has some background in computational complexity theory, including the notions of Turing machine, and hardness and completeness of a problem for a complexity class, as can be found in standard textbooks, e.g., in [27, 42]. In what follows, we briefly recall the complexity classes that we encounter in our complexity results. The complexity class PSPACE (resp., PTIME, EXPTIME, and 2EXPTIME) contains all the decision problems that can be solved in polynomial space (resp., polynomial, exponential, and double exponential time) via a deterministic Turing machine. The complexity classes NP and NEXPTIME contain all the decision problems that can be solved in polynomial and exponential time via a non-deterministic Turing machine, respectively, while coNP and coNEXPTIME are their complementary classes, where "Yes" and "No" instances are interchanged. The class Θ_2^P is the class of all decision problems that can be decided in polynomial time by a deterministic Turing machine using a logarithmic number of calls to an NP-oracle. The class Σ_2^P is the class of problems that can be solved in non-deterministic polynomial time using an NP-oracle, and Π_2^P is the complement of Σ_2^P. The complexity class AC_0 is the class of all languages that are decidable via uniform families of Boolean circuits of polynomial size and constant depth. The above complexity classes and their inclusion relationships (which are all currently believed to be strict) are shown below:

$$\begin{aligned}\mathrm{AC}_0 \;\subseteq\; \textsc{PTime} \subseteq \mathrm{NP}, \mathrm{coNP} \subseteq \Theta_2^P \subseteq \Sigma_2^P, \Pi_2^P \subseteq \textsc{PSpace} \subseteq \textsc{ExpTime} \\ \subseteq \textsc{NExpTime}, \mathrm{co}\textsc{NExpTime} \subseteq \mathrm{P}^{\textsc{NExpTime}} \subseteq 2\textsc{ExpTime}.\end{aligned}$$

3 Ontological query answering: Overview and new results

It is well known that $\mathsf{QAns}(\mathsf{TGD}_\perp)$ is undecidable; this is implicit in [2], which studies the implication problem for database dependencies. A stronger result of this kind can be found in [10], where it is shown that $\mathsf{QAns}(\mathsf{TGD}_\perp)$ is undecidable even in data complexity. Actually, the above negative results hold

	c-complexity	ba-complexity	fp-complexity	d-complexity
$\mathsf{G}_\perp$	2ExpTime	ExpTime	NP	PTime
$\mathsf{L}_\perp$	PSpace	NP	NP	in AC_0
$\mathsf{A}_\perp$	NExpTime	NExpTime	NP	in AC_0
$\mathsf{S}_\perp$	ExpTime	NP	NP	in AC_0

Table 1: Complexity of $\mathsf{QAns}(\mathsf{C}_\perp)$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$. Apart from the AC_0 upper bounds, the rest are completeness results.

even without considering NCs. Let us stress that Propositions 2.1 and 2.2 do not provide a chase-based decision procedure for query answering, since the chase is (in general) infinite. This has led to an intensive research activity for identifying syntactic restrictions on sets of TGDs that lead to decidability. Such restrictions can be classified into three main syntactic paradigms: guardedness (which includes linearity), acyclicity, and stickiness. We proceed to recall each of those paradigms, and discuss the complexity of query answering (summarized in Table 1).

3.1 Guardedness

A TGD σ is called *guarded* if $\mathsf{body}(\sigma)$ has an atom, called *guard*, that contains all the variables occurring in $\mathsf{body}(\sigma)$. Although the chase under a set of guarded TGDs does not necessarily terminate, query answering is decidable. This follows from the fact that the result of the chase procedure is "treelike", or, in other words, has finite treewidth [10]. Let G be the class of sets of guarded TGDs. Then:

Proposition 3.1 ([10])**.** $\mathsf{QAns}(\mathsf{G}_\perp)$ *is* 2ExpTime*-complete in* c*-complexity,* Exp-Time*-complete in* ba*-complexity, NP-complete in* fp*-complexity, and* PTime*-comp-lete in* d*-complexity.*

A key subclass of guarded TGDs is the class of *linear* TGDs, i.e., TGDs whose body consists of a single atom [11]. Let L be the class of sets of linear TGDs.

Proposition 3.2 ([10])**.** $\mathsf{QAns}(\mathsf{L}_\perp)$ *is PSpace-complete in* c*-complexity, NP-comp-lete in* ba*-complexity and* fp*-complexity, and in AC_0 in* d*-complexity.*

3.2 Acyclicity

The *predicate graph* of a set Σ of TGDs is defined as follows: its nodes are the predicates occurring in Σ, and there is an edge from P to R iff there is a TGD $\sigma \in \Sigma$ such that P occurs in $\mathsf{body}(\sigma)$ and R occurs in $\mathsf{head}(\sigma)$. We call Σ *acyclic* (a.k.a. *non-recursive*) if its *predicate graph* contains no directed cycles. It is easy to see that acyclicity ensures the termination of the chase. Thus, by Propositions 2.1 and 2.2, we immediately get the decidability of $\mathsf{QAns}(\mathsf{A}_\perp)$, where A denotes the class of acyclic sets of TGDs. However, the exact complexity of the problem has not been studied before the conference paper [37], which is one of the works on which the present journal paper is based. We proceed to show that:

Proposition 3.3. $\mathsf{QAns}(\mathsf{A}_\perp)$ *is NExpTime-complete in* c*- and* ba*-complexity, NP-complete in* fp*-complexity, and in AC_0 in* d*-complexity.*

To establish the upper bounds for $\mathsf{QAns}(\mathsf{A}_\perp)$, Proposition 2.1 suggests that it suffices to establish the same upper bounds for $\mathsf{QAns}(\mathsf{A})$. At this point, one may be tempted to think that this can be done by simply constructing the chase instance I, and then checking whether the given tuple belongs to the evaluation of the CQ over I. Although this simple algorithm shows that indeed $\mathsf{QAns}(\mathsf{A})$ is in NP in fp-complexity, it does not lead to optimal upper bounds in the case of c-complexity, ba-complexity, and d-complexity. In particular, it yields a 2ExpTime upper bound in c-complexity and ba-complexity, and a PTime upper bound in d-complexity. The next example shows that indeed this is the best that we can achieve via the naive procedure that explicitly constructs the chase instance.

Example 3.1. Consider the family of acyclic sets of TGDs

$$\left\{\Sigma_n \;=\; \{R_i(x), R_i(y) \rightarrow \exists z\, P_{i+1}(x,y,z), R_{i+1}(z)\}_{i \in \{0,\ldots,n\}}\right\}_{n \geq 0}.$$

Consider also the family of databases

$$\{D_m \;=\; \{R_0(c_1), \ldots, R_0(c_m)\}\}_{m \geq 0}.$$

The i-th stratum of Σ_n computes all the pairs of terms using the $m^{(2^i)}$ terms stored in the predicate R_i, and for each such pair generates a fresh null that is stored in the predicate R_{i+1}. It is easy to verify that the predicate R_n in $\mathsf{chase}(D_m, \Sigma_n)$ contains $m^{(2^n)}$ nulls. Since each chase step generates exactly one null, the chase procedure starting from D_m and applying TGDs of Σ_n terminates after double-exponentially many steps in n, and polynomially many steps in m.[5] ∎

We proceed to provide more refined procedures that lead to optimal complexity upper bounds for $\mathsf{QAns}(\mathsf{A})$. In Section 3.2.1 we establish the desired upper bounds, and we show that they are indeed worst-case optimal in Section 3.2.2.

3.2.1 Upper bounds

Our main technical result, which in turn allows us to obtain the desired upper bounds for $\mathsf{QAns}(\mathsf{A})$, essentially shows that, for query answering purposes under acyclic sets of TGDs, it suffices to apply exponentially many chase steps, while this exponential bound does not depend on the input database. To formalize this statement, we first need to recall the notion of stratification.

Consider a set Σ of TGDs. Let $\mathsf{sch}(\Sigma)$ be the set of predicates occurring in Σ. A *stratification* of Σ is a partition $\{\Sigma_1, \ldots, \Sigma_n\}$, where $n \geq 1$, of Σ such that, for some function $f : \mathsf{sch}(\Sigma) \rightarrow \{1, \ldots, n\}$, the following holds:

- For each predicate $R \in \mathsf{sch}(\Sigma)$, all the TGDs with R in their head belong to $\Sigma_{f(R)}$, i.e., they belong to the same set of the partition.
- If there exists a TGD in Σ such that the predicate R appears in its body, while the predicate P appears in its head, then $f(R) < f(P)$.

The *depth* of a set Σ of TGDs that admits a stratification, denoted $\mathsf{depth}(\Sigma)$, is defined as the cardinality of its smallest stratification.

It is easy to verify that if a set of TGDs is acyclic, then it admits a stratification, and thus we can refer to its depth. Consider now a CQ q and a set $\Sigma \in \mathsf{A}$. Let $|q|$ be the number of atoms occurring in q, and $\mathsf{width}(\Sigma)$ be the maximum number of atoms occurring in the body of a TGD of Σ. We define the function

$$g(q, \Sigma) \;=\; \begin{cases} |q| \cdot \left\lfloor \frac{\mathsf{width}(\Sigma)^{\mathsf{depth}(\Sigma)+1} - 1}{\mathsf{width}(\Sigma) - 1} \right\rfloor & \text{if } \mathsf{width}(\Sigma) > 1 \\[2ex] |q| \cdot \mathsf{depth}(\Sigma) & \text{if } \mathsf{width}(\Sigma) = 1. \end{cases}$$

Roughly speaking, $g(q, \Sigma)$ provides an upper bound on the number of chase steps that we need to apply in order to be able to safely conclude whether the query q is entailed by the instance $\mathsf{chase}(D, \Sigma)$ for any database D.

Lemma 3.1. *Consider a database D, a set $\Sigma \in \mathsf{A}$ of TGDs, a CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$. If $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$, then there exists a sequence of instances $(J_i)_{0 \leq i \leq g(q,\Sigma)}$ with $J_0 = D$ and $J_i\langle \sigma, \mu \rangle J_{i+1}$ for some TGD $\sigma \in \Sigma$ and homomorphism μ from $\mathsf{body}(\sigma)$ to $\mathsf{dom}(J_i)$ such that $\bar{c} \in q(J_{g(q,\Sigma)})$.*

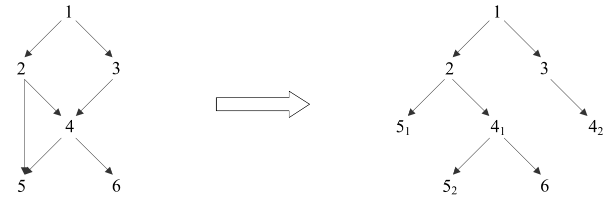
Proof. By hypothesis, there is a chase sequence $s = (I_i)_{0 \leq i \leq n}$ with $I_i\langle \sigma_i, h_i \rangle I_{i+1}$ for D under Σ such that $\bar{c} \in q(I_n)$ with $h(\bar{x}) = \bar{c}$. Consider an arbitrary atom α of q. It is clear that $h(\alpha) \in I_n$. We are going to establish an upper bound on the number of chase steps of s that are really needed to generate $h(\alpha)$. To this end, we first need to recall the so-called *chase relation* of s, denoted $\prec_s$, which is a binary relation over the atoms of I_n such that $(\beta, \gamma) \in \prec_s$ iff there exists $i \in \{0, \ldots, n-1\}$ with $\beta \in h_i(\mathsf{body}(\sigma_i))$ and $\gamma \in I_{i+1} \setminus I_i$. In simple words, $\prec_s$ encodes which atoms generate some other atom via a single chase step. For convenience, we write $\beta \prec_s \gamma$ for the fact that (β, γ) belongs to $\prec_s$. We also write $\prec_s^\star$ for the transitive closure of $\prec_s$. Let $\prec_{s,h(\alpha)}$ be the subrelation of $\prec_s$ defined as

$$\{(\beta, \gamma) \mid \beta \prec_s \gamma \text{ and } \gamma = h(\alpha) \text{ or } \gamma \prec_s^\star h(\alpha)\}.$$

Roughly, $\prec_{s,h(\alpha)}$ collects only the sequences of atoms that lead to $h(\alpha)$. Observe that the inverse relation of $\prec_{s,h(\alpha)}$, denoted $\prec_{s,h(\alpha)}^-$, is a directed acyclic graph, where its nodes are atoms of I_n, with $h(\alpha)$ being the root, i.e., is the only node without an incoming edge. We can now transform $\prec_{s,h(\alpha)}^-$ into the binary

[5] This example provides a lower bound even if we consider the restricted version of the chase, where a TGD σ is applicable w.r.t. an instance I only if it is really violated, i.e., the homomorphism that maps $\mathsf{body}(\sigma)$ to I cannot be extended to a homomorphism that maps $\mathsf{head}(\sigma)$ to I.

relation $\prec_{s,h(\alpha)}^{-,\Delta}$ that represents a rooted tree, where $h(\alpha)$ is the root node, and each node has at most $\mathsf{width}(\Sigma)$ children. This is done in the obvious way by duplicating some nodes of $\prec_{s,h(\alpha)}^{-}$ as shown in the figure below; the formal definition is omitted:



It should be clear that by providing an upper bound on the number of nodes occurring in $\prec_{s,h(\alpha)}^{-,\Delta}$, we immediately get an upper bound on the number of nodes in $\prec_{s,h(\alpha)}^{-}$. Due the fact that Σ is acyclic, the depth of the rooted tree $\prec_{s,h(\alpha)}^{-,\Delta}$ is at most $\mathsf{depth}(\Sigma)$. Thus, the number of nodes in $\prec_{s,h(\alpha)}^{-,\Delta}$ is at most

$$\hat{g}(\Sigma) \;=\; \begin{cases} \left\lfloor \frac{\mathsf{width}(\Sigma)^{\mathsf{depth}(\Sigma)+1}-1}{\mathsf{width}(\Sigma)-1} \right\rfloor & \text{if } \mathsf{width}(\Sigma) > 1 \\[2ex] \mathsf{depth}(\Sigma) & \text{if } \mathsf{width}(\Sigma) = 1. \end{cases}$$

This allows us to conclude that $\prec_{s,h(\alpha)}$ has at most $\hat{g}(\Sigma)$ nodes, which means that the number of atoms in I_n that are needed to generate $h(\alpha)$ is at most $\hat{g}(\Sigma)$. This implies that the number of chase steps of s that are needed to generate $h(\alpha)$ is at most $\hat{g}(\Sigma)$. Hence, we can construct from s a sequence of instances $(J_i)_{0\leq i\leq |q|\cdot\hat{g}(\Sigma)}$ with $J_0 = D$ and $J_i\langle\sigma, h\rangle J_{i+1}$ for some $\sigma \in \Sigma$ and homomorphism μ from $\mathsf{body}(\sigma)$ to $\mathsf{dom}(J_i)$ such that $\bar{c} \in q(J_{|q|\cdot\hat{g}(\Sigma)})$. By definition, $|q| \cdot \hat{g}(\Sigma) = g(q, \Sigma)$, and the claim follows. $\square$

Having the above lemma in place, it is now easy to devise a non-deterministic algorithm for $\mathsf{QAns}(\mathsf{A})$ that runs in exponential time in general, and in polynomial time whenever the set of TGDs is fixed: guess a sequence of instances $(J_i)_{0\leq i\leq g(q,\Sigma)}$ with $J_0 = D$, a sequence of pairs $(\sigma_i, h_i)_{0\leq i\leq g(q,\Sigma)-1}$, where $\sigma_i \in \Sigma$ and h_i is a substitution from the set of variables occurring in $\mathsf{body}(\sigma_i)$ to $\mathsf{dom}(J_i)$, and a substitution h from the variables occurring in $q(\bar{x})$ to $\mathsf{dom}(J_g(q, \Sigma))$ with $h(\bar{x}) = \bar{c}$, and then check that $J_i\langle\sigma_i, h_i\rangle J_{i+1}$ for each $i \in \{0, \ldots, g(q, \Sigma) - 1\}$, and h maps q to $J_{g(q,\Sigma)}$. Therefore, we obtain that $\mathsf{QAns}(\mathsf{A})$ is in NExpTime in c- and ba-complexity, and in NP in fp-complexity. However, the above algorithm provides only a PTime upper bound for $\mathsf{QAns}(\mathsf{A})$ in d-complexity. For the latter type of complexity we need to argue a bit more.

It is implicit in [11] that Lemma 3.1 above implies that the class of acyclic sets of TGDs is *UCQ-rewritable*: given a set $\Sigma \in \mathsf{A}$ of TGDs and a CQ $q(\bar{x})$, we can construct a (finite) UCQ $Q_{q,\Sigma}(\bar{x})$ such that, for every database D and tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$, $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$ iff $\bar{c} \in Q_{q,\Sigma}(D)$. Since evaluating a *fixed* UCQ over a database is in AC_0 [47], we get that $\mathsf{QAns}(\mathsf{A})$ is in AC_0 in d-complexity.

3.2.2 Lower bounds

We now proceed to establish the complexity lower bounds stated in Proposition 3.3. Actually, the NP-hardness of $\mathsf{QAns}(\mathsf{A})$ in fp-complexity is inherited from the well-known fact that deciding whether a tuple of constants belongs to the evaluation of a CQ over a database is NP-hard, even if the underlying schema is fixed. It remains to show the following:

Lemma 3.2. $\mathsf{QAns}(\mathsf{A})$ *is NExpTime-hard in* ba*-complexity.*

The above result is shown via a reduction from the standard exponential tiling problem. A *tiling system* is a tuple $\mathcal{T} = (n, m, H, V, s)$, where n and m are numbers in unary, H and V are subsets of $\{1, \ldots, m\} \times \{1, \ldots, m\}$, and s is a sequence of numbers of $\{1, \ldots, m\}$; let $s[i]$ be the i-th element of s. An *exponential tiling* for $\mathcal{T}$ is a function $f : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, 2^n - 1\} \to \{1, \ldots, m\}$ such that:

- $f(i, 0) = s[i]$, for each $0 \leq i \leq (|s| - 1)$,
- $(f(i, j), f(i + 1, j)) \in H$, for each $0 \leq i \leq 2^n - 2$ and $0 \leq j \leq 2^n - 1$, and
- $(f(i, j), f(i, j + 1)) \in V$, for each $0 \leq i \leq 2^n - 1$ and $0 \leq j \leq 2^n - 2$.

The exponential tiling problem is defined as follows:

PROBLEM :	ExpTiling
INPUT :	A tiling system $\mathcal{T}$.
QUESTION :	Is there an exponential tiling for $\mathcal{T}$?

The goal is to provide a polynomial time reduction from ExpTiling to QAns(A). Given a tiling system $\mathcal{T} = (n, m, H, V, s)$, we are going to construct in polynomial time a database $D_{\mathcal{T}}$, and a set $\Sigma_{\mathcal{T}} \in \mathsf{A}$ of TGDs that mentions only predicates of fixed arity such that $\mathcal{T}$ has an exponential tiling iff $\mathsf{cert}(\mathrm{Yes}(), D_{\mathcal{T}}, \Sigma_{\mathcal{T}}) \neq \varnothing$

The database $D_{\mathcal{T}}$. It simply stores the horizontal and vertical compatibility relations H and V, respectively, together with the sequence of numbers s:

$$D_{\mathcal{T}} \;=\; \{H(i,j) \mid (i,j) \in H\} \;\cup\; \{V(i,j) \mid (i,j) \in V\} \;\cup\; \{S_i(s[i])\}_{i \in \{0,\dots,|s|-1\}}.$$

The set of TGDs $\Sigma_{\mathcal{T}}$. The idea underlying $\Sigma_{\mathcal{T}}$ is, during the chase, to inductively construct tilings of size $2^i \times 2^i$ from tilings of size $2^{i-1} \times 2^{i-1}$. This exploits the following simple fact, which has been already observed in [18] where Datalog with complex values is studied: the left square in the figure below of size $2^i \times 2^i$, for $i > 1$, with each x_i, y_i, z_i, w_i being of size $2^{i-2} \times 2^{i-2}$, satisfies the horizontal and vertical compatibility relations iff the nine subsquares of size $2^{i-1} \times 2^{i-1}$ depicted on the right in the following figure satisfy the compatibility relations. Let us clarify that the origin of a grid is considered to be the upper-left cell.

x_1	x_2	y_1	y_2
x_3	x_4	y_3	y_4
z_1	z_2	w_1	w_2
z_3	z_4	w_3	w_4

$$\begin{array}{|c|c|}\hline x_1 & x_2 \\ \hline x_3 & x_4 \\ \hline\end{array}\quad\begin{array}{|c|c|}\hline x_2 & y_1 \\ \hline x_4 & y_3 \\ \hline\end{array}\quad\begin{array}{|c|c|}\hline y_1 & y_2 \\ \hline y_3 & y_4 \\ \hline\end{array}$$

$$\begin{array}{|c|c|}\hline x_3 & x_4 \\ \hline z_1 & z_2 \\ \hline\end{array}\quad\begin{array}{|c|c|}\hline x_4 & y_3 \\ \hline z_2 & w_1 \\ \hline\end{array}\quad\begin{array}{|c|c|}\hline y_3 & y_4 \\ \hline w_1 & w_2 \\ \hline\end{array}$$

$$\begin{array}{|c|c|}\hline z_1 & z_2 \\ \hline z_3 & z_4 \\ \hline\end{array}\quad\begin{array}{|c|c|}\hline z_2 & w_1 \\ \hline z_4 & w_3 \\ \hline\end{array}\quad\begin{array}{|c|c|}\hline w_1 & w_2 \\ \hline w_3 & w_4 \\ \hline\end{array}$$

To achieve this construction, we encode $2^i \times 2^i$ squares, for $i > 0$, of the form

ul	ur
ll	lr

where ul, ur, ll, lr are squares of size $2^{i-1} \times 2^{i-1}$, as relational atoms of the form

$$T_i(id, ul, ur, ll, lr),$$

where id is the identity of the encoded square, and ul, ur, ll, lr are the identities of its four subsquares as shown above.

We first construct squares of size 2×2 that satisfy the compatibility relations directly from H and V stored in the database $D_{\mathcal{T}}$. This is done via the TGD

$$H(x_1, x_2), H(x_3, x_4), V(x_1, x_3), V(x_2, x_4) \rightarrow \exists z\, T_1(z, x_1, x_2, x_3, x_4).$$

The inductive construction of squares of size $2^i \times 2^i$, for $i \in \{2, \dots, n\}$, which satisfy the compatibility relations, from squares of size $2^{i-1} \times 2^{i-1}$, is done via the following TGDs. For each $i \in \{1, \dots, n-1\}$, we have the TGD

$$\begin{array}{l} T_i(u_1, x_1, x_2, x_3, x_4), T_i(u_2, x_2, y_1, x_4, y_3), T_i(u_3, y_1, y_2, y_3, y_4), \\ T_i(u_4, x_3, x_4, z_1, z_2), T_i(u_5, x_4, y_3, z_2, w_1), T_i(u_6, y_3, y_4, w_1, w_2), \\ T_i(u_7, z_1, z_2, z_3, z_4), T_i(u_8, z_2, w_1, z_4, w_3), T_i(u_9, w_1, w_2, w_3, w_4) \rightarrow \exists u\, T_{i+1}(u, u_1, u_3, u_7, u_9). \end{array}$$

We now need to verify the initial condition. To this end, we need to extract from the squares of size $2^n \times 2^n$ the tiles at positions $(0,0),(1,0),\ldots,(|s|-1,0)$. This is done by defining relational atoms of the form

$$\mathrm{Top}_i^j(x,y),$$

where $1 \leq i \leq n$ and $0 \leq j \leq |s|-1$, to express that in the $2^i \times 2^i$ square x, at position $(j,0)$, we have the tile $y \in \{1,\ldots,m\}$. We then need to add the TGDs

$$T_1(x,x_1,x_2,x_3,x_4) \rightarrow \mathrm{Top}_1^0(x,x_1), \mathrm{Top}_1^1(x,x_2),$$

for each $i \in \{2,\ldots,\lceil \log |s| \rceil\}$,

$$T_i(x,x_1,x_2,x_3,x_4), \mathrm{Top}_{i-1}^0(x_1,y_0),\ldots,\mathrm{Top}_{i-1}^{2^{i-1}-1}(x_1,y_{2^{i-1}-1}) \rightarrow \mathrm{Top}_i^0(x,y_0),\ldots,\mathrm{Top}_i^{2^{i-1}-1}(x,y_{2^{i-1}-1}),$$

$$T_i(x,x_1,x_2,x_3,x_4), \mathrm{Top}_{i-1}^0(x_2,y_0),\ldots,\mathrm{Top}_{i-1}^{2^{i-1}-1}(x_2,y_{2^{i-1}-1}) \rightarrow \mathrm{Top}_i^{2^{i-1}}(x,y_0),\ldots,\mathrm{Top}_i^{2^{i}-1}(x,y_{2^{i-1}-1}),$$

and, for each $i \in \{\lceil \log |s| \rceil + 1,\ldots,n\}$,

$$T_i(x,x_1,x_2,x_3,x_4), \mathrm{Top}_{i-1}^0(x_1,y_0),\ldots,\mathrm{Top}_{i-1}^{|s|-1}(x_1,y_{|s|-1}) \rightarrow \mathrm{Top}_i^0(x,y_0),\ldots,\mathrm{Top}_i^{|s|-1}(x,y_{|s|-1}).$$

We finally add the TGD

$$\mathrm{Top}_n^0(x,y_0), S_0(y_0),\ldots,\mathrm{Top}_n^{|s|-1}(x,y_{|s|-1}), S_{|s|-1}(y_{|s|-1}) \rightarrow \mathrm{Yes}(),$$

which simply checks whether there exists a square of size $2^n \times 2^n$ that complies with H and V, and, in addition, the tile at position $(i,0)$, for $i \in \{0,\ldots,|s|-1\}$, is $s[i]$, i.e., it checks whether an exponential tiling for $\mathcal{T}$ has been found.

It should be clear that $\mathcal{T}$ has an exponential tiling iff the atom Yes() occurs in $\mathsf{chase}(D_{\mathcal{T}},\Sigma_{\mathcal{T}})$, which implies that the above construction is correct. It should be also clear that $D_{\mathcal{T}}$ and $\Sigma_{\mathcal{T}}$ can be constructed in polynomial time, while $\Sigma_{\mathcal{T}}$ mentions only predicates of arity at most five. This completes the proof of Lemma 3.2.

3.3 Stickiness

This condition, introduced in [13], is inherently different from guardedness and acyclicity. It ensures neither finite treewidth nor termination of the chase. Instead, the decidability of query answering is obtained via backward-chaining techniques. The goal of stickiness is to capture joins among variables that are not expressible via guarded TGDs, but without forcing the chase to terminate. The key property underlying this condition is that, during the chase, terms that unify with variables that appear more than once in the body of a TGD (i.e., join variables) are always propagated (or "stick") to the inferred atoms; this is graphically illustrated as

$$\begin{array}{ccc} R(x,y),\ P(y,z) \rightarrow \exists w\, T(x,y,w) & \qquad & R(x,y),\ P(y,z) \rightarrow \exists w\, T(x,y,w) \\ T(x,y,z) \rightarrow \exists w\, S(y,w) & \qquad & T(x,y,z) \rightarrow \exists w\, S(x,w) \\ & & \times \end{array}$$

where the first set of TGDs is sticky, while the second is not. The formal definition is based on an inductive procedure that marks the variables that may violate the semantic property described above. Roughly, during the base step of this procedure, a variable that appears in the body of a TGD σ but not in every head atom of σ is marked. Then, the marking is inductively propagated from head to body as follows

$$\begin{array}{c} R(x,\boldsymbol{y}),\ P(\boldsymbol{y},z) \rightarrow \exists w\, T(x,y,w) \\ T(x,\boldsymbol{y},z) \rightarrow \exists w\, S(x,w) \end{array}$$

Stickiness requires every marked variable to appear only once in the body of a TGD. Let us now give the formal definition.

Consider a set Σ of TGDs; we can always assume that the TGDs in Σ do not share variables. For brevity, given an atom $R(\bar{t})$ and a variable $x \in \bar{t}$, $\mathsf{pos}(R(\bar{t}), x)$ is the set of positions in $R(\bar{t})$ at which x occurs; a position $R[i]$ identifies the i-th attribute of the predicate R. Let $\sigma \in \Sigma$ and x a variable in the body of σ. We inductively define when x *is marked in* Σ as follows:

- If there is an atom $R(\bar{t}) \in \mathsf{head}(\sigma)$ such that $x \notin \bar{t}$, then x is marked in Σ.
- Assuming that there exists an atom $R(\bar{t}) \in \mathsf{head}(\sigma)$ such that $x \in \bar{t}$, if there is $\sigma' \in \Sigma$ that has in its body an atom of the form $R(\bar{t}')$, and each variable in $R(\bar{t}')$ at a position of $\mathsf{pos}(R(\bar{t}), x)$ is marked in Σ, then x is marked in Σ.

The set Σ is *sticky* if there is no TGD whose body contains two occurrences of a variable that is marked in Σ. Let S be the class of sticky sets of TGDs. Then:

Proposition 3.4 ([13, 23])**.** $\mathsf{QAns}(\mathsf{S}_\perp)$ *is* EXPTIME*-complete in* c*-complexity, NP-complete in* ba*-complexity and* fp*-complexity, and in* AC_0 *in* d*-complexity.*

Let us clarify that in [13], where stickiness has been introduced, only the c-complexity, fp-complexity, and d-complexity have been considered. Moreover, the query answering algorithm devised in [13] does not provide an NP upper bound in the case of ba-complexity. This result is implicit in [23], where a result analogous to Lemma 3.1 is shown. In particular, there is a function $g(x, y)$, which is polynomial in x and exponential in y, such that, for every database D, set $\Sigma \in \mathsf{S}$, CQ $q(\bar{x})$, and tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$, $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$ implies there exists a sequence of instances $(I_i)_{0 \leq i \leq g(|q|+n,m)}$ with $J_0 = D$ and $I_i\langle\sigma, h\rangle I_{i+1}$ for some TGD $\sigma \in \Sigma$ and homomorphism h, where n is the number of predicates in Σ, and m the maximum arity over all those predicates, such that $\bar{c} \in q(I_{g(|q|+n,m)})$. Thus, if the arity of the underlying schema is bounded by a constant, $g(|q| + n, m)$ is a polynomial, which in turn leads to an easy guess and check algorithm for $\mathsf{QAns}(\mathsf{S})$ that runs in polynomial time. Therefore, $\mathsf{QAns}(\mathsf{S})$ is in NP in ba-complexity.

4 Consistent ontological query answering

As already discussed in Section 2, the set of certain answers to a CQ $q(\bar{x})$ w.r.t. a database D and a set Σ of TGDs and NCs that are inconsistent consists of all the tuples $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$. This exposes the main weakness of the standard certain answers semantics as defined above: the answers that we obtain from databases that are inconsistent with the given set of TGDs and NCs are not meaningful in practice. For this reason, several inconsistency-tolerant semantics have been proposed in the literature. All these inconsistency-tolerant semantics are based on the key notion of repair, which is essentially a $\subseteq$-maximal consistent subset of the given database.

Definition 4.1 (Repairs)**.** Consider a database D and a set Σ of TGDs and NCs. A *repair* of D and Σ is a database $D' \subseteq D$ such that

1. $\mathsf{mods}(D', \Sigma) \neq \emptyset$, and
2. there is no $\alpha \in D \setminus D'$ such that $\mathsf{mods}(D' \cup \{\alpha\}, \Sigma) \neq \emptyset$.

We write $\mathsf{reps}(D, \Sigma)$ for the set of repairs of D and Σ. ■

A simple example that illustrates the notion of repair follows. Note that this example will also serve as a running example in the rest of the section for illustrating the different inconsistency-tolerant semantics that we consider in our work.

Example 4.2. Consider the database

$$D \;=\; \{\text{Professor}(p), \text{Postdoc}(p), \text{Group}(g), \text{LeaderOf}(p, g)\},$$

asserting that p is a professor, a postdoc, and the leader of the research group g. Consider also the set Σ of TGDs and NCs consisting of

$$\text{Professor}(x) \rightarrow \exists y\, \text{Researcher}(x), \text{WorksOn}(x, y), \text{Project}(y)$$
$$\text{Postdoc}(x) \rightarrow \exists y\, \text{Researcher}(x), \text{WorksOn}(x, y), \text{Project}(y)$$
$$\text{LeaderOf}(x, y) \rightarrow \text{Professor}(x), \text{Group}(y)$$
$$\text{Professor}(x), \text{Postdoc}(x) \rightarrow \perp,$$

expressing that professors and postdocs are researchers who work on some project, the domain (range) of the relation LeaderOf$(\cdot,\cdot)$ consists of professors (research groups), and professors and postdocs form disjoint sets. Clearly, $\mathsf{mods}(D,\Sigma) = \varnothing$ since p violates the disjointness assertion. The repairs of D and Σ are

$$\begin{aligned} D_1 &= \{\text{Professor}(p), \text{Group}(g), \text{LeaderOf}(p,g)\} \\ D_2 &= \{\text{Postdoc}(p), \text{Group}(g)\}. \end{aligned}$$

To obtain D_1 it suffices to remove the atom Postdoc(p) from D. To obtain D_2, apart from removing Professor(p), we also need to remove LeaderOf(p,g), which, together with the third TGD above, implies Professor(p). ∎

4.1 ABox repair semantics

Having the notion of repair in place, we can now recall the main inconsistency-tolerant semantics, i.e., the ABox repair (AR) semantics [28]. Notice that this semantics has been proposed in the context of description logics, where the database is called assertional box (ABox); hence the name ABox repair. The underlying idea is very simple: a tuple is a certain answer if it is entailed by every repair.

Definition 4.3 (AR Semantics)**.** Consider a database D, and a set Σ of TGDs and NCs with $\mathsf{reps}(D,\Sigma) = \{D_1,\ldots,D_n\}$. The *AR-certain answers* to q w.r.t. D and Σ is defined as $\mathsf{cert}_{\mathsf{AR}}(q,D,\Sigma) = \bigcap_{i\in\{1,\ldots,n\}} \mathsf{cert}(q,D_i,\Sigma)$. ∎

A simple example that illustrates the AR semantics follows:

Example 4.4. Consider the database D and the set Σ of TGDs and NCs in Example 4.2. Consider also the Boolean CQs

$$\begin{aligned} q_1 &= \exists x\, \text{Group}(x) & \qquad q_2 &= \text{Researcher}(p) \\ q_3 &= \exists x\, \text{Project}(x) & q_4 &= \text{Professor}(p). \end{aligned}$$

The query q_1 asks whether a group exists, q_2 whether p is a researcher, q_3 whether a project exists, and q_4 whether p is a professor. Recall that $\mathsf{reps}(D,\Sigma) = \{D_1,D_2\}$ as in Example 4.2. Observe that, for each $i \in \{1,2\}$ and $j \in \{1,2,3\}$, it holds that $\mathsf{cert}(q_j,D_i,\Sigma) \neq \varnothing$; hence, $\mathsf{cert}_{\mathsf{AR}}(q_j,D,\Sigma) \neq \varnothing$. On the other hand, even if $\mathsf{cert}(q_4,D_2,\Sigma) \neq \varnothing$, $\mathsf{cert}(q_2,D_2,\Sigma) = \varnothing$, and thus $\mathsf{cert}_{\mathsf{AR}}(q_4,D,\Sigma) = \varnothing$. ∎

Although the AR semantics provides an elegant way to deal with inconsistency in ontological query answering, finding AR-certain answers is most commonly in coNP, and very often coNP-hard in d-complexity [28]. The reason is the fact that we need to consider many repairs, in general, exponentially many in the size of the database. This led to a large body of work on defining sound approximations of the AR semantics with the aim of reducing the complexity of query answering. Two of the most prominent such approximations are the intersection of repairs (IAR) semantics [28] and the intersection of closed repairs (ICR) semantics [3].

4.2 Intersection of ABox repairs semantics

The key idea is, instead of considering all the possible repairs, to focus on one repair that approximates all the others in a sound way. The obvious repair with this property is the one obtained by computing the intersection of all the repairs.

Definition 4.5 (IAR Semantics)**.** Consider a database D, and a set Σ of TGDs and NCs with $\mathsf{reps}(D,\Sigma) = \{D_1,\ldots,D_n\}$. The *IAR-certain answers* to a CQ q w.r.t. D and Σ is defined as $\mathsf{cert}_{\mathsf{IAR}}(q,D,\Sigma) = \mathsf{cert}(q, \bigcap_{i\in\{1,\ldots,n\}} D_i, \Sigma)$. ∎

Clearly, $\mathsf{cert}_{\mathsf{IAR}}(q,D,\Sigma) \subseteq \mathsf{cert}_{\mathsf{AR}}(q,D,\Sigma)$, i.e., the IAR semantics is indeed a sound approximation of the AR semantics. Here is a simple example:

Example 4.6. Consider the database D and the set Σ of TGDs and NCs in Example 4.2, and the CQs in Example 4.4. Recall that $\mathsf{reps}(D,\Sigma) = \{D_1,D_2\}$ given in Example 4.2, and thus $D_1 \cap D_2 = \{\text{Group}(g)\}$. Hence, $\mathsf{cert}_{\mathsf{IAR}}(q_1,D,\Sigma) = \mathsf{cert}(q_1, D_1 \cap D_2, \Sigma) \neq \varnothing$, while $\mathsf{cert}_{\mathsf{IAR}}(q_i,D,\Sigma) = \varnothing$ for each $i \in \{2,3,4\}$. ∎

	c-complexity	ba-complexity	fp-complexity	d-complexity
$\mathsf{G}_\perp$	2ExpTime	ExpTime	Π_2^P	coNP
$\mathsf{L}_\perp$	PSpace	Π_2^P	Π_2^P	coNP
$\mathsf{A}_\perp$	$\mathrm{P}^{\mathrm{NExpTime}}$	$\mathrm{P}^{\mathrm{NExpTime}}$	Π_2^P	coNP
$\mathsf{S}_\perp$	ExpTime	Π_2^P	Π_2^P	coNP

Table 2: Complexity of $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{C}_\perp)$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$; these are completeness results.

4.3 Intersection of closed repairs semantics

The key idea is, as in the case of the IAR semantics, to focus on one repair that approximates all the others in a sound way. This can be done in a more refined way than by considering the intersection of all repairs, which corresponds to closing the repairs w.r.t. the given set of TGDs before intersecting them. Given a database D and a set Σ of TGDs, we denote by $\mathsf{cl}(D, \Sigma)$ the set of ground atoms that can be entailed by D and Σ, i.e., the set of atoms $\bigcap \mathsf{mods}(D, \Sigma)$.

Definition 4.7 (ICR Semantics)**.** Consider a database D, and a set Σ of TGDs and NCs with $\mathsf{reps}(D, \Sigma) = \{D_1, \ldots, D_n\}$. The *ICR-certain answers* to a CQ q w.r.t. D and Σ is $\mathsf{cert}_{\mathsf{ICR}}(q, D, \Sigma) = \mathsf{cert}(q, \bigcap_{i \in \{1,\ldots,n\}} \mathsf{cl}(D_i, \tau(\Sigma)), \Sigma)$. ∎

Here is a simple example based on our running example:

Example 4.8. Consider the database D and the set Σ of TGDs and NCs in Example 4.2, and the CQs in Example 4.4. Recall that $\mathsf{reps}(D, \Sigma) = \{D_1, D_2\}$ as in Example 4.2; thus, $\mathsf{cl}(D_1, \tau(\Sigma)) \cap \mathsf{cl}(D_2, \tau(\Sigma)) = \{\mathrm{Research}(p), \mathrm{Group}(g)\}$. Hence, $\mathsf{cert}_{\mathsf{ICR}}(q_i, D, \Sigma) \neq \varnothing$ for $i \in \{1, 2\}$, and $\mathsf{cert}_{\mathsf{ICR}}(q_i, D, \Sigma) = \varnothing$ for $i \in \{3, 4\}$. ∎

Observe that in the scenario adopted in the above simple examples, where the repairs of D and Σ are the databases D_1 and D_2 given in Example 4.2, it holds that $D_1 \cap D_2 \subset \mathsf{cl}(D_1, \tau(\Sigma)) \cap \mathsf{cl}(D_2, \tau(\Sigma))$, which explains the fact that more queries are entailed in the case of the ICR semantics. In general, we can show that

$$\mathsf{cert}_{\mathsf{IAR}}(q, D, \Sigma) \;\subseteq\; \mathsf{cert}_{\mathsf{ICR}}(q, D, \Sigma) \;\subseteq\; \mathsf{cert}_{\mathsf{AR}}(q, D, \Sigma),$$

which justifies the statement that the ICR semantics is a finer approximation of the AR semantics than the IAR semantics.

4.4 Inconsistency-tolerant ontological query answering

Having the above semantics in place, we are now ready to revisit ontological query answering in order to ensure conceptually meaningful answers. The intention is not to compute the certain answers, but the s-certain answers, where s is one of the inconsistency-tolerant semantics described above, i.e., $\mathsf{s} \in \{\mathsf{AR}, \mathsf{IAR}, \mathsf{ICR}\}$. This gives rise to the following problems; as usual, C denotes a class of TGDs:

PROBLEM :	$\mathsf{QAns}_{\mathsf{s}}(\mathsf{C}_\perp)$
INPUT :	A database D, a set $\Sigma \in \mathsf{C}_\perp$, a CQ $q(\bar{x})$, and $\bar{c} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$.
QUESTION :	Does $\bar{c} \in \mathsf{cert}_{\mathsf{s}}(q, D, \Sigma)$?

We may also write $\mathsf{QAns}_{\mathsf{s}}(\mathsf{NC})$ for the problem that takes as input a database D, a set Σ of NCs (i.e., Σ does not contain any TGDs), a CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$, and asks whether $\bar{c} \in \mathsf{cert}_{\mathsf{s}}(q, D, \Sigma)$. Pinpointing the exact complexity of the above problems is the main goal of the present work. We are going to consider different complexity measures, that is, the combined complexity, the bounded-arity and fixed-program combined complexity, and the data complexity of $\mathsf{QAns}_{\mathsf{s}}(\mathsf{C}_\perp)$, which are defined in the obvious way. Each one of the next three sections focuses on one of the semantics in question.

5 ABox repair semantics

We first focus on $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{C}_\perp)$, where C is one of the classes of TGDs discussed above. The main result of this section follows:

Input: database D, set Σ of TGDs and NCs, CQ $q(\bar{x})$, and tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$
Output: accept, if $\bar{c} \notin \mathsf{cert}_{\mathsf{AR}}(q, D, \Sigma)$; otherwise, reject

guess a database $D' \subseteq D$
if there exists $\sigma \in \nu(\Sigma)$ such that $\mathsf{cert}(q_\sigma, D', \tau(\Sigma)) \neq \varnothing$ **then**
  return reject
foreach $\alpha \in D \setminus D'$ **do**
  if there is no $\sigma \in \nu(\Sigma)$ such that $\mathsf{cert}(q_\sigma, D' \cup \{\alpha\}, \tau(\Sigma)) \neq \varnothing$ **then**
    return reject
if $\bar{c} \in \mathsf{cert}(q, D', \tau(\Sigma))$ **then**
  return reject
else
  return accept

Algorithm 1: AlgorithmAR

Theorem 5.1. *The* t*-complexity of* $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{C}_\perp)$*, where* $\mathsf{t} \in \{\mathsf{c}, \mathsf{ba}, \mathsf{fp}, \mathsf{d}\}$ *and* $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$*, is as shown in Table 2.*

The rest of the section is devoted to establishing the above result. We first show, in Section 5.1, the upper bounds, and then, in Section 5.2, the lower bounds.

5.1 Upper bounds

Interestingly, all the upper bounds in Table 2 are obtained via the simple algorithm that checks whether there exists a repair that does not entail the given tuple of constants. The formal definition of this algorithm, called AlgorithmAR, is given in Algorithm 1. It is clear that this non-deterministic algorithm is correct. It is also easy to see that $\mathsf{AlgorithmAR}(D, \Sigma, q, \bar{c})$ runs in polynomial time, assuming access to an oracle that is powerful enough for solving the problem $\mathsf{QAns}(\mathsf{C})$, where C is the class of TGDs from which $\tau(\Sigma)$ is coming from. Therefore:

Lemma 5.1. *For a class* C *of TGDs,* $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{C}_\perp)$ *is in* $\mathrm{coNP}^{\mathcal{C}}$ *in* t*-complexity, where* $\mathsf{t} \in \{\mathsf{c}, \mathsf{ba}, \mathsf{fp}, \mathsf{d}\}$*, assuming that* $\mathsf{QAns}(\mathsf{C})$ *is in* $\mathcal{C}$ *in* t*-complexity.*

The desired upper bounds given in Table 2 are obtained from Propositions 3.1, 3.2, 3.3, and 3.4, which provide the t-complexity of $\mathsf{QAns}(\mathsf{C})$, for $\mathsf{t} \in \{\mathsf{c}, \mathsf{ba}, \mathsf{fp}, \mathsf{d}\}$, Lemma 5.1, and the following complexity facts:

$$\mathrm{coNP}^{\mathcal{C}} = \begin{cases} \mathrm{coNP} & \text{if } \mathcal{C} \subseteq \mathrm{PTime} \\ \Pi_2^P & \text{if } \mathcal{C} = \mathrm{coNP} \\ \mathcal{C} & \text{if } \mathcal{C} \in \{\mathrm{PSpace}, \mathrm{ExpTime}, 2\mathrm{ExpTime}\} \\ \mathrm{P}^{\mathrm{NExpTime}} & \text{if } \mathcal{C} = \mathrm{NExpTime}. \end{cases}$$

The first three facts are actually well-known. We only need to argue why the fourth one holds. The complexity class $\mathrm{NP}^{\mathrm{NExpTime}}$ lies at a higher level of the so-called strong exponential hierarchy. We know that the strong exponential hierarchy collapses to its Δ_2 level, which implies that $\mathrm{NP}^{\mathrm{NExpTime}} = \mathrm{P}^{\mathrm{NExpTime}}$ [26]. Observe that the class $\mathrm{P}^{\mathrm{NExpTime}}$ is a deterministic one, since the oracle machines in terms of which it is defined are deterministic, and therefore $\mathrm{coP}^{\mathrm{NExpTime}} = \mathrm{P}^{\mathrm{NExpTime}}$.

5.2 Lower bounds

We now proceed to establish the complexity lower bounds claimed in Table 2. In fact, the $\mathcal{C}$-hardness results, where $\mathcal{C} \in \{\mathrm{PSpace}, \mathrm{ExpTime}, 2\mathrm{ExpTime}\}$, are coming for free, since already $\mathsf{QAns}(\mathsf{C})$ is $\mathcal{C}$-hard. Therefore, to complete the picture, it suffices to establish the following results:

1. $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{A}_\perp)$ is $\mathrm{P}^{\mathrm{NExpTime}}$-hard in ba-complexity.
2. $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{NC})$ is Π_2^P-hard in fp-complexity.

3. $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{NC})$ is coNP-hard in d-complexity.

Notice that the last two statements refer only to negative constraints. The rest of the section is devoted to establishing the above three lower bounds.

Theorem 5.2. $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{A}_{\perp})$ *is* P^{NExpTime}*-hard in* ba*-complexity.*

Actually, the above result has been recently shown in [20] by relying on the construction, given in the proof of Lemma 3.2, for showing that $\mathsf{QAns}(\mathsf{A})$ is NExpTime-hard. As we will need it later, we recall the proof of this result, which relies on a $\mathrm{P}^{\text{NExpTime}}$-hard variation of the exponential tiling problem, introduced in [20]. An *extended tiling system* is a tuple $\mathcal{E} = (k, n, m, H_1, V_1, H_2, V_2)$, where k, n, and m are numbers in unary, and H_1, V_1, H_2, and V_2 are subsets of $\{1, \ldots, m\} \times \{1, \ldots, m\}$. We say that $\mathcal{E}$ is *valid* if the following holds: for every sequence s of length k of numbers from $\{1, \ldots, m\}$, there is no exponential tiling for the tiling system $\mathcal{T}_1 = (n, m, H_1, V_1, s)$, or there is an exponential tiling for the tiling system $\mathcal{T}_2 = (n, m, H_2, V_2, s)$. The *extended exponential tiling problem* follows:

PROBLEM :	ExtendedExpTiling
INPUT :	An extended tiling system $\mathcal{E}$.
QUESTION :	Is $\mathcal{E}$ valid?

We are now ready to recall the proof of Theorem 5.2 given in [20].

Proof of Theorem 5.2. Let $\mathcal{E} = (k, n, m, H_1, V_1, H_2, V_2)$ be an extended tiling system. We construct a database $D_{\mathcal{E}}$, and a set $\Sigma_{\mathcal{E}} \in \mathsf{A}_{\perp}$ that mentions only predicates of bounded arity, such that $\mathcal{E}$ is valid iff $\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), D_{\mathcal{E}}, \Sigma_{\mathcal{E}}) \neq \varnothing$.

The database $D_{\mathcal{E}}$. As expected, it stores the horizontal and vertical compatibility relations H_i and V_i, for $i \in \{1, 2\}$, respectively. In addition, it stores all the possible sequences of length k of numbers from $\{1, \ldots, m\}$. More precisely,

$$\begin{array}{rcl} D_{\mathcal{E}} & = & \{H_\ell(i,j) \mid (i,j) \in H_\ell\}_{\ell \in \{1,2\}} \;\cup\; \{V_\ell(i,j) \mid (i,j) \in V_\ell\}_{\ell \in \{1,2\}} \;\cup \\ & & \{S_i^1(j), S_i^2(j)\}_{i \in \{0,\ldots,k-1\}, j \in \{1,\ldots,m\}} \;\cup\; \{\mathrm{No}_1()\}. \end{array}$$

The atom $\mathrm{No}_1()$ is an auxiliary atom that will help us to check whether, for every sequence s of length k, there is no exponential tiling for (n, m, H_1, V_1, s).

The set of TGDs and NCs $\Sigma_{\mathcal{E}}$. We first add the following NCs. For each $i \in \{0, \ldots, k-1\}$, $j \in \{1, 2\}$, and $\ell, \ell' \in \{1, \ldots, m\}$ such that $\ell \neq \ell'$, we have

$$S_i^j(\ell), S_i^j(\ell') \rightarrow \perp.$$

For each $i \in \{0, \ldots, k-1\}$, and $\ell, \ell' \in \{1, \ldots, m\}$ such that $\ell \neq \ell'$, we also have

$$S_i^1(\ell), S_i^2(\ell') \rightarrow \perp.$$

The above set of NCs guarantees that in each repair exactly one atom of the form $S_i^j(\ell)$ occurs, for each $j \in \{1, 2\}$, where $\ell \in \{1, \ldots, m\}$. That is, in each repair, we keep a proper sequence s of length k of numbers from $\{1, \ldots, m\}$ such that $\mathcal{T}_i = (n, m, H_i, V_i, s)$, for each $i \in \{1, 2\}$, are proper tiling systems.

We then add the TGDs $\Sigma_{\mathcal{T}_1}$ and $\Sigma_{\mathcal{T}_2}$ that encode the tiling systems $\mathcal{T}_1$ and $\mathcal{T}_2$ as defined in the proof of Lemma 3.2; $\Sigma_{\mathcal{T}_1}$ and $\Sigma_{\mathcal{T}_2}$ use different predicates. Assuming that the final TGD of $\Sigma_{\mathcal{T}_i}$, which checks for the existence of an exponential tiling for $\mathcal{T}_i$, has in its head the atom $\mathrm{Yes}_i()$, we finally add the following NC and TGDs:

$$\mathrm{Yes}_1(), \mathrm{No}_1() \rightarrow \perp \qquad \mathrm{No}_1() \rightarrow \mathrm{Yes}() \qquad \mathrm{Yes}_2() \rightarrow \mathrm{Yes}().$$

The above NC ensures that $\mathrm{No}_1()$ appears in *every* repair iff the atom $\mathrm{Yes}_1()$ is not entailed, or, equivalently, for every sequence s of length k, the tiling system (n, m, H_1, V_1, s) does not admit an exponential tiling. It should then be clear that the following are equivalent:

1. For every $D \in \mathsf{reps}(D_{\mathcal{E}}, \Sigma_{\mathcal{E}})$, the atom $\mathrm{Yes}()$ occurs in $\mathsf{chase}(D, \tau(\Sigma_{\mathcal{E}}))$.

2. For every sequence s of length k of numbers from $\{1, \ldots, m\}$, there is no exponential tiling for $\mathcal{T}_1 = (n, m, H_1, V_1, s)$, or there is an exponential tiling for $\mathcal{T}_2 = (n, m, H_2, V_2, s)$, i.e., $\mathcal{E}$ is valid.

Hence, the above reduction is correct. It is also easy to verify that $D_{\mathcal{E}}$ and $\Sigma_{\mathcal{E}}$ can be constructed in polynomial time, and $\Sigma_{\mathcal{E}}$ mentions only predicates of bounded arity. This completes the proof of Theorem 5.2. $\square$

Theorem 5.3. $\mathsf{QAns_{AR}(NC)}$ *is* Π_2^P*-hard in* fp*-complexity.*

The proof of the above result exploits the satisfiability problem for quantified Boolean formulas with two alternations of quantifiers starting with universal quantifiers (2QBF$_\forall$). We actually consider formulas of the form

$$\varphi \;=\; \forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m \, \psi,$$

where $\psi = C_1 \wedge \cdots \wedge C_k$ is a 3CNF formula with C_i being a clause of the form $(\ell_i^1 \vee \ell_i^2 \vee \ell_i^3)$, while each literal ℓ_i^j is either a variable or the negation of a variable. The formula φ is *satisfiable* if, for every assignment of truth values to variables $x_1, \ldots, x_n$, there is an assignment of truth values to variables $y_1, \ldots, y_m$ such that ψ evaluates to true. The Π_2^P-hard problem of interest follows:

PROBLEM :	2QBF$_\forall$-SAT
INPUT :	A 2QBF$_\forall$ formula φ.
QUESTION :	Is φ satisfiable?

We are now ready to give the proof of Theorem 5.3.

Proof of Theorem 5.3. Given a 2QBF$_\forall$ formula φ as above, we construct a database D_φ, and a BCQ q_φ, such that φ is satisfiable iff $\mathsf{cert_{AR}}(q_\varphi, D_\varphi, \Sigma) \neq \varnothing$ with

$$\Sigma \;=\; \{S(x, \_, z), S(\_, x, z) \rightarrow \bot\},$$

where "_" denotes a "don't care" variable that occurs only once.

The database D_φ. Roughly, in D_φ we store, for each clause C, all the valuations that make C true. A valuation for C is a function f from the variables in C to $\{0, 1\}$. For a literal $\ell = x$ (resp., $\ell = \neg x$), $f(\ell) = f(x)$ (resp., $f(\ell) = \neg f(x)$). A valuation f satisfies a clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$ if $(f(\ell_1) \vee f(\ell_2) \vee f(\ell_3))$ evaluates to true. For a clause C, let $T(C)$ be the set of valuations for C that make C true. For a literal ℓ, we write $\mathsf{var}(\ell)$ for its variable. The database D_φ is defined as

$$\bigcup_{1 \leq i \leq k} \; \bigcup_{f \in T(C_i)} \; \bigcup_{1 \leq j \leq 3} \left\{ P_i^j \left(c_i^f, f(\mathsf{var}(\ell_i^j)) \right) \right\} \;\cup\; \bigcup_{1 \leq i \leq n} \{S(0, 1, d_i), S(1, 0, d_i)\}.$$

The atom $P_i^j(c_i^f, f(\mathsf{var}(\ell_i^j)))$ simply states the following: according to the valuation f, the variable of the literal ℓ_i^j of C_i takes the value $f(\mathsf{var}(\ell_i^j))$. The S-atoms are auxiliary atoms, and their purpose is explained below.

The query q_φ. Observe that the set $\mathsf{reps}(D, \Sigma)$ consists of all the subsets of D that can be formed by keeping either the atom $S(0, 1, d_i)$ or the atom $S(1, 0, d_i)$, for each $i \in \{1, \ldots, n\}$. In fact, each repair D' corresponds to a possible assignment $\mu_{D'}$ of truth values to the universally quantified variables of φ. More precisely, the atom $S(0, 1, d_i)$ (resp., $S(1, 0, d_i)$) states that the universally quantified variable x_i is assigned the value 0 (resp., 1). Therefore, it suffices to check whether, for every $D' \in \mathsf{reps}(D, \Sigma)$, there exists a valuation for φ, which is compatible with $\mu_{D'}$, that makes φ true. This is achieved via the Boolean CQ

$$q_\varphi \;=\; \bigwedge_{1 \leq i \leq k} \; \bigwedge_{1 \leq j \leq 3} P_i^j(z_i, \mathsf{var}(\ell_i^j)) \;\wedge\; \bigwedge_{1 \leq i \leq n} S(x_i, \_, d_i),$$

where all the variables occurring in q_φ are existentially quantified. This completes the proof of Theorem 5.3. $\square$

Theorem 5.4. $\mathsf{QAns_{AR}(NC)}$ *is* co*NP-hard in* d*-complexity.*

The above result relies on a variation of the unsatisfiability problem for Boolean formulas. A $(2+2)$ Boolean formula is a CNF formula where each clause has two positive and two negative literals. The coNP-hard problem of interest follows:

	c-complexity	ba-complexity	fp-complexity	d-complexity
$\mathsf{G}_\perp$	2ExpTime	ExpTime	Θ_2^P	coNP
$\mathsf{L}_\perp$	PSpace	Π_2^P	NP	in AC_0
$\mathsf{A}_\perp$	$\mathrm{P}^{\mathrm{NExpTime}}$	$\mathrm{P}^{\mathrm{NExpTime}}$	NP	in AC_0
$\mathsf{S}_\perp$	ExpTime	Π_2^P	NP	in AC_0

Table 3: Complexity of $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{C}_\perp)$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$. Recall that $\Theta_2^P = \mathrm{P}^{\mathrm{NP}[O(\log n)]}$, i.e., it collects all the problems that are solvable in polynomial time with logarithimically many calls to an NP-oracle. Apart from the AC_0 upper bounds, the rest are completeness results.

PROBLEM :	2+2UNSAT
INPUT :	A $(2+2)$ Boolean formula φ.
QUESTION :	Is φ unsatisfiable?

We are now ready to give the proof of Theorem 5.4.

Proof of Theorem 5.4. Given a $(2+2)$ formula $\varphi = C_1 \wedge \cdots \wedge C_n$ over the variables $x_1, \ldots, x_m$, we define the database D_φ as follows:

$$\{\mathrm{Pos}_1(i, x_i^1), \mathrm{Pos}_2(i, x_i^2), \mathrm{Neg}_1(i, x_i^3), \mathrm{Neg}_2(i, x_i^4) \mid C_i = x_i^1 \vee x_i^2 \vee x_i^3 \vee x_i^4\}$$
$$\cup\ \{\mathrm{True}(x_i), \mathrm{False}(x_i) \mid 1 \leq i \leq m\},$$

which essentially stores the formula φ, and also assigns to each variable in φ both the value 1 and the value 0. It is not difficult to verify that φ is unsatisfiable iff $\mathsf{cert}_{\mathsf{AR}}(q, D_\varphi, \Sigma) \neq \varnothing$, where

$$\Sigma \ = \ \{\mathrm{True}(x), \mathrm{False}(x) \rightarrow \bot\}$$

and q is the Boolean CQ

$$\exists x \exists y_1 \cdots \exists y_4\ (\mathrm{Pos}_1(x, y_1) \wedge \mathrm{False}(y_1) \wedge \mathrm{Pos}_2(x, y_2) \wedge \mathrm{False}(y_2) \wedge$$
$$\mathrm{Neg}_1(x, y_3) \wedge \mathrm{True}(y_3) \wedge \mathrm{Neg}_2(x, y_4) \wedge \mathrm{True}(y_4))\,.$$

It is clear that, for each variable x of φ, a repair of $\mathsf{reps}(D, \Sigma)$ keeps either the atom $\mathrm{True}(x)$ or the atom $\mathrm{False}(x)$, i.e., each $D' \in \mathsf{reps}(D, \Sigma)$ corresponds to a possible assignment $\mu_{D'}$ of truth values to the variables of φ. The query q checks that each such assignment evaluates φ to false, which is actually done by checking that at least one clause of φ evaluates to false. ☐

6 Intersection of repairs semantics

We now concentrate on $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{C}_\perp)$, where C is one of the classes of TGDs discussed above. The main result of this section follows:

Theorem 6.1. *The* t*-complexity of* $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{C}_\perp)$*, where* $\mathsf{t} \in \{\mathsf{c}, \mathsf{ba}, \mathsf{fp}, \mathsf{d}\}$ *and* $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$*, is as shown in Table 3.*

The rest of the section is devoted to establishing the above result. We first show, in Section 6.1, the upper bounds, and then, in Section 6.2, the lower bounds.

6.1 Upper bounds

Although for the ABox repair semantics we were able to establish all the upper bounds (see Table 2) in a uniform way via the algorithm AlgorithmAR, this is not the case for the intersection of repairs semantics. We are able, however, to partition the cells of Table 3 into four groups, and establish the upper bounds claimed in the cells of each such group in a uniform way. The four groups are as follows:

1. The c-complexity and the ba-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$, as well as the d-complexity for $\mathsf{G}_\perp$.

```
Input: database D, set Σ of TGDs and NCs, CQ q(x̄), and a tuple c̄ ∈ dom(D)^|x̄|
Output: accept, if c̄ ∉ cert_IAR(q, D, Σ); otherwise, reject

guess a database D* ⊆ D
foreach α ∈ D \ D* do
    guess a database D_α ⊆ D
    if α ∈ D_α then
        return reject
    else
        foreach β ∈ D \ D_α do
            if there is no σ ∈ ν(Σ) s.t. cert(q_σ, D_α ∪ {β}, τ(Σ)) ≠ ∅ then
                return reject
if c̄ ∈ cert(q, D*, τ(Σ)) then
    return reject
else
    return accept
```

Algorithm 2: AlgorithmIAR1

2. The fp-complexity for $\mathsf{G}_\perp$.
3. The fp-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$.
4. The d-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$.

We proceed to give more details for each of the above groups.

6.1.1 The c-complexity and the ba-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$, as well as the d-complexity for $\mathsf{G}_\perp$

The upper bounds are obtained via the simple procedure AlgorithmIAR1, depicted in Algorithm 2, that checks whether there exists a superset of the intersection of repairs that does not entail the given tuple $\bar{c}$ of constants. More precisely, the algorithm guesses a subset $D^\star$ of the input database D, and then checks that for every atom $\alpha \in D \setminus D^\star$, there exists $D_\alpha \in \mathsf{reps}(D, \Sigma)$, where Σ is the input set of TGDs and NCs, such that $\alpha \notin D_\alpha$, and thus, α is not in the intersection of repairs. This implies that $D^\star$ is a superset of the intersection of repairs. Finally, the algorithm rejects if $\bar{c} \in \mathsf{cert}(q, D^\star, \tau(\Sigma))$; otherwise, it accepts. Indeed, this algorithm is correct due to the following lemma:

Lemma 6.1. *Consider a database D, a set Σ of TGDs and NCs, a CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$. The following are equivalent:*

1. *$\bar{c} \notin \mathsf{cert}_{\mathsf{IAR}}(q, D, \Sigma)$.*
2. *There exists $D^\star \supseteq \bigcap_{D' \in \mathsf{reps}(D,\Sigma)} D'$ such that $\bar{c} \notin \mathsf{cert}(q, D^\star, \tau(\Sigma))$.*

Proof. The fact that (1) implies (2) holds trivially with $D^\star$ being exactly the intersection of repairs. The other direction follows by the monotonicity of the CQ q, i.e., for every two instances I_1 and I_2, $I_1 \subseteq I_2$ implies $q(I_1) \subseteq q(I_2)$. □

It is also easy to see that the non-deterministic algorithm AlgorithmIAR1 runs in polynomial time, assuming access to an oracle that can solve $\mathsf{QAns}(\mathsf{C})$, where C is the class of TGDs from which the input set of TGDs is coming from. Therefore:

Lemma 6.2. *For a class C of TGDs, $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{C}_\perp)$ is in $\mathrm{coNP}^{\mathcal{C}}$ in t-complexity, where $\mathsf{t} \in \{\mathsf{c}, \mathsf{ba}, \mathsf{d}\}$, assuming that $\mathsf{QAns}(\mathsf{C})$ is in $\mathcal{C}$ in t-complexity.*

Since, by Lemma 6.1, AlgorithmIAR1 is correct, the desired upper bounds for Group 1 given in Table 3 are obtained from Propositions 3.1, 3.2, 3.3 and 3.4, Lemma 6.2, and the following complexity facts that

Input: database D, set Σ of TGDs and NCs, CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$
Output: accept if $\bar{c} \in \mathsf{cert}_{\mathsf{IAR}}(q, D, \Sigma)$; otherwise, reject

```
D⋆ := D
foreach α ∈ D do
    if there exists D_α ∈ reps(D, Σ) such that α ∉ D_α then
        D⋆ := D⋆ \ {α}
if c̄ ∈ cert(q, D⋆, τ(Σ)) then
    return accept
else
    return reject
```

Algorithm 3: AlgorithmIAR2

have been already discussed in the previous section and we recall again for the sake of readability:

$$\mathrm{coNP}^{\mathcal{C}} \;=\; \begin{cases} \mathrm{coNP} & \text{if } \mathcal{C} \subseteq \mathrm{PT{\small IME}} \\ \Pi_2^P & \text{if } \mathcal{C} = \mathrm{coNP} \\ \mathcal{C} & \text{if } \mathcal{C} \in \{\mathrm{PS{\small PACE}}, \mathrm{E{\small XP}T{\small IME}}, \mathrm{2E{\small XP}T{\small IME}}\} \\ \mathrm{P}^{\mathrm{NE{\small XP}T{\small IME}}} & \text{if } \mathcal{C} = \mathrm{NE{\small XP}T{\small IME}}. \end{cases}$$

6.1.2 The fp-complexity for $\mathsf{G}_\perp$

We need to establish a $\Theta_2^P = \mathrm{P}^{\mathrm{NP}[O(\log n)]}$ upper bound. To this end, we exploit the procedure AlgorithmIAR2, depicted in Algorithm 3, that constructs the intersection of repairs $D^\star$, and accepts if the given tuple $\bar{c}$ belongs to $\mathsf{cert}(q, D^\star, \tau(\Sigma))$; otherwise, it rejects. The intersection of repairs $D^\star$ is constructed by starting from the input database D, and removing all the atoms α for which there exists at least one repair $D_\alpha \in \mathsf{reps}(D, \Sigma)$ such that $\alpha \notin D_\alpha$. More precisely, $D^\star$ is constructed via polynomially many parallel calls to an NP-oracle. In fact, for each atom $\alpha \in D$, we call in parallel an NP-oracle that does the following:

1. Guess a database $D_\alpha \subseteq D$.
2. If $\alpha \in D_\alpha$, then reject.
3. For each atom $\beta \in D \setminus D_\alpha$, if there is no $\sigma \in \nu(\Sigma)$ such that $\mathsf{cert}(q_\sigma, D_\alpha \cup \{\beta\}, \tau(\Sigma)) \neq \varnothing$, then return reject; otherwise; return accept.

It is crucial to clarify that the check whether $\mathsf{cert}(q_\sigma, D_\alpha \cup \{\beta\}, \tau(\Sigma)) \neq \varnothing$ is feasible in polynomial time since q_σ and $\tau(\Sigma)$ are fixed – recall that we are studying the fp-complexity of $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\perp)$, while, by Proposition 3.1, $\mathsf{QAns}(\mathsf{G})$ is in PTIME in d-complexity. Therefore, the above oracle in indeed an NP-oracle. It is clear that, for an atom $\alpha \in D$, if the above oracle returns accept, then there exists a repair D_α such that $\alpha \notin D_\alpha$, and thus, α does not belong to the intersection of repairs. Consequently, the intersection of repairs $D^\star$ is constructed by simply removing from D all the atoms α for which the oracle returns accept. Since $D^\star$ can be constructed in polynomial time via parallel NP-oracle calls, we can conclude that it can also be constructed in polynomial time via logarithimically many NP-oracle calls; see, e.g., [42]. Once we have $D^\star$ in place, we need one more call to an NP-oracle for checking whether $\bar{c} \in \mathsf{cert}(q, D^\star, \tau(\Sigma))$; the latter is indeed in NP since, by Proposition 3.1, $\mathsf{QAns}(\mathsf{G})$ is in NP in fp-complexity. Hence, we get the desired $\Theta_2^P = \mathrm{P}^{\mathrm{NP}[O(\log n)]}$ upper bound.

6.1.3 The fp-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$

We need to establish an NP upper bound. A crucial notion that we are going to exploit is that of culprit, which is essentially a minimal inconsistent subset of a database D w.r.t. a set Σ of TGDs and NCs. Formally, a *culprit* of D w.r.t. Σ is a subset D' of D such that the following conditions hold:

1. $\mathsf{mods}(D', \Sigma) = \varnothing$, and

Input: database D, set Σ of TGDs and NCs, CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$
Output: accept if $\bar{c} \in \mathsf{cert}_{\mathsf{IAR}}(q, D, \Sigma)$; otherwise, reject

$D^\star := D \setminus \bigcup_{D' \in \mathsf{culprit}(D,\Sigma)} D'$
if $\bar{c} \in \mathsf{cert}(q, D^\star, \tau(\Sigma))$ **then**
  return accept
else
  return reject

Algorithm 4: AlgorithmIAR3

2. there is no $D'' \subsetneq D'$ such that $\mathsf{mods}(D'', \Sigma) = \varnothing$.

Intuitively, a culprit is a minimal subset of D that, together with $\tau(\Sigma)$, entails some NC $\sigma \in \nu(\Sigma)$; a culprit for σ is a "minimal explanation" [14] of q_σ. We write $\mathsf{culprit}(D, \Sigma)$ for the set of all culprits of D w.r.t. Σ. By deleting from D a *minimal* set of facts S intersecting *all* culprits from $\mathsf{culprit}(D, \Sigma)$,[6] we obtain a repair $R = D \setminus S$. By this, it is an easy exercise to show that the intersection of the repairs of D w.r.t. Σ is precisely the subset of D obtained after eliminating its culprits (w.r.t. Σ):

Lemma 6.3. *Consider a database D, and a set Σ of TGDs and NCs. It holds that*

$$\bigcap_{D' \in \mathsf{reps}(D,\Sigma)} D' \;=\; D \setminus \bigcup_{D' \in \mathsf{culprit}(D,\Sigma)} D'.$$

From the above lemma, we immediately get the algorithm AlgorithmIAR3, depicted in Algorithm 4, that explicitly constructs the intersection of repairs $D^\star$, and accepts if the given tuple $\bar{c}$ belongs to $\mathsf{cert}(q, D^\star, \tau(\Sigma))$; otherwise, it rejects. It remains to argue how we get the desired NP upper bounds. To this end, it suffices to show that $\mathsf{culprit}(D, \Sigma)$ can be computed in polynomial time when the set Σ is fixed and falls in $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$. In such a case, $D^\star$ can be computed in polynomial time, and the claim follows by Propositions 3.2, 3.3 and 3.4, which state that $\mathsf{QAns}(\mathsf{C})$ for $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$ is in NP in fp-complexity.

The key property of the classes in question that we are going to exploit is *UCQ-rewritability* shown in [24]: given a set of TGDs $\Sigma \in \mathsf{C}$ and a CQ $q(\bar{x})$, we can construct a UCQ $Q_{q,\Sigma}(\bar{x})$ such that, for *every* database D and tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$,

$$\bar{c} \in \mathsf{cert}(q, D, \Sigma) \iff \bar{c} \in Q_{q,\Sigma}(D).$$

Consider now a set $\Sigma \in \mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$. By exploiting UCQ-rewritability, we can first "embed" the TGDs of $\tau(\Sigma)$ into the NCs of $\nu(\Sigma)$, which leads to a set of NCs denoted $\langle \Sigma \rangle$, and then compute the culprits of a database D w.r.t. Σ by collecting all the images of the so-called minimal specializations of the NCs of $\langle \Sigma \rangle$ in D via injective mappings. Let us formalize the above discussion.

- Let Q_Σ be the UCQ $\bigcup_{\sigma \in \nu(\Sigma)} Q_{q_\sigma, \tau(\Sigma)}$; recall that q_σ denotes the Boolean CQ that corresponds to σ. The *embedding* of $\tau(\Sigma)$ into $\nu(\Sigma)$ is the set

$$\langle \Sigma \rangle \;=\; \{\phi(\bar{x}) \rightarrow \perp \mid \exists \bar{x}\, \phi(\bar{x}) \in Q_\Sigma\}.$$

- A *specialization* of a NC σ is a NC obtained from σ by identifying some of the variables occurring in $\mathsf{body}(\sigma)$. For example, $R(x, y, x), S(x) \rightarrow \perp$ is a specialization of $R(x, y, z), S(z) \rightarrow \perp$ obtained by identifying x and z. Notice that a NC is trivially a specialization of itself. We write $\mathsf{sp}(\sigma)$ for the set of all specializations of a NC σ, and for a set of NCs Σ' we define $\mathsf{sp}(\Sigma')$ as the set $\bigcup_{\sigma \in \Sigma'} \mathsf{sp}(\sigma)$. Moreover, we define $\mathsf{msp}(\Sigma')$ as the largest subset of $\mathsf{sp}(\Sigma')$ such that, for every $\sigma \in \mathsf{msp}(\Sigma')$, there is no $\sigma' \in \mathsf{msp}(\Sigma')$ with $\mathsf{body}(\sigma') \subsetneq \mathsf{body}(\sigma)$ (up to variable renaming). Clearly, $\mathsf{msp}(\Sigma')$ is unique (up to variable renaming). Finally, for a database D, let

$$I_{D,\Sigma} \;=\; \{h(\mathsf{body}(\sigma)) \mid \sigma \in \mathsf{msp}(\langle \Sigma \rangle) \text{ and } h \text{ is an injective homomorphism from } \mathsf{body}(\sigma) \text{ to } D\}.$$

[6] Such a deletion S is a (hypergraph) transversal of $\mathsf{culprit}(D, \Sigma)$, and hence S is minimal iff, for each fact $f \in S$, there is a culprit $C \in \mathsf{culprit}(D, \Sigma)$ such that $C \cap S = \{f\}$ [16, 22].

We proceed to show the following:

Lemma 6.4. *For a database D and a set $\Sigma \in \mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, it holds that* $\mathsf{culprit}(D, \Sigma) = I_{D,\Sigma}$.

Proof. ($\subseteq$) Consider an arbitrary $C \in \mathsf{culprit}(D, \Sigma)$. There is $\sigma \in \langle\Sigma\rangle$ and a homomorphism h from $\mathsf{body}(\sigma)$ to C, and for every $\sigma' \in \langle\Sigma\rangle$, there is no homomorphism from $\mathsf{body}(\sigma')$ to a strict subset of C. Let σ_h be the NC with $\mathsf{body}(\sigma_h)$ being the conjunction of atoms obtained from C by converting each constant c into a variable x_c. We claim that $\sigma_h \in \mathsf{msp}(\langle\Sigma\rangle)$ (up to variable renaming), which in turn implies that $C \in I_{D,\Sigma}$. By contradiction, assume that this is not the case. It is clear that σ_h is a specialization of σ (up to variable renaming), and thus $\sigma_h \in \mathsf{sp}(\langle\Sigma\rangle)$ (up to variable renaming). This implies that there exists $\sigma' \in \mathsf{sp}(\langle\Sigma\rangle)$ such that $\mathsf{body}(\sigma') \subsetneq \mathsf{body}(\sigma)$ (up to variable renaming). Therefore the NC $\sigma'' \in \langle\Sigma\rangle$ from which σ' is obtained via specialization can be mapped via a homomorphism to a strict subset of C, which is a contradiction.

($\supseteq$) Consider now an arbitrary $C \in I_{D,\Sigma}$. By definition, there exists $\hat{\sigma} \in \mathsf{msp}(\langle\Sigma\rangle)$ and an injective mapping $\hat{h}$ that maps $\mathsf{body}(\hat{\sigma})$ to C. By contradiction, assume that $C \notin \mathsf{culprit}(D, \Sigma)$. Suppose that $\hat{\sigma}$ is the specialized version of $\sigma \in \langle\Sigma\rangle$. Clearly, there exists a homomorphism from $\mathsf{body}(\sigma)$ to C. Hence, the fact that $C \notin \mathsf{culprit}(D, \Sigma)$ allows us to conclude that there exists $\sigma' \in \langle\Sigma\rangle$ and a homomorphism h' from $\mathsf{body}(\sigma')$ to a strict subset C' of C. Consider the NC σ'' with $\mathsf{body}(\sigma'')$ being the conjunction of atoms obtained from C' by converting each constant c into a variable x_c. Clearly, $\sigma'' \in \mathsf{sp}(\langle\Sigma\rangle)$ (up to variable renaming). But since $\hat{h}(\mathsf{body}(\hat{\sigma})) = C$ with $\hat{h}$ being an injective homomorphism, we can conclude that $\mathsf{body}(\sigma'') \subsetneq \mathsf{body}(\hat{\sigma})$ (up to variable renaming). But this contradicts the fact $\hat{\sigma} \in \mathsf{msp}(\langle\Sigma\rangle)$. Therefore, $C \in \mathsf{culprit}(D, \Sigma)$, and the claim follows. ☐

The fact that for a fixed set $\Sigma \in \mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, the culprits of a database D w.r.t. Σ can be computed in polynomial time is an easy consequence of Lemma 6.4. Indeed, we can compute in constant time the set of NCs $\mathsf{msp}(\langle\Sigma\rangle)$. This in turn allows us to compute the set of databases $I_{D,\Sigma}$ in polynomial time in the size of D, which, by Lemma 6.4, coincides with $\mathsf{culprit}(D, \Sigma)$.

6.1.4 The d-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$

Our goal is to establish an AC_0 upper bound. To this end, we are going to show the following technical result, which essentially states that computing the IAR-certain answers to a CQ in the case of linear, acyclic and sticky sets of TGDs boils down to evaluating a first-order query over the input database.

Lemma 6.5. *Consider a set of TGDs and NCs $\Sigma \in \mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, and a CQ $q(\bar{x})$. We can construct a first-order query $\Phi_{q,\Sigma}(\bar{x})$ such that, for every database D,* $\mathsf{cert}_{\mathsf{IAR}}(q, D, \Sigma) = \Phi_{q,\Sigma}(D)$.

Having the above lemma in place, it is straightforward to obtain the desired AC_0 upper bound. The key fact is that the first-order query $\Phi_{q,\Sigma}$ does not depend on the input database. Therefore, for a fixed set Σ of TGDs and NCs, and a fixed CQ $q(\bar{x})$, we can construct the first-order query $\Phi_{q,\Sigma}(\bar{x})$ in constant time. Then, given a database D and a tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$, we simply need to check whether $\bar{c} \in \Phi_{q,\Sigma}(D)$. The latter is in AC_0 since first-order query evaluation is in AC_0 in data complexity. It remains to establish Lemma 6.5.

We first establish the following useful auxiliary result:

Lemma 6.6. *Consider a set of TGDs and NCs $\Sigma \in \mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, and a CQ $q(\bar{x})$. We can construct a first-order query $\Psi_{q,\Sigma}(\bar{x})$ such that, for every database D, $q\left(D \setminus \bigcup_{D' \in \mathsf{culprit}(D,\Sigma)} D'\right) = \Psi_{q,\Sigma}(D)$.*

Proof. We assume that $q(\bar{x}) = \exists \bar{y}\, \phi(\bar{x}, \bar{y})$. We also assume, w.l.o.g., that the NCs of $\langle\Sigma\rangle$ and q do not share variables. We define $\Psi_{q,\Sigma}(\bar{x})$ as

$$\exists y \left(\phi(\bar{x}, \bar{y}) \wedge \bigwedge_{\psi(\bar{z}) \rightarrow \perp \in \mathsf{msp}(\langle\Sigma\rangle)} \forall \bar{z} \left(\psi(\bar{z}) \wedge \bigwedge_{v,w \in \bar{z}, v \neq w} v \neq w \rightarrow \bigwedge_{\alpha \in \phi(\bar{x},\bar{y}), \beta \in \psi(\bar{z})} \alpha \neq \beta \right) \right).$$

Given a database D, $\Psi_{q,\Sigma}(D)$ is the set of tuples $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$ such that $q(\bar{c})$ is mapped to D via a homomorphism h, while $h(D)$ does not contain an atom of $I_{D,\Sigma}$, and thus, by Lemma 6.4, $h(D)$ does not contain an atom of $\mathsf{culprit}(D, \Sigma)$. In other words, $\Psi_{q,\Sigma}(D) = q(D \setminus \bigcup_{D' \in \mathsf{culprit}(D,\Sigma)} D')$, and the claim follows. ☐

We are now ready to give the proof of Lemma 6.5. We claim that the desired first-order query $\Phi_{q,\Sigma}(\bar{x})$ is the query

$$\bigvee_{q' \in Q_{q,\tau(\Sigma)}} \Psi_{q',\Sigma}(\bar{x}),$$

where $\Psi_{q',\Sigma}(\bar{x})$ is the first-order query provided by Lemma 6.6. Given a database D, by Lemma 6.3 and UCQ-rewritability, we get that

$$\mathsf{cert}_{\mathsf{IAR}}(q, D, \Sigma) \;=\; \bigcup_{q' \in Q_{q,\tau(\Sigma)}} q' \left(D \setminus \bigcup_{D' \in \mathsf{culprit}(D,\Sigma)} D' \right).$$

Since $\Psi_{q',\Sigma}(D)$ is precisely $q' \left(D \setminus \bigcup_{D' \in \mathsf{culprit}(D,\Sigma)} D' \right)$, the claim follows.

6.2 Lower bounds

We now proceed to establish the complexity lower bounds claimed in Table 3. The $\mathcal{C}$-hardness results, where $\mathcal{C} \in \{\mathrm{NP}, \mathrm{PSPACE}, \mathrm{EXPTIME}, 2\mathrm{EXPTIME}\}$, are coming for free since already $\mathsf{QAns}(\mathsf{C})$ is $\mathcal{C}$-hard. Therefore, to complete the picture, it suffices to establish the following hardness results:

1. $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{A}_\perp)$ is $\mathrm{P}^{\mathrm{NEXPTIME}}$-hard in ba-complexity.
2. $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{NC})$ is Π_2^P-hard in ba-complexity.
3. $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\perp)$ is Θ_2^P-hard in fp-complexity.
4. $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\perp)$ is coNP-hard in d-complexity.

Notice that the second statement refers only to negative constraints. The rest of the section is devoted to establishing the above four lower bounds.

Theorem 6.2. $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{A}_\perp)$ *is* $P^{NEXPTIME}$*-hard in* ba*-complexity.*

Proof. The proof is via a reduction from the extended exponential tiling problem, which has been used for showing Theorem 5.2, that is, the $\mathrm{P}^{\mathrm{NEXPTIME}}$-hardness in ba-complexity of $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{A}_\perp)$. In fact, the proof is an adaptation of the proof of Theorem 5.2. Recall that in the proof of Theorem 5.2, given an exponential tiling system $\mathcal{E} = (k, n, m, H_1, V_1, H_2, V_2)$, we construct the database

$$\begin{aligned} D_{\mathcal{E}} \quad = \quad & \{H_\ell(i,j) \mid (i,j) \in H_\ell\}_{\ell \in \{1,2\}} \;\cup\; \{V_\ell(i,j) \mid (i,j) \in V_\ell\}_{\ell \in \{1,2\}} \;\cup \\ & \{S_i^1(j), S_i^2(j)\}_{i \in \{0,\ldots,k-1\}, j \in \{1,\ldots,m\}} \;\cup\; \{\mathrm{No}_1()\} \end{aligned}$$

and a set $\Sigma_{\mathcal{E}} \in \mathsf{A}_\perp$, which mentions only predicates of bounded arity, such that

$$\mathcal{E} \text{ is valid} \iff \mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), D_{\mathcal{E}}, \Sigma_{\mathcal{E}}) \neq \varnothing. \tag{3}$$

For showing Theorem 6.2, we adapt $D_{\mathcal{E}}$ and $\Sigma_{\mathcal{E}}$ as follows:

$$\begin{aligned} \hat{D}_{\mathcal{E}} \;&=\; D_{\mathcal{E}} \;\cup\; \{\mathrm{Yes}(), \mathrm{No}_2()\} \\ \hat{\Sigma}_{\mathcal{E}} \;&=\; \Sigma_{\mathcal{E}} \;\cup \\ & \quad\; \{\phi(\bar{x}), \mathrm{Yes}(), \mathrm{No}() \rightarrow \perp, \\ & \quad\; \mathrm{Yes}_2(), \mathrm{No}_2() \rightarrow \perp, \\ & \quad\; \mathrm{Yes}_1(), \mathrm{No}_2() \rightarrow \mathrm{No}()\}, \end{aligned}$$

where $\phi(\bar{x})$ is the conjunction of atoms

$$\bigwedge_{\ell \in \{1,2\}} \left(\bigwedge_{(i,j) \in H_\ell} H_\ell(i,j) \;\wedge\; \bigwedge_{(i,j) \in V_\ell} V_\ell(i,j) \;\wedge\; \bigwedge_{i \in \{0,\ldots,k-1\}} S_i^\ell(x_i) \right).$$

We first show that

$$\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), D_{\mathcal{E}}, \Sigma_{\mathcal{E}}) \neq \varnothing \iff \mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), \hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing. \tag{4}$$

For the direction ($\Rightarrow$), assume that $\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), \hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}}) = \varnothing$. This implies that there exists $D' \in \mathsf{reps}(\hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}})$ such that $\mathsf{cert}(\mathrm{Yes}(), D', \hat{\Sigma}_{\mathcal{E}}) = \varnothing$. It is clear that $\mathrm{Yes}() \notin D'$. Observe that $\mathsf{cert}(q, D', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$, where q is the Boolean CQ $\exists \bar{x}\, \phi(\bar{x}), \mathrm{No}()$, since otherwise $\mathsf{mods}(D' \cup \mathrm{Yes}(), \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$, which contradicts the fact that $D' \in \mathsf{reps}(\hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}})$. This in turn implies $\mathsf{cert}(\mathrm{Yes}_1(), D', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$. Let $D'' = D' \setminus \{\mathrm{No}_2()\}$. We proceed to show that $D'' \in \mathsf{reps}(D_{\mathcal{E}}, \Sigma_{\mathcal{E}})$ and $\mathsf{cert}(\mathrm{Yes}(), D'', \Sigma_{\mathcal{E}}) = \varnothing$, which in turn implies that $\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), D_{\mathcal{E}}, \Sigma_{\mathcal{E}}) = \varnothing$, as needed. It is clear that $D'' \subseteq D_{\mathcal{E}}$. Moreover, since $D'' \subseteq D'$ and $\Sigma_{\mathcal{E}} \subseteq \hat{\Sigma}_{\mathcal{E}}$, we immediately get that $\mathsf{cert}(\mathrm{Yes}(), D'', \Sigma_{\mathcal{E}}) = \varnothing$ (since $\mathsf{cert}(\mathrm{Yes}(), D', \hat{\Sigma}_{\mathcal{E}}) = \varnothing$), and $\mathsf{mods}(D'', \Sigma_{\mathcal{E}}) \neq \varnothing$ (since $\mathsf{mods}(D', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$). To show that $D'' \in \mathsf{reps}(D_{\mathcal{E}}, \Sigma_{\mathcal{E}})$, it remains to show that there is no $\alpha \in D_{\mathcal{E}} \setminus D''$ such that $\mathsf{mods}(D'' \cup \{\alpha\}, \Sigma_{\mathcal{E}}) \neq \varnothing$. We first observe that in $D_{\mathcal{E}} \setminus D''$ we only have atoms of the form $S_i^j(\ell)$ for $i \in \{0, \ldots, k-1\}$, $j \in \{1, 2\}$ and $\ell \in \{1, \ldots, m\}$, and the atom $\mathrm{No}_1()$. It is clear that adding to D'' an atom α of the form $S_i^j(\ell)$ leads to inconsistency, i.e., $\mathsf{mods}(D'' \cup \{\alpha\}, \Sigma_{\mathcal{E}}) = \varnothing$. Observe now that $\mathsf{cert}(\mathrm{Yes}_1(), D', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$ implies $\mathsf{cert}(\mathrm{Yes}_1(), D'', \Sigma_{\mathcal{E}}) \neq \varnothing$ since the facts (resp., the TGDs and NCs) in $D' \setminus D''$ (resp., $\hat{\Sigma}_{\mathcal{E}} \setminus \Sigma_{\mathcal{E}}$) do not affect the entailment of $\mathrm{Yes}_1()$. Therefore, $\mathsf{cert}(\mathrm{Yes}_1(), D'' \cup \{\mathrm{No}_1()\}, \Sigma_{\mathcal{E}}) \neq \varnothing$, which in turn implies that $\mathsf{mods}(D'' \cup \{\mathrm{No}_1()\}, \Sigma_{\mathcal{E}}) = \varnothing$. We conclude that $D'' \in \mathsf{reps}(D_{\mathcal{E}}, \Sigma_{\mathcal{E}})$, and the claim follows.

For the direction ($\Leftarrow$), assume that $\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), D_{\mathcal{E}}, \Sigma_{\mathcal{E}}) = \varnothing$, which, by equivalence (3), implies that $\mathcal{E}$ is not valid. This means that there exists $D' \in \mathsf{reps}(D_{\mathcal{E}}, \Sigma_{\mathcal{E}})$ such that $\mathsf{cert}(\mathrm{Yes}_1(), D', \Sigma_{\mathcal{E}}) \neq \varnothing$ and $\mathsf{cert}(\mathrm{Yes}_2(), D', \Sigma_{\mathcal{E}}) = \varnothing$. Observe that $\mathsf{cert}(\mathrm{No}_1(), D', \Sigma_{\mathcal{E}}) = \varnothing$ because otherwise $\mathsf{mods}(D', \Sigma_{\mathcal{E}}) = \varnothing$, which contradicts the fact that $D' \in \mathsf{reps}(D_{\mathcal{E}}, \Sigma_{\mathcal{E}})$. Let $D'' = D' \cup \{\mathrm{No}_2()\}$. We proceed to show that $D'' \in \mathsf{reps}(\hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}})$ and $\mathsf{cert}(\mathrm{Yes}(), D'', \hat{\Sigma}_{\mathcal{E}}) = \varnothing$, which in turn implies that $\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), \hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}}) = \varnothing$, as needed. It is clear that $D'' \subseteq \hat{D}_{\mathcal{E}}$. Moreover, it can be easily verified that adding $\mathrm{No}_2()$ to D', and adding $\hat{\Sigma}_{\mathcal{E}} \setminus \Sigma_{\mathcal{E}}$ to $\Sigma_{\mathcal{E}}$, do not affect the non-entailment of $\mathrm{No}_1()$ and $\mathrm{Yes}_2()$, and thus, $\mathsf{cert}(\mathrm{No}_1(), D'', \hat{\Sigma}_{\mathcal{E}}) = \varnothing$ and $\mathsf{cert}(\mathrm{Yes}_2(), D'', \hat{\Sigma}_{\mathcal{E}}) = \varnothing$. As a consequence, we have that $\mathsf{cert}(\mathrm{Yes}(), D'', \hat{\Sigma}_{\mathcal{E}}) = \varnothing$. The last three statements allow us also to conclude that $\mathsf{mods}(D'', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$. To show that $D'' \in \mathsf{reps}(\hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}})$, it remains to show that there is no $\alpha \in \hat{D}_{\mathcal{E}} \setminus D''$ such that $\mathsf{mods}(D'' \cup \{\alpha\}, \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$. Notice that in $\hat{D}_{\mathcal{E}} \setminus D''$ we only have atoms of the form $S_i^j(\ell)$ for $i \in \{0, \ldots, k-1\}$, $j \in \{1, 2\}$ and $\ell \in \{1, \ldots, m\}$, and the atoms $\mathrm{No}_1()$ and $\mathrm{Yes}()$. It is clear that adding to D'' an atom α of the form $S_i^j(\ell)$ leads to inconsistency, i.e., $\mathsf{mods}(D'' \cup \{\alpha\}, \hat{\Sigma}_{\mathcal{E}}) = \varnothing$. Since $\mathsf{cert}(\mathrm{Yes}_1(), D', \Sigma_{\mathcal{E}}) \neq \varnothing$, $D' \subseteq D''$, and $\Sigma_{\mathcal{E}} \subseteq \hat{\Sigma}_{\mathcal{E}}$, we get that $\mathsf{cert}(\mathrm{Yes}_1(), D'', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$, which implies that $\mathsf{mods}(D'' \cup \{\mathrm{No}_1()\}, \hat{\Sigma}_{\mathcal{E}}) = \varnothing$. Finally, we show that $\mathsf{cert}(\mathrm{No}(), D'', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$ and $\mathsf{cert}(q, D'', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$, where q is the Boolean CQ $\exists \bar{x}\, \phi(\bar{x})$, which imply that $\mathsf{mods}(D'' \cup \{\mathrm{Yes}()\}, \hat{\Sigma}_{\mathcal{E}}) = \varnothing$, as needed. The fact that $\mathsf{cert}(\mathrm{No}(), D'', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$ follows from $\mathsf{cert}(\mathrm{No}_2(), D'', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$ since $\mathrm{No}_2() \in D''$, and $\mathsf{cert}(\mathrm{Yes}_1(), D'', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$ since $\mathsf{cert}(\mathrm{Yes}_1(), D', \Sigma_{\mathcal{E}}) \neq \varnothing$, $D' \subseteq D''$ and $\Sigma_{\mathcal{E}} \subseteq \hat{\Sigma}_{\mathcal{E}}$. The fact that $\mathsf{cert}(q, D'', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$ follows from $\mathsf{cert}(q, D', \Sigma_{\mathcal{E}}) \neq \varnothing$, $D' \subseteq D''$ and $\Sigma_{\mathcal{E}} \subseteq \hat{\Sigma}_{\mathcal{E}}$.

It is also not difficult to verify that

$$\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), \hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing \iff \mathsf{cert}_{\mathsf{IAR}}(\mathrm{Yes}(), \hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing. \tag{5}$$

The direction ($\Leftarrow$) holds trivially, since the IAR semantics is an approximation of the AR semantics. For the direction ($\Rightarrow$), we observe that, for each repair $D' \in \mathsf{reps}(\hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}})$ such that $\mathsf{cert}(\mathrm{Yes}(), D', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$, it holds that $\mathrm{Yes}() \in D'$. Since, by hypothesis, for each $D' \in \mathsf{reps}(\hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}})$, we have that $\mathsf{cert}(\mathrm{Yes}(), D', \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$, we conclude that each repair of $\mathsf{reps}(\hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}})$ contains the atom $\mathrm{Yes}()$. Therefore, $\bigcap_{D' \mathsf{reps}(\hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}})} D'$ contains $\mathrm{Yes}()$, and $\mathsf{cert}_{\mathsf{IAR}}(\mathrm{Yes}(), \hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing$ follows.

By putting together the equivalences (3), (4), and (5), we conclude that

$$\mathcal{E} \text{ is valid} \iff \mathsf{cert}_{\mathsf{IAR}}(\mathrm{Yes}(), \hat{D}_{\mathcal{E}}, \hat{\Sigma}_{\mathcal{E}}) \neq \varnothing,$$

which shows the correctness of our reduction, and Theorem 6.2 follows. $\square$

Theorem 6.3. $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{NC})$ *is* Π_2^P*-hard in* ba*-complexity.*

The proof of the above result exploits a variant of $\mathsf{2QBF}_\forall\text{-}\mathsf{SAT}$. A $2\mathrm{NQBF}_\forall$ formula is a $2\mathrm{QBF}_\forall$ formula

$$\varphi = \forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m\ \psi,$$

where ψ is a 3CNF formula of the form

$$\bigwedge_{1 \leq i \leq k} C_i \wedge \bigwedge_{1 \leq i \leq n} (C_i^+ \wedge C_i^-)$$

with $C_i = (\ell_i^1 \vee \ell_i^2 \vee \ell_i^3)$, $C_i^+ = (x_i \vee \neg y_{t(i)})$ and $C_i^- = (\neg x_i \vee y_{t(i)})$ for some $t(i) \in \{1, \ldots, m\}$, and for each $i \in \{1, \ldots, n\}$, the universally quantified variable x_i occurs only in the clauses C_i^+ and C_i^-. In other words, the latter says that, for each $i \in \{1, \ldots, n\}$, the truth value of the variable $y_{t(i)}$ is determined by the value of x_i. The Π_2^P-hard problem of interest follows [25, 45]:

PROBLEM :	$\mathsf{2NQBF}_\forall$-SAT
INPUT :	A $\mathrm{2NQBF}_\forall$ formula φ.
QUESTION :	Is φ satisfiable?

We are now ready to give the proof of Theorem 6.3.

Proof of Theorem 6.3. For a $\mathrm{2NQBF}_\forall$ formula φ as defined above, our goal is to construct a database D_φ and a set $\Sigma_\varphi \in \mathsf{NC}$, which mention only predicates of bounded arity, such that φ is satisfiable iff $\mathsf{cert}_{\mathsf{IAR}}(\mathrm{Sat}(), D_\varphi, \Sigma_\varphi) \neq \varnothing$.

The database D_φ. Our intention is to store (i) the values that a universally quantified variable can take, (ii) an auxiliary atom Sat(), which indicates that φ is satisfiable, (iii) all the satisfying assignments for each clause C_i (not C_i^+ or C_i^-) of φ, (iv) auxiliary atoms that would allow us to force the existentially quantified variable $y_{t(i)}$, for $i \in \{1, \ldots, n\}$, to take the same value as x_i, and (v) "consistency" atoms that would allow us to ensure that an assignment to the existentially quantified variables of φ is consistent among its clauses, i.e., a variable is assigned the same value in every clause that it appears. The formal definition of D_φ follows:

$$
\begin{aligned}
&\bigcup_{1 \le i \le n} \{\mathrm{Value}(x_i, 0), \mathrm{Value}(x_i, 1)\} \ \cup\ \{\mathrm{Sat}()\} \\
&\qquad \bigcup_{1 \le i \le k} \ \bigcup_{\substack{b_1, b_2, b_3 \in \{0,1\}, \\ b_1 \vee b_2 \vee b_3 = 1}} \left\{\mathrm{Clause}\left(c_i, \ell_i^1, b_1, \ell_i^2, b_2, \ell_i^3, b_3\right)\right\} \ \cup \\
&\quad \bigcup_{1 \le i \le n} \left(\bigcup_{b \in \{0,1\}} \left\{\mathrm{Force}\left(x_i, b, y_{t(i)}, b\right)\right\} \ \cup \bigcup_{\substack{b_1, b_2 \in \{0,1\} \\ b_1 \oplus b_2 = 1}} \left\{\mathrm{Force}\left(x_i, b_1, \neg y_{t(i)}, b_2\right)\right\} \right) \ \cup \\
&\qquad \bigcup_{1 \le i \le m} \left(\bigcup_{b \in \{0,1\}} \{\mathrm{Cons}(y_i, b, y_i, b), \mathrm{Cons}(\neg y_i, b, \neg y_i, b)\} \ \cup \right. \\
&\qquad\qquad \left. \bigcup_{\substack{b_1, b_2 \in \{0,1\} \\ b_1 \oplus b_2 = 1}} \{\mathrm{Cons}(y_i, b_1, \neg y_i, b_2), \mathrm{Cons}(\neg y_i, b_1, y_i, b_2)\} \right).
\end{aligned}
$$

The set Σ_φ of NCs. This set consists of three NCs. Note that, in what follows, we use x_i and y_j for the actual constants used in the database D_φ in order to represent the variables of φ. To avoid notational clutter, for variables we will use only the symbols z and w (possibly with subscripts and superscripts). The first NC of Σ_φ simply states that a universally quantified variable can take only one value:

$$\mathrm{Value}(z, 0), \mathrm{Value}(z, 1) \rightarrow \bot.$$

The second NC encodes the satisfiability of φ once an assignment to the universally quantified variables has been fixed (which is provided by a repair due to the NC above). Before defining this NC, let us introduce some auxiliary conjunctions of atoms, which will eventually give rise to the desired NC. The first one is

$$\mathrm{Config} \ = \ \bigwedge_{\alpha \in \mathit{Struct}} \alpha,$$

where $\mathit{Struct} = D_\varphi \setminus (\{\mathrm{Sat}()\} \cup \{\mathrm{Value}(x_i, 0), \mathrm{Value}(x_i, 1)\}_{1 \le i \le n})$. The second conjunction is defined as

$$\forall\mathrm{Assign} \ = \ \bigwedge_{1 \le i \le n} \mathrm{Value}(x_i, w_i),$$

which "reads" the assignment to the universally quantified variables. The third conjunction aims at "copying" the assignment for the universally quantified variables to the associated (according to C_i^+ and C_i^-) existentially quantified variables:

$$\text{Copy} \;=\; \bigwedge_{1 \le i \le n} \; \bigwedge_{\substack{1 \le j \le k, 1 \le r \le 3, \\ \mathsf{var}(\ell_j^r) = y_{t(i)}}} \text{Force}\left(x_i, w_i, z_j^r, w_j^r\right).$$

The fourth conjunction is defined as

$$\exists\text{Consistency} \;=\; \bigwedge_{\substack{1 \le j_1, j_2 \le k, \\ 1 \le r_1, r_2 \le 3, \\ \mathsf{var}(\ell_{j_1}^{r_1}) = \mathsf{var}(\ell_{j_2}^{r_2})}} \text{Cons}\left(z_{j_1}^{r_1}, w_{j_1}^{r_1}, z_{j_2}^{r_2}, w_{j_2}^{r_2}\right),$$

which states that an assignment to the existentially quantified variables is consistent among the clauses of φ. Finally, the fifth conjunction is defined as

$$\text{Satisfied} \;=\; \bigwedge_{1 \le i \le k} \text{Clause}\left(c_i, z_i^1, w_i^1, z_i^2, w_i^2, z_i^3, w_i^3\right),$$

which simply encodes the fact the φ is satisfiable. Having the above conjunctions in place, the desired NC is defined as

$$\text{Config}, \forall\text{Assign}, \text{Copy}, \exists\text{Consistency}, \text{Satisfied} \rightarrow \bot.$$

We also add to Σ_φ the NC

$$\text{Config}, \forall\text{Assign}, \text{Sat}() \rightarrow \bot.$$

This completes the construction of Σ_φ. We proceed to show that the above reduction is correct, i.e., φ is satisfiable iff $\mathsf{cert}_{\mathsf{IAR}}(\text{Sat}(), D_\varphi, \Sigma_\varphi) \neq \varnothing$.

($\Rightarrow$) Consider an arbitrary repair $D \in \mathsf{reps}(D_\varphi, \Sigma_\varphi)$. It suffices to show that $\text{Sat}() \in D$. We proceed by considering the following two cases on the shape of D:

1. There exists a universally quantified variable x_i such that none of the atoms $\text{Value}(x_i, b)$, for $b \in \{0, 1\}$, occurs in D, which means that D does not assign a value to x_i. In this case, Sat() necessarily belongs to D; otherwise, D is not a repair since adding Sat() would not violate any of the NCs.
2. Assume now that D assigns a value to every universally quantified variable of φ. By hypothesis, φ is satisfiable, which allows us (by definition of Σ_φ) to conclude that $\mathit{Struct} \setminus D \neq \varnothing$; otherwise, the body of the second NC is satisfied, which cannot be the case since D is a repair. But then Sat() necessarily belongs to D because otherwise D is not a repair since adding Sat() would not violate any of the NCs; in particular, the third NC would not be violated as $\mathit{Struct} \setminus D \neq \varnothing$.

($\Leftarrow$) Assume now that φ is not satisfiable. Thus, there exists an assignment μ to the universally quantified variables such that, for every (valid) assignment to the existentially quantified variables, φ is not satisfied. Let D be the subset of D_φ that keeps only one Value-atom for each universally quantified variable as dictated by μ, and all the other atoms of D_φ apart from Sat(). It should be clear that D satisfies Σ_φ. In particular, the second NC is satisfied since φ is not satisfiable, while the third NC is satisfied since Sat() is not in D. Moreover, D is maximal since by adding either Sat() or a Value-atom, one of the NCs will be violated. Therefore, $D \in \mathsf{reps}(D_\varphi, \Sigma_\varphi)$, which in turn implies that $\mathsf{cert}_{\mathsf{IAR}}(\text{Sat}(), D_\varphi, \Sigma_\varphi) = \varnothing$. ☐

Remark. Interestingly, the above proof applies even if we consider the AR semantics. Thus, we get an alternative proof for the fact that $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{NC})$ is Π_2^P-hard in ba-complexity. However, the proof given in Section 5 shows that the Π_2^P-hardness holds even for fp-complexity, but it exploits a more complex CQ.

Theorem 6.4. *$\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\bot)$ is Θ_2^P-hard in fp-complexity.*

The proof of the above result relies on a Θ_2^P-hard variant of 3SAT that involves counting of satisfiable formulas [32, 33]. For a set A of 3CNF formulas, we write $\#A$ for the cardinality of $\{\varphi \in A \mid \varphi \text{ is satisfiable}\}$. The problem follows:

PROBLEM :	Comp3SAT
INPUT :	Two sets A and B of 3CNF formulas.
QUESTION :	Is $\#A > \#B$?

The above problem remains Θ_2^P-hard even if we pose several simplifying assumptions on A and B. In particular, we can assume that $|A| = |B|$, all formulas in A and B are over the same set of variables and have the same number of clauses, and $A = \{\varphi_1, \ldots, \varphi_m\}$, $B = \{\psi_1, \ldots, \psi_m\}$ are such that φ_{i+1} (resp., ψ_{i+1}) is satisfiable implies φ_i (resp., ψ_i) is satisfiable, for $i \in \{1, \ldots, m-1\}$. It should be clear, due to the last assumption, that $\#A > \#B$ iff there exists $i \in \{1, \ldots, m\}$ such that φ_i is satisfiable and ψ_i is unsatisfiable.

We are now ready to proceed with the proof of Theorem 6.4.

Proof of Theorem 6.4. Given $A = \{\varphi_1, \ldots, \varphi_m\}$ and $B = \{\psi_1, \ldots, \psi_m\}$, we are going to construct a database $D_{A,B}$ and a Boolean CQ $q_{A,B}$ such that the following are equivalent:

1. There exists $i \in \{1, \ldots, m\}$ such that φ_i is satisfiable and ψ_i is unsatisfiable.
2. $\mathsf{cert}_{\mathsf{IAR}}(q_{A,B}, D_{A,B}, \Sigma) \neq \varnothing$ for some fixed set $\Sigma \in \mathsf{G}_\perp$.

Our goal is to devise $D_{A,B}$ and $q_{A,B}$ in a modular way, where the parts that are coming from A are independent from those that are coming from B. To this end, we are first going to construct a database D_A and a CQ $q_A(x)$ such that, for each $i \in \{1, \ldots, m\}$, φ_i is satisfiable iff $\langle f_i \rangle \in q_A(D_A)$; the constant f_i should be understood as the identifier for the formula φ_i. Moreover, we are going to construct a database D_B and a CQ $q_B(x)$ such that, for each $i \in \{1, \ldots, m\}$, ψ_i is unsatisfiable iff $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{IAR}}(q_B, D_B, \Sigma)$ for some fixed set $\Sigma \in \mathsf{G}_\perp$; here, f_i acts as the identifier for the formula ψ_i. Once we have the above databases and CQs in place, it will be easy to construct $D_{A,B}$ and $q_{A,B}$. In the sequel, we assume that all the formulas in A and B are over the variables $x_1, \ldots, x_n$ and have k clauses.

The database D_A and the CQ q_A. Given a 3CNF formula $\varphi_i = C_{i,1} \wedge \cdots \wedge C_{i,k}$ from A with $C_{i,j} = (\ell_{i,j}^1 \vee \ell_{i,j}^2 \vee \ell_{i,j}^3)$, the database D_A stores all the truth assignments that make a certain clause of φ_i true. To this end, we use an 8-ary predicate AClause. For example, given the clause $C_{i,j} = x_{i_1} \vee x_{i_2} \vee \neg x_{i_3}$,

$$\mathrm{AClause}(f_i, c_j, x_{i_1}, 1, x_{i_2}, 0, \neg x_{i_3}, 1)$$

encodes the clause itself (recall that f_i is the identifier of φ_i, while c_j is the identifier of $C_{i,j}$), and at the same time encodes the truth assignment that sets x_{i_1} to true, x_{i_2} to false, and x_{i_3} to false (i.e., $\neg x_{i_3}$ to true). Formally, let $D_{A,1}$ be the database

$$\bigcup_{1 \leq i \leq m} \; \bigcup_{1 \leq j \leq k} \; \bigcup_{\substack{b_1, b_2, b_3 \in \{0,1\}, \\ b_1 \vee b_2 \vee b_3 = 1}} \left\{ \mathrm{AClause}\left(f_i, c_j, \ell_{i,j}^1, b_1, \ell_{i,j}^2, b_2, \ell_{i,j}^3, b_3\right) \right\}.$$

To check whether a formula φ_i is satisfiable, we need to check whether each of its clauses is satisfiable; this is the purpose of the CQ q_A given below. To this end, we need a mechanism that allows us to ensure that an assignment for φ_i is consistent among its clauses, i.e., a variable is assigned the same value in every clause that it appears. This can be done via the following "consistency" atoms that form the database $D_{A,2}$:

$$\bigcup_{1 \leq i \leq n} \; \bigcup_{b \in \{0,1\}} \{\mathrm{Cons}(x_i, b, x_i, b), \mathrm{Cons}(\neg x_i, b, \neg x_i, b)\} \; \cup$$
$$\bigcup_{1 \leq i \leq n} \; \bigcup_{\substack{b_1, b_2 \in \{0,1\}, \\ b_1 \oplus b_2 = 1}} \{\mathrm{Cons}(x_i, b_1, \neg x_i, b_2), \mathrm{Cons}(\neg x_i, b_1, x_i, b_2)\}.$$

The database D_A is defined as the union $D_{A,1} \cup D_{A,2}$.

Let us now define the CQ $q_A(x)$. Its purpose is essentially to check whether there exists $\varphi_i \in A$ that is satisfiable, which can be done by checking that each clause of φ_i is satisfiable. This can be easily achieved via the CQ $q_A(x)$ defined below, where all the involved variables, apart from x, are existentially quantified; recall that $\mathsf{var}(\ell)$ is the variable of the literal ℓ:

$$\bigwedge_{1 \leq j \leq k} \mathrm{AClause}\left(x, c_j, y_j^1, z_j^1, y_j^2, z_j^2, y_j^3, z_j^3\right) \; \wedge \bigwedge_{1 \leq i \leq m} \; \bigwedge_{\substack{1 \leq j_1, j_2 \leq k, \\ 1 \leq r_1, r_2 \leq 3, \\ \mathsf{var}(\ell_{i,j_1}^{r_1}) = \mathsf{var}(\ell_{i,j_2}^{r_2})}} \mathrm{Cons}\left(y_{j_1}^{r_1}, z_{j_1}^{r_1}, y_{j_2}^{r_2}, z_{j_2}^{r_2}\right).$$

This completes the definition of q_A. By construction, we get that:

Lemma 6.7. *For each $i \in \{1,\dots,m\}$, φ_i is satisfiable iff $\langle f_i\rangle \in q_A(D_A)$.*

The database D_B and the CQ q_B. Given a 3CNF formula $\psi_i = C_{i,1} \wedge \cdots \wedge C_{i,k}$ from B with $C_{i,j} = (\ell^1_{i,j} \vee \ell^2_{i,j} \vee \ell^3_{i,j})$, the database D_B assigns to the variables of ψ_i both the values true and false, and it also stores all the clauses of ψ_i. The latter is achieved via a 5-ary predicate $\mathrm{BClause}^{s_1s_2s_3}$, where $s_1, s_2, s_3 \in \{\mathsf{p}, \mathsf{n}\}$. For example, the clause $C_{i,j} = x_{i_1} \vee x_{i_2} \vee \neg x_{i_3}$ is encoded via the atom

$$\mathrm{BClause}^{\mathsf{ppn}}(f_i, c_j, x_{i_1}, x_{i_2}, x_{i_3}),$$

with the superscript ppn indicating that the variable of the first (resp., second, third) literal appears positively (resp., positively, negatively) in $C_{i,j}$. Moreover, D_B stores some auxiliary atoms that would allow us to check (via a fixed set $\Sigma \in \mathsf{G}_\bot$) whether ψ_i is unsatisfiable. We proceed to formally define D_B.

For a literal ℓ, let $\mathsf{sign}(\ell) = \mathsf{p}$ (resp., $\mathsf{sign}(\ell) = \mathsf{n}$) if $\ell = x$ (resp., $\ell = \neg x$) for a variable x. For a clause $C_{i,j}$ of ψ_i, we write $s^r_{i,j}$ for $\mathsf{sign}(\ell^r_{i,j})$, where $r \in \{1,2,3\}$. Recall that $\mathsf{var}(\ell)$ is the variable of the literal ℓ. The database D_B is defined as

$$\begin{aligned}
&\bigcup_{1\le i\le m}\ \bigcup_{1\le j\le n} \{\mathrm{True}(f_i,x_j), \mathrm{False}(f_i,x_j)\} \ \cup \\
&\qquad\bigcup_{1\le i\le m}\ \bigcup_{1\le j\le k} \left\{\mathrm{BClause}^{s^1_{i,j},s^2_{i,j},s^3_{i,j}}\left(f_i, c_j, \mathsf{var}(\ell^1_{i,j}), \mathsf{var}(\ell^2_{i,j}), \mathsf{var}(\ell^3_{i,j})\right)\right\} \ \cup \\
&\qquad\qquad\bigcup_{1\le i\le m}\ \bigcup_{0\le j\le k-1} \{\mathrm{SuccCl}(f_i, c_j, c_{j+1})\} \ \cup \\
&\qquad\qquad\qquad\bigcup_{1\le i\le m} \{\mathrm{MinCl}(f_i, c_0), \mathrm{MaxCl}(f_i, c_k), \mathrm{Unsat}(f_i)\}.
\end{aligned}$$

The sequence of atoms $(\mathrm{SuccCl}(f_i, c_j, c_{j+1}))_{0\le j\le k-1}$ essentially tells us that in ψ_i the clause $C_{i,j}$ comes immediately after the clause $C_{i,j-1}$. The fact that the first atom of the sequence refers to the clause $C_{i,0}$, which does not exist, is a technicality that will become clear below. The remaining atoms give us access to the (virtually) first clause $C_{i,0}$ and the last clause $C_{i,k}$ of ψ_i, and also state that ψ_i is unsatisfiable.

The CQ $q_B(x)$, is defined as the atomic query $\mathrm{Unsat}(x)$, which simply asks whether there exists a formula in B that is unsatisfiable.

We claim that there exists a set $\Sigma \in \mathsf{G}_\bot$ such that, for each $i \in \{1,\dots,m\}$, ψ_i is unsatisfiable iff $\langle f_i\rangle \in \mathsf{cert}_{\mathsf{IAR}}(q_B, D_B, \Sigma)$. In particular, $\Sigma = \Sigma_{\mathsf{cons}} \cup \Sigma_{\mathsf{sat}}$ with Σ_{cons} being a set of NCs that perform a consistency check (i.e., a variable is either true of false, and a formula is either satisfiable or unsatisfiable), and Σ_{sat} being a set of guarded TGDs that evaluates each formula $\psi_i \in B$ and derives the atom $\mathrm{Sat}(f_i)$ if ψ_i is satisfiable. More precisely, Σ_{cons} consists of the NCs:

$$\begin{aligned}
\mathrm{True}(x,y), \mathrm{False}(x,y) &\ \rightarrow\ \bot \\
\mathrm{Sat}(x), \mathrm{Unsat}(x) &\ \rightarrow\ \bot.
\end{aligned}$$

The set Σ_{sat} consists of the following TGDs; a $\star$ symbol is a placeholder for p or n, while, as usual, $\_$ is a "don't care" variable that occurs only once:

$$\begin{aligned}
\mathrm{BClause}^{\mathsf{p}\star\star}(x,y,z,_,_), \mathrm{True}(x,z) &\ \rightarrow\ \mathrm{SatCl}(x,y) \\
\mathrm{BClause}^{\mathsf{n}\star\star}(x,y,z,_,_), \mathrm{False}(x,z) &\ \rightarrow\ \mathrm{SatCl}(x,y) \\
\mathrm{BClause}^{\star\mathsf{p}\star}(x,y,_,z,_), \mathrm{True}(x,z) &\ \rightarrow\ \mathrm{SatCl}(x,y) \\
\mathrm{BClause}^{\star\mathsf{n}\star}(x,y,_,z,_), \mathrm{False}(x,z) &\ \rightarrow\ \mathrm{SatCl}(x,y) \\
\mathrm{BClause}^{\star\star\mathsf{p}}(x,y,_,_,z), \mathrm{True}(x,z) &\ \rightarrow\ \mathrm{SatCl}(x,y) \\
\mathrm{BClause}^{\star\star\mathsf{n}}(x,y,_,_,z), \mathrm{False}(x,z) &\ \rightarrow\ \mathrm{SatCl}(x,y) \\
\mathrm{MinCl}(x,y) &\ \rightarrow\ \mathrm{SatChain}(x,y) \\
\mathrm{SatChain}(x,y), \mathrm{SuccCl}(x,y,z), \mathrm{SatCl}(x,z) &\ \rightarrow\ \mathrm{SatChain}(x,z) \\
\mathrm{MaxCl}(x,y), \mathrm{SatChain}(x,y) &\ \rightarrow\ \mathrm{Sat}(x).
\end{aligned}$$

This completes the construction of Σ. By construction, we get that:

Lemma 6.8. *For $i \in \{1, \dots, m\}$, ψ_i is unsatisfiable iff $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{IAR}}(q_B, D_B, \Sigma)$.*

The database $D_{A,B}$ and the CQ $q_{A,B}$. We can now easily construct the database $D_{A,B}$ and the Boolean CQ $q_{A,B}$ with the desired property:

$$D_{A,B} = D_A \cup D_B \quad \text{and} \quad q_{A,B} = \exists x (q_A(x) \wedge q_B(x)).$$

Indeed, we can show the following, where $\Sigma \in \mathsf{G}_\perp$ is the set devised above:

Lemma 6.9. *The following are equivalent:*

1. *There exists $i \in \{1, \dots, m\}$ such that φ_i is satisfiable and ψ_i is unsatisfiable.*
2. $\mathsf{cert}_{\mathsf{IAR}}(q_{A,B}, D_{A,B}, \Sigma) \neq \varnothing$.

Proof. We rely on the following key observation, which is easy to verify since D_A is always consistent with Σ:

$$\bigcap_{D \in \mathsf{reps}(D_{A,B}, \Sigma)} D \quad = \quad D_A \cup \left(\bigcap_{D \in \mathsf{reps}(D_B, \Sigma)} D \right). \tag{6}$$

We can now proceed with the proof of the claim.

(1) $\Rightarrow$ (2). By Lemma 6.7 and 6.8, we get that $\langle f_i \rangle \in q_A(D_A)$ and $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{IAR}}(q_B, D_B, \Sigma)$. Therefore, by (6), $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{IAR}}(q_A(x) \wedge q_B(x), D_{A,B}, \Sigma)$, which in turn implies that $\mathsf{cert}_{\mathsf{IAR}}(q_{A,B}, D_{A,B}, \Sigma) \neq \varnothing$.

(2) $\Rightarrow$ (1). By hypothesis, there exists $i \in \{1, \dots, m\}$ such that $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{IAR}}(q_A(x) \wedge q_B(x), D_{A,B}, \Sigma)$. By the equality (6), we can conclude that $\langle f_i \rangle \in q_A(D_A)$ and $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{IAR}}(q_B, D_B, \Sigma)$. Therefore, by Lemma 6.7 and 6.8, we get that φ_i is satisfiable and ψ_i is unsatisfiable, and the claim follows. ☐

With Lemma 6.9 in place, we can conclude that

$$\#A > \#B \iff \mathsf{cert}_{\mathsf{IAR}}(q_{A,B}, D_{A,B}, \Sigma) \neq \varnothing$$

for a fixed $\Sigma \in \mathsf{G}_\perp$, and thus, $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\perp)$ is Θ_2^P-hard in fp-complexity. ☐

Theorem 6.5. $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\perp)$ *is* co*NP-hard in* d*-complexity.*

The proof of the above result exploits the unsatisfiability problem of Boolean formulas in negation normal form. A Boolean formula is in *negation normal form* (NNF) if it uses only $\neg$, $\wedge$ and $\vee$, and $\neg$ is only applied to variables. The coNP-hard problem of interest follows:

PROBLEM :	NNF-UNSAT
INPUT :	A Boolean formula φ in NNF.
QUESTION :	Is φ unsatisfiable?

We can now proceed with the proof of Theoram 6.5.

Proof of Theorem 6.5. Given a formula φ in NNF over the variables $x_1, \dots, x_m$, we define the database D_φ as follows:

$$\begin{array}{ll} & \{\mathrm{And}(\psi, \psi_1, \psi_2) \mid \psi = \psi_1 \wedge \psi_2 \text{ is a subformula of } \varphi\} \\ \cup & \{\mathrm{Or}(\psi, \psi_1, \psi_2) \mid \psi = \psi_1 \vee \psi_2 \text{ is a subformula of } \varphi\} \\ \cup & \{\mathrm{Not}(\psi, \psi') \mid \psi = \neg\psi' \text{ is a subformula of } \varphi\} \\ \cup & \{\mathrm{True}(x_i), \mathrm{False}(x_i) \mid 1 \leq i \leq m\} \;\cup\; \{\mathrm{Unsat}(\varphi)\}, \end{array}$$

which essentially stores the formula φ, it assigns to each variable in φ both the value 1 and the value 0, and it states that φ is unsatisfiable.

	c-complexity	ba-complexity	fp-complexity	d-complexity
$\mathsf{G}_\perp$	2ExpTime	ExpTime	Θ_2^P	coNP
$\mathsf{L}_\perp$	PSpace	Π_2^P	Θ_2^P	coNP
$\mathsf{A}_\perp$	$\mathrm{P}^{\mathrm{NExpTime}}$	$\mathrm{P}^{\mathrm{NExpTime}}$	Θ_2^P	coNP
$\mathsf{S}_\perp$	ExpTime	Π_2^P	Θ_2^P	coNP

Table 4: Complexity of $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$. These are completeness results.

It is not difficult to show that φ is unsatisfiable iff $\mathsf{cert}_{\mathsf{IAR}}(\mathrm{Unsat}(\varphi), D_\varphi, \Sigma) \neq \varnothing$, where Σ consists of the guarded TGDs

$$
\begin{array}{rcl}
\mathrm{And}(x,y,z), \mathrm{True}(y), \mathrm{True}(z) & \rightarrow & \mathrm{True}(x)\\
\mathrm{Or}(x,y,z), \mathrm{True}(y) & \rightarrow & \mathrm{True}(x)\\
\mathrm{Or}(x,y,z), \mathrm{True}(z) & \rightarrow & \mathrm{True}(x)\\
\mathrm{Not}(x,y), \mathrm{False}(y) & \rightarrow & \mathrm{True}(x),
\end{array}
$$

which are responsible for evaluating the formula φ, and the NCs

$$
\begin{array}{rcl}
\mathrm{True}(x), \mathrm{False}(x) & \rightarrow & \perp\\
\mathrm{True}(x), \mathrm{Unsat}(x) & \rightarrow & \perp
\end{array}
$$

with the obvious meaning. We proceed to show that the above is a reduction.

($\Rightarrow$) Assume that $\mathsf{cert}_{\mathsf{IAR}}(\mathrm{Unsat}(\varphi), D_\varphi, \Sigma) = \varnothing$. Hence, there exists $D' \in \mathsf{reps}(D, \Sigma)$ such that $\mathrm{Unsat}(\varphi) \notin D'$. By definition of repairs, $\mathrm{True}(\varphi) \in D'$. Therefore, D' encodes a satisfying assignment for φ; simply set x_i to 1 (resp., 0) if $\mathrm{True}(x_i) \in D'$ (resp., $\mathrm{False}(x_i) \in D'$). Thus, φ is satisfiable, as needed.

($\Leftarrow$) Conversely, assume that φ is satisfiable. Thus, there is $D' \in \mathsf{reps}(D, \Sigma)$ such that $\mathrm{Unsat}(\varphi) \notin D'$; otherwise, the NC $\mathrm{True}(x), \mathrm{Unsat}(x) \rightarrow \perp$ would be violated. Hence, $\mathrm{Unsat}(\varphi) \notin \bigcap_{D' \in \mathsf{reps}(D,\Sigma)} D'$, which in turn implies that $\mathsf{cert}_{\mathsf{IAR}}(\mathrm{Unsat}(\varphi), D_\varphi, \Sigma) = \varnothing$, and the claim follows. $\square$

7 Intersection of closed repairs semantics

We now concentrate on $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$, where C is one of the classes of TGDs in question. The main result of this section follows:

Theorem 7.1. *The* t*-complexity of* $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$*, where* $\mathsf{t} \in \{\mathsf{c}, \mathsf{ba}, \mathsf{fp}, \mathsf{d}\}$ *and* $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$*, is as shown in Table 4.*

The rest of the section is devoted to establishing the above result. We first show, in Section 7.1, the upper bounds, and then, in Section 7.2, the lower bounds.

7.1 Upper bounds

We can partition the cells of Table 4 into five groups in such a way that the claimed upper bounds can be established in a uniform way:

1. The c-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{S}\}$.
2. The c-complexity for $\mathsf{A}_\perp$.
3. The c-complexity for $\mathsf{L}_\perp$.
4. The ba-complexity and the d-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$.
5. The fp-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$.

We proceed to give more details for each of the above groups.

Input: database D, set Σ of TGDs and NCs, CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$
Output: accept if $\bar{c} \in \mathsf{cert}_{\mathsf{ICR}}(q, D, \Sigma)$; otherwise, reject

```
D⋆ := ∅
foreach α ∈ B(D, Σ) do
    if cert_AR(α, D, Σ) ≠ ∅ then
        D⋆ := D⋆ ∪ {α}
if c̄ ∈ cert(q, D⋆, τ(Σ)) then
    return accept
else
    return reject
```

Algorithm 5: AlgorithmICR1

7.1.1 The c-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{S}\}$

The upper bounds are obtained via the procedure AlgorithmICR1, depicted in Algorithm 5, which constructs the intersection of closed repairs $D^\star$, and accepts if the given tuple $\bar{c}$ belongs to $\mathsf{cert}(q, D^\star, \Sigma)$; otherwise, it rejects. The intersection of closed repairs $D^\star$ is constructed by keeping from $\mathsf{B}(D, \Sigma)$, i.e., the set of all ground atoms that can be formed using constants from $\mathsf{dom}(D)$ and predicates occurring in Σ, only the atoms α that belong to $\mathsf{cl}(D', \tau(\Sigma))$ for each $D' \in \mathsf{reps}(D, \Sigma)$, or, equivalently, for which $\mathsf{cert}_{\mathsf{AR}}(\alpha, D, \Sigma) \neq \varnothing$. The fact that $\mathsf{B}(D, \Sigma)$ consists of exponentially many atoms, allows us to conclude that, for a class C of TGDs, if $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{C}_\perp)$ is in $\mathcal{C} \in \{\textsc{ExpTime}, 2\textsc{ExpTime}\}$ in c-complexity (and thus, $\mathsf{QAns}(\mathsf{C})$ is in $\mathcal{C}$ in c-complexity), and the complexity bound inherited from the algorithm underlying the membership of $\mathsf{QAns}(\mathsf{C})$ in $\mathcal{C}$ in c-complexity depends polynomially on the input database, then AlgorithmICR1 shows that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$ is also in $\mathcal{C}$ in c-complexity. By Theorem 5.1, $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{G}_\perp)$ is in $2\textsc{ExpTime}$, and $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{S}_\perp)$ is in $\textsc{ExpTime}$ in c-complexity. Moreover, we know from [10] that the complexity bound inherited from the algorithm underlying the fact that $\mathsf{QAns}(\mathsf{G})$ is in $2\textsc{ExpTime}$ in c-complexity depends polynomially on the input database. The same holds for the class S [13], and the desired upper bounds follow.

7.1.2 The c-complexity for $\mathsf{A}_\perp$

For showing that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{A}_\perp)$ is in $\mathrm{P}^{\textsc{NExpTime}}$ in c-complexity, we rely again on AlgorithmICR1, but we need a more refined complexity analysis than the one given above for the classes $\mathsf{G}_\perp$ and $\mathsf{S}_\perp$. Since (i) $\mathsf{B}(D, \Sigma)$ consists of exponentially many atoms, (ii) $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{A}_\perp)$ is in $\mathrm{P}^{\textsc{NExpTime}}$ in c-complexity by Theorem 5.1, (iii) $\mathsf{QAns}(\mathsf{A})$ is in $\textsc{NExpTime}$ in c-complexity by Proposition 3.3, and (iv) the complexity bound inherited from the algorithm underlying the fact that $\mathsf{QAns}(\mathsf{A})$ is in $\textsc{NExpTime}$ in c-complexity depends polynomially on the input database, AlgorithmICR1 allows us to conclude that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{A}_\perp)$ is in $\textsc{NExpTime}^{\textsc{NExpTime}}$ in c-complexity. We know that $\mathrm{P}^{\textsc{NExpTime}}$ is included in $\textsc{NExpTime}^{\textsc{NExpTime}}$, but we also know from [46] that the two complexity classes coincide if, whenever the $\textsc{NExpTime}$ oracle is called, its input is of polynomial size, which gives rise to the complexity class $\textsc{NExpTime}^{\textsc{NExpTime}[poly]}$ (we borrow the notation from [46]). We now observe that during the execution of AlgorithmICR1, the input to the $\textsc{NExpTime}$ oracle, which is responsible for checking whether $\mathsf{cert}_{\mathsf{AR}}(\alpha, D, \Sigma) \neq \varnothing$ for an atom $\alpha \in \mathsf{B}(D, \Sigma)$, is always of polynomial size w.r.t. D and Σ. This implies that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{A}_\perp)$ is in $\textsc{NExpTime}^{\textsc{NExpTime}[poly]}$ (and thus, in $\mathrm{P}^{\textsc{NExpTime}}$) in c-complexity, and the claim follows.

7.1.3 The c-complexity for $\mathsf{L}_\perp$

For showing that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{L}_\perp)$ is in $\textsc{PSpace}$, we need to rely on a refined version of the procedure AlgorithmICR1. Indeed, AlgorithmICR1 only shows that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{L}_\perp)$ is in $\textsc{ExpTime}$ in c-complexity, despite the fact that $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{L}_\perp)$ is in $\textsc{PSpace}$ in c-complexity, since $\mathsf{B}(D, \Sigma)$ consists of exponentially many atoms. The key ingredient underlying this refined procedure is the following property of linear TGDs, which is implicit in [11], that essentially states that for computing the certain answers of a CQ, we only need linearly many database atoms.

Lemma 7.1. *Consider a database D, a set $\Sigma \in \mathsf{L}$, a CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$. The following are equivalent:*

Input: database D, set $\Sigma \in \mathsf{L}_\perp$, CQ $q(\bar{x})$, tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$
Output: accept if $\bar{c} \in \mathsf{cert}_{\mathsf{ICR}}(q, D, \Sigma)$; otherwise, reject

```
guess a database D* ⊆ B(D, Σ) with |D*| ≤ |q|
foreach α ∈ D* do
    if cert_AR(α, D, Σ) = ∅ then
        return reject
if c̄ ∈ cert(q, D*, τ(Σ)) then
    return accept
else
    return reject
```

Algorithm 6: AlgorithmICR2

Input: database D, set Σ of TGDs and NCs, CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$
Output: accept if $\bar{c} \notin \mathsf{cert}_{\mathsf{ICR}}(q, D, \Sigma)$; otherwise, reject

```
guess a database D* ⊆ cl(D, τ(Σ))
foreach α ∈ cl(D, τ(Σ)) \ D* do
    guess a database D_α ⊆ D
    if α ∈ cl(D_α, τ(Σ)) then
        return reject
    else
        foreach β ∈ D \ D_α do
            if there is no σ ∈ ν(Σ) s.t. cert(q_σ, D_α ∪ {β}, τ(Σ)) ≠ ∅ then
                return reject
if c̄ ∈ cert(q, D*, τ(Σ)) then
    return reject
else
    return accept
```

Algorithm 7: AlgorithmICR3

1. $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$.

2. There exists $D' \subseteq D$ *with* $|D'| \leq |q|$ *such that* $\bar{c} \in \mathsf{cert}(q, D', \Sigma)$.

By Lemma 7.1, we obtain the decision procedure AlgorithmICR2, depicted in Algorithm 6, for $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{L}_\perp)$ by adapting AlgorithmICR1 as follows: instead of deterministically computing the intersection of closed repairs, we simply guess $|q|$ atoms of $\mathsf{B}(D, \Sigma)$, and then verify that are indeed members of the intersection of closed repairs. Since, by Theorem 5.1, $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{L}_\perp)$ is in PSPACE in c-complexity, AlgorithmICR2 uses polynomial space, and the claim follows.

7.1.4 The ba-complexity and the d-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$

The upper bounds are obtained via the simple procedure AlgorithmICR3, depicted in Algorithm 6, which is similar in spirit to the procedure AlgorithmIAR1, and checks whether there exists a superset of the intersection of closed repairs that does not entail the given tuple $\bar{c}$ of constants. More precisely, the algorithm guesses a subset $D^\star$ of $\mathsf{cl}(D, \tau(\Sigma))$, that is, the set of ground atoms that can be entailed by D and $\tau(\Sigma)$, and then checks that for every atom $\alpha \in \mathsf{cl}(D, \tau(\Sigma)) \setminus D^\star$, there exists $D_\alpha \in \mathsf{reps}(D, \Sigma)$ such that $\alpha \notin \mathsf{cl}(D_\alpha, \tau(\Sigma))$, and thus, α is not in the intersection of closed repairs. This implies that $D^\star$ is a superset of the intersection of closed repairs. Finally, the algorithm rejects if $\bar{c} \in \mathsf{cert}(q, D^\star, \tau(\Sigma))$; otherwise, it accepts. This is correct due to the following lemma that can be shown as Lemma 6.1:

Lemma 7.2. *Consider a database* D*, a set* Σ *of TGDs and NCs, a CQ* $q(\bar{x})$*, and a tuple* $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$*. The following are equivalent:*

1. $\bar{c} \notin \mathsf{cert}_{\mathsf{ICR}}(q, D, \Sigma)$.

2. There is $D^\star \supseteq \bigcap_{D' \in \mathsf{reps}(D,\Sigma)} \mathsf{cl}(D', \tau(\Sigma))$ *such that* $\bar{c} \notin \mathsf{cert}(q, D^\star, \tau(\Sigma))$.

Input: database D, set Σ of TGDs and NCs, CQ $q(\bar{x})$, tuple $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$
Output: accept if $\bar{c} \in \mathsf{cert}_{\mathsf{ICR}}(q, D, \Sigma)$; otherwise, reject

$D^\star := \mathsf{B}(D, \Sigma)$
foreach $\alpha \in \mathsf{B}(D, \Sigma)$ **do**
  if there exists $D_\alpha \in \mathsf{reps}(D, \Sigma)$ such that $\alpha \notin \mathsf{cl}(D_\alpha, \tau(\Sigma))$ **then**
    $D^\star := D^\star \setminus \{\alpha\}$

if $\bar{c} \in \mathsf{cert}(q, D^\star, \tau(\Sigma))$ **then**
  return accept
else
  return reject

Algorithm 8: AlgorithmICR4

Since we focus on predicates of bounded arity, the non-deterministic procedure AlgorithmICR3 runs in polynomial time, assuming access to an oracle that is powerful enough for solving $\mathsf{QAns}(\mathsf{C})$, where C is the class from which the input set of TGDs is coming from. Notice that for computing the database $\mathsf{cl}(D, \tau(\Sigma))$ (or $\mathsf{cl}(D_\alpha, \tau(\Sigma))$) we simply need to enumerate the polynomially many ground atoms that can be formed using constants from $\mathsf{dom}(D)$ and predicates occurring in Σ, and for each such atom γ check whether $\mathsf{cert}(\gamma, D, \tau(\Sigma)) \neq \emptyset$. Therefore:

Lemma 7.3. *For a class C of TGDs, $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$ is in $\mathrm{coNP}^{\mathcal{C}}$ in t-complexity, where $\mathsf{t} \in \{\mathsf{ba}, \mathsf{d}\}$, assuming that $\mathsf{QAns}(\mathsf{C})$ is in $\mathcal{C}$ in t-complexity.*

Since, by Lemma 7.2, AlgorithmICR3 is correct, the desired upper bounds for Group 2 are obtained from Propositions 3.1, 3.2, 3.3 and 3.4, Lemma 7.3, and the usual complexity facts that have been discussed in the previous sections.

Remark. Let us observe that we could also employ the procedure AlgorithmICR1 for obtaining the ExpTime and $\mathrm{P}^{\mathrm{NExpTime}}$ upper bounds for $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{G}_\perp)$ and $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{A}_\perp)$, respectively, since, by Theorem 5.1, $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{G}_\perp)$ is in ExpTime, and $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{A}_\perp)$ is in $\mathrm{P}^{\mathrm{NExpTime}}$ in ba-complexity.

7.1.5 The fp-complexity for $\mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$

We finally discuss how the $\Theta_2^P = \mathrm{P}^{\mathrm{NP}[O(\log n)]}$ upper bound for $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$, where $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$, can be established. Actually, this is done by exploiting the procedure AlgorithmICR4, depicted in Algorithm 8, which is an adaptation of AlgorithmIAR2, that constructs the intersection of closed repairs $D^\star$, and accepts if the given tuple $\bar{c}$ belongs to $\mathsf{cert}(q, D^\star, \tau(\Sigma))$; otherwise, it rejects. The intersection of closed repairs $D^\star$ is constructed by starting from $\mathsf{B}(D, \Sigma)$, i.e., the set of all ground atoms that can be formed using constants from $\mathsf{dom}(D)$ and predicates occurring in Σ, which are polynomially many since the arity is bounded, and removing all the atoms α for which there exists at least one repair $D_\alpha \in \mathsf{reps}(D, \Sigma)$ such that $\alpha \notin \mathsf{cl}(D_\alpha, \tau(\Sigma))$. More precisely, $D^\star$ is constructed via polynomially many parallel calls to an NP-oracle. In fact, for each atom $\alpha \in \mathsf{B}(D, \Sigma)$, we call in parallel an NP-oracle that does the following:

1. Guess a database $D_\alpha \subseteq D$.

2. If $\alpha \in \mathsf{cl}(D_\alpha, \tau(\Sigma))$, then reject.

3. For each atom $\beta \in D \setminus D_\alpha$, if there is no $\sigma \in \nu(\Sigma)$ such that $\mathsf{cert}(q_\sigma, D_\alpha \cup \{\beta\}, \tau(\Sigma)) \neq \emptyset$, then return reject; otherwise; return accept.

The checks $\alpha \in \mathsf{cl}(D_\alpha, \tau(\Sigma))$ and $\mathsf{cert}(q_\sigma, D_\alpha \cup \{\beta\}, \tau(\Sigma)) \neq \emptyset$ are feasible in polynomial time since $\tau(\Sigma)$ and q_σ are fixed, while for each $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$, $\mathsf{QAns}(\mathsf{C})$ is in PTime in d-complexity. Therefore, the above oracle is indeed an NP-oracle. It is clear that, for an atom $\alpha \in \mathsf{B}(D, \Sigma)$, if the above oracle returns accept, then α does not belong to the intersection of closed repairs. Consequently, the intersection of closed repairs $D^\star$ is constructed by simply removing from D all the atoms α for which the oracle returns accept. Since $D^\star$ can be constructed in polynomial time via parallel NP-oracle calls, we can conclude that it can also be constructed in polynomial time via logarithimically many NP-oracle calls; see, e.g., [42]. Once we have $D^\star$ in place, we need one more call to an NP-oracle for checking whether $\bar{c} \in \mathsf{cert}(q, D^\star, \tau(\Sigma))$; the latter is indeed in NP since, for each $\mathsf{C} \in \{\mathsf{G}, \mathsf{L}, \mathsf{A}, \mathsf{S}\}$, $\mathsf{QAns}(\mathsf{C})$ is in NP in fp-complexity. The claim follows.

7.2 Lower bounds

We now concentrate on the complexity lower bounds claimed in Table 4. The $\mathcal{C}$-hardness results, where $\mathcal{C} \in \{\text{PSPACE}, \text{EXPTIME}, 2\text{EXPTIME}\}$, are coming for free since $\mathsf{QAns}(\mathsf{C})$ is $\mathcal{C}$-hard. Therefore, to complete the picture, it suffices to establish the following hardness results:

1. $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{A}_\perp)$ is $\mathrm{P}^{\mathrm{NEXPTIME}}$-hard in ba-complexity.
2. $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{NC})$ is Π_2^P-hard in ba-complexity.
3. $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, is Θ_2^P-hard in fp-complexity.
4. $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, is coNP-hard in d-complexity.

The rest of the section is devoted to establishing the above lower bounds. But let us first establish an auxiliary lemma, which will be useful for our later analysis. It states that for ground atomic CQs the AR and the ICR semantics coincide:

Lemma 7.4. *Consider a database D, a set Σ of TGDs and NCs, and a ground atom α. Then,* $\mathsf{cert}_{\mathsf{AR}}(\alpha, D, \Sigma) \neq \varnothing$ *iff* $\mathsf{cert}_{\mathsf{ICR}}(\alpha, D, \Sigma) \neq \varnothing$.

Proof. ($\Rightarrow$) By hypothesis, for every $D' \in \mathsf{reps}(D, \Sigma)$, $\mathsf{cert}(\alpha, D', \tau(\Sigma)) \neq \varnothing$. This implies that, for every $D' \in \mathsf{reps}(D, \Sigma)$, $\alpha \in \mathsf{cl}(D', \tau(\Sigma))$. Therefore, $\alpha \in \bigcap_{D' \in \mathsf{reps}(D,\Sigma)} \mathsf{cl}(D', \tau(\Sigma))$, which means that $\mathsf{cert}_{\mathsf{ICR}}(\alpha, D, \Sigma) \neq \varnothing$, as needed.

($\Leftarrow$) Conversely, assume that $\mathsf{cert}_{\mathsf{AR}}(\alpha, D, \Sigma) = \varnothing$. Hence, there exists $D_\alpha \in \mathsf{reps}(D, \Sigma)$ such that $\mathsf{cert}(\alpha, D_\alpha, \tau(\Sigma)) = \varnothing$, and thus, $\alpha \notin \mathsf{cl}(D_\alpha, \tau(\Sigma))$. Assume now that $\mathsf{cert}(\alpha, D^\star, \tau(\Sigma)) \neq \varnothing$ with $D^\star = \bigcap_{D' \in \mathsf{reps}(D,\Sigma)} \mathsf{cl}(D', \tau(\Sigma))$. This implies that there exists $D'' \subseteq \mathsf{cl}(D_\alpha, \tau(\Sigma))$ such that $\mathsf{cert}(\alpha, D'', \tau(\Sigma)) \neq \varnothing$. But this allows us to conclude that $\alpha \in \mathsf{cl}(D_\alpha, \tau(\Sigma))$, which is a contradiction. Therefore, $\mathsf{cert}(\alpha, D^\star, \tau(\Sigma)) = \varnothing$, which means that $\mathsf{cert}_{\mathsf{ICR}}(\alpha, D, \Sigma) = \varnothing$. ☐

We proceed with the proofs of the claimed lower bounds.

Theorem 7.2. $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{A}_\perp)$ *is* $\mathrm{P}^{\mathrm{NEXPTIME}}$*-hard in* ba*-complexity.*

Proof. By Lemma 7.4, we can apply the proof for the fact that $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{A}_\perp)$ is $\mathrm{P}^{\mathrm{NEXPTIME}}$-hard in ba-complexity. Recall that for showing the latter we reduce from the extended exponential tiling problem. In fact, given an extended tiling system $\mathcal{E}$, we construct a database $D_{\mathcal{E}}$, and a set $\Sigma_{\mathcal{E}} \in \mathsf{A}_\perp$ that mentions only predicates of bounded arity, such that $\mathcal{E}$ is valid iff $\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), D, \Sigma) \neq \varnothing$, where Yes is a 0-ary predicate indicating that $\mathcal{E}$ is indeed valid. By Lemma 7.4, we can conclude that $\mathsf{cert}_{\mathsf{AR}}(\mathrm{Yes}(), D, \Sigma) \neq \varnothing$ iff $\mathsf{cert}_{\mathsf{ICR}}(\mathrm{Yes}(), D, \Sigma) \neq \varnothing$, which shows that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{A}_\perp)$ is $\mathrm{P}^{\mathrm{NEXPTIME}}$-hard in ba-complexity. ☐

Theorem 7.3. $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{NC})$ *is* Π_2^P*-hard in* ba*-complexity.*

Proof. The proof of Theorem 6.3, showing that $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{NC})$ is Π_2^P-hard in ba-complexity, applies also to the ICR semantics. The reason is that $\mathsf{cl}(D, \Sigma) = D$ for every database D and Σ in NC, since Σ does not include any TGD. Hence, the IAR and ICR semantics coincide for the class NC. ☐

Theorem 7.4. $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$*, where* $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$*, is* Θ_2^P*-hard in* fp*-complexity.*

Proof. We reduce from $\mathsf{Comp3SAT}$ [32, 33]. Recall that given two sets A and B of 3CNF formulas, this problem asks whether $\#A > \#B$, i.e., whether A contains more satisfiable formulas than B. Recall also that this problem remains Θ_2^P-hard even if $|A| = |B|$, all formulas in A and B are over the same set of variables and have the same number of clauses, and $A = \{\varphi_1, \ldots, \varphi_m\}$, $B = \{\psi_1, \ldots, \psi_m\}$ are such that φ_{i+1} (resp., ψ_{i+1}) is satisfiable implies φ_i (resp., ψ_i) is satisfiable, for each $i \in \{1, \ldots, m-1\}$. Clearly, $\#A > \#B$ iff there exists $i \in \{1, \ldots, m\}$ such that φ_i is satisfiable and ψ_i is unsatisfiable.

Given $A = \{\varphi_1, \ldots, \varphi_m\}$ and $B = \{\psi_1, \ldots, \psi_m\}$, our goal is to construct a database $D_{A,B}$ and a Boolean CQ $q_{A,B}$ such that the following are equivalent:

1. There exists $i \in \{1, \ldots, m\}$ such that φ_i is satisfiable and ψ_i is unsatisfiable.
2. $\mathsf{cert}_{\mathsf{ICR}}(q_{A,B}, D_{A,B}, \Sigma) \neq \varnothing$ for some fixed $\Sigma \in \mathsf{C}_\perp$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$.

In fact, the construction is along the lines of the one given in Section 6 for showing that $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\perp)$ is Θ_2^P-hard in fp-complexity. We first construct a database D_A and a CQ $q_A(x)$ such that, for each $i \in \{1,\ldots,m\}$, φ_i is satisfiable iff $\langle f_i \rangle \in q_A(D_A)$; the constant f_i should be understood as the identifier for the formula φ_i. Moreover, we construct a database D_B and a CQ $q_B(x)$ such that, for $i \in \{1,\ldots,m\}$, ψ_i is unsatisfiable iff $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{ICR}}(q_B, D_B, \Sigma) \neq \varnothing$ for some fixed set $\Sigma \in \mathsf{C}_\perp$, for $\mathsf{C} \in \{\mathsf{L},\mathsf{A},\mathsf{S}\}$; here, f_i is the identifier of ψ_i. Once we have the above in place, we can easily construct $D_{A,B}$ and $q_{A,B}$. We assume that all the formulas in A and B are over the variables $x_1,\ldots,x_n$ and have k clauses.

The database D_A and the CQ q_A. Actually, for D_A and q_A we can use exactly the same construction as in the proof of the fact that $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\perp)$ is Θ_2^P-hard in fp-complexity given in Section 6.

The database D_B and the CQ q_B. Let us now explain the construction of D_B and q_B, which is significantly different (at least the construction of D_B) than the one given in the previous section. Given a 3CNF formula $\psi_i = C_{i,1} \wedge \cdots \wedge C_{i,k}$ from B with $C_{i,j} = (\ell_{i,j}^1 \vee \ell_{i,j}^2 \vee \ell_{i,j}^3)$, the database D_B essentially stores all the possible truth assignments for each clause of ψ_i. To this end, we use a 5-ary predicate $\mathrm{BClause}_{b_1b_2b_3}^{s_1s_2s_3}$, where $s_1,s_2,s_3 \in \{\mathsf{p},\mathsf{n}\}$ and $b_1,b_2,b_3 \in \{0,1\}$. For example, given the clause $C_{i,j} = x_{i_1} \vee x_{i_2} \vee \neg x_{i_3}$, the atom

$$\mathrm{BClause}_{101}^{\mathsf{ppn}}(f_i, c_j, x_{i_1}, x_{i_2}, x_{i_3})$$

encodes the clause itself, with the superscript ppn indicating that the variable of the first (resp., second, third) literal appears positively (resp., positively, negatively) in $C_{i,j}$, and at the same time encodes the truth assignment that sets x_{i_1} to true, x_{i_2} to false, and x_{i_3} to true. We proceed to formally define D_B.

Recall that for a literal ℓ, $\mathsf{sign}(\ell) = \mathsf{p}$ (resp., $\mathsf{sign}(\ell) = \mathsf{n}$) if $\ell = x$ (resp., $\ell = \neg x$). For a clause $C_{i,j}$ of φ, we write $s_{i,j}^r$ for $\mathsf{sign}(\ell_{i,j}^r)$, where $r \in \{1,2,3\}$. Recall that $\mathsf{var}(\ell)$ is the variable of the literal ℓ. The database D_B is defined as

$$\bigcup_{1 \leq i \leq m} \; \bigcup_{1 \leq j \leq k} \; \bigcup_{b_1,b_2,b_3 \in \{0,1\}} \left\{ \mathrm{BClause}_{b_1b_2b_3}^{s_{i,j}^1 s_{i,j}^2 s_{i,j}^3} \left(f_i, c_j, \mathsf{var}(\ell_{i,j}^1), \mathsf{var}(\ell_{i,j}^2), \mathsf{var}(\ell_{i,j}^3)\right) \right\}.$$

Regarding the CQ $q_B(x)$, is defined as the atomic query $\mathrm{Unsat}(x)$, which simply asks whether there exists a formula in B that is unsatisfiable.

We claim that there exists a set $\Sigma \in \mathsf{C}_\perp$, for $\mathsf{C} \in \{\mathsf{L},\mathsf{A},\mathsf{S}\}$, such that, for each $i \in \{1,\ldots,m\}$, ψ_i is unsatisfiable iff $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{ICR}}(q_B, D_B, \Sigma)$. In particular, $\Sigma = \Sigma_{\mathsf{cons}} \cup \Sigma_{\mathsf{unsat}}$ with Σ_{cons} being a set of NCs that performs a consistency check on the truth assignment for ψ_i, i.e., each variable of ψ_i is assigned exactly one value, and Σ_{unsat} being a set of TGDs that entails the ground atom $\mathrm{Unsat}(f_i)$ whenever a clause of ψ_i evaluates to false. More precisely, Σ_{cons} consists of the following NCs; a $\star$ symbol in the superscript is a placeholder for p or n, a $\star$ in the subscript is a placeholder for 0 or 1, and, as usual, _ is a "don't care" variable:

$$\begin{array}{rcl}
\mathrm{BClause}_{1\star\star}^{\star\star\star}(x,_,y,_,_), \mathrm{BClause}_{0\star\star}^{\star\star\star}(x,_,y,_,_) & \rightarrow & \perp \\
\mathrm{BClause}_{1\star\star}^{\star\star\star}(x,_,y,_,_), \mathrm{BClause}_{\star0\star}^{\star\star\star}(x,_,_,y,_) & \rightarrow & \perp \\
\mathrm{BClause}_{1\star\star}^{\star\star\star}(x,_,y,_,_), \mathrm{BClause}_{\star\star0}^{\star\star\star}(x,_,_,_,y) & \rightarrow & \perp \\
\mathrm{BClause}_{\star1\star}^{\star\star\star}(x,_,_,y,_), \mathrm{BClause}_{0\star\star}^{\star\star\star}(x,_,y,_,_) & \rightarrow & \perp \\
\mathrm{BClause}_{\star1\star}^{\star\star\star}(x,_,_,y,_), \mathrm{BClause}_{\star0\star}^{\star\star\star}(x,_,_,y,_) & \rightarrow & \perp \\
\mathrm{BClause}_{\star1\star}^{\star\star\star}(x,_,_,y,_), \mathrm{BClause}_{\star\star0}^{\star\star\star}(x,_,_,_,y) & \rightarrow & \perp \\
\mathrm{BClause}_{\star\star1}^{\star\star\star}(x,_,_,_,y), \mathrm{BClause}_{0\star\star}^{\star\star\star}(x,_,y,_,_) & \rightarrow & \perp \\
\mathrm{BClause}_{\star\star1}^{\star\star\star}(x,_,_,_,y), \mathrm{BClause}_{\star0\star}^{\star\star\star}(x,_,_,y,_) & \rightarrow & \perp \\
\mathrm{BClause}_{\star\star1}^{\star\star\star}(x,_,_,_,y), \mathrm{BClause}_{\star\star0}^{\star\star\star}(x,_,_,_,y) & \rightarrow & \perp.
\end{array}$$

Moreover, the set of TGDs Σ_{unsat} consists of:

$$\begin{array}{rcl}
\mathrm{BClause}^{\mathsf{nnn}}_{111}(x, _, _, _, _) & \rightarrow & \mathrm{Unsat}(x) \\
\mathrm{BClause}^{\mathsf{nnp}}_{110}(x, _, _, _, _) & \rightarrow & \mathrm{Unsat}(x) \\
\mathrm{BClause}^{\mathsf{npn}}_{101}(x, _, _, _, _) & \rightarrow & \mathrm{Unsat}(x) \\
\mathrm{BClause}^{\mathsf{npp}}_{100}(x, _, _, _, _) & \rightarrow & \mathrm{Unsat}(x) \\
\mathrm{BClause}^{\mathsf{pnn}}_{011}(x, _, _, _, _) & \rightarrow & \mathrm{Unsat}(x) \\
\mathrm{BClause}^{\mathsf{pnp}}_{010}(x, _, _, _, _) & \rightarrow & \mathrm{Unsat}(x) \\
\mathrm{BClause}^{\mathsf{ppn}}_{001}(x, _, _, _, _) & \rightarrow & \mathrm{Unsat}(x) \\
\mathrm{BClause}^{\mathsf{ppp}}_{000}(x, _, _, _, _) & \rightarrow & \mathrm{Unsat}(x).
\end{array}$$

Observe that Σ_{unsat} falls in C, for each $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$. We proceed to show that:

Lemma 7.5. *For $i \in \{1, \ldots, m\}$, ψ_i is unsatisfiable iff $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{ICR}}(q_B, D_B, \Sigma)$.*

Proof. ($\Rightarrow$) Consider an arbitrary repair $D \in \mathsf{reps}(D_B, \Sigma)$. It is easy to verify that, due to Σ_{cons}, D encodes a consistent assignment μ_D of truth values to the variables of ψ_i, i.e., for each clause $C_{i,j}$ of ψ_i, D contains exactly one atom of the form

$$\mathrm{BClause}^{s_i^1 s_i^2 s_i^3}_{b_1 b_2 b_3}\left(f_i, c_j, \mathsf{var}(\ell^1_{i,j}), \mathsf{var}(\ell^2_{i,j}), \mathsf{var}(\ell^3_{i,j})\right).$$

Since, by hypothesis, ψ_i is unsatisfiable, there exists a clause $C_{i,j}$ of ψ_i such that, according to μ_D, evaluates to false. This implies that a TGD of Σ_{unsat} will be triggered, and thus, $\mathrm{Unsat}(f_i) \in \mathsf{cl}(D, \tau(\Sigma))$. Hence,

$$\mathrm{Unsat}(f_i) \;\in \bigcap_{D' \in \mathsf{reps}(D_B, \Sigma)} \mathsf{cl}(D', \tau(\Sigma)),$$

which in turn implies that $\langle f_i \rangle \in \mathsf{cert}_{\mathsf{ICR}}(q_B, D_B, \Sigma)$.

($\Leftarrow$) Conversely, assume that ψ_i is satisfiable, and let μ be a satisfying assignment that witnesses this fact. It is easy to verify that there is a repair $D_\mu \in \mathsf{reps}(D_B, \Sigma)$ that encodes μ. Since μ is a satisfying assignment, all the clauses of ψ_i evaluate to true. This means that none of the TGDs of Σ_{unsat} will be triggered, and thus, $\langle f_i \rangle \notin \mathsf{cert}_{\mathsf{AR}}(q_B, D_B, \Sigma)$. By Lemma 7.4, we can conclude that $\langle f_i \rangle \notin \mathsf{cert}_{\mathsf{ICR}}(q_B, D_B, \Sigma) = \varnothing$, and the claim follows. ☐

The database $D_{A,B}$ and the CQ $q_{A,B}$. We can now easily construct the database $D_{A,B}$ and the Boolean CQ $q_{A,B}$ with the desired property:

$$D_{A,B} = D_A \cup D_B \quad \text{ and } \quad q_{A,B} = \exists x (q_A(x) \wedge q_B(x)).$$

With Σ being the set devised above, it is not difficult to verify that

$$\bigcap_{D \in \mathsf{reps}(D_{A,B}, \Sigma)} \mathsf{cl}(D, \Sigma) \;=\; D_A \cup \left(\bigcap_{D \in \mathsf{reps}(D_B, \Sigma)} \mathsf{cl}(D, \Sigma) \right)$$

since D_A is consistent with Σ. By exploiting this observation, and Lemmas 6.7 and 7.5, we can provide a proof that mimics the one of Lemma 6.9, and show that:

Lemma 7.6. *The following are equivalent:*

1. *There exists $i \in \{1, \ldots, m\}$ such that φ_i is satisfiable and ψ_i is unsatisfiable.*
2. $\mathsf{cert}_{\mathsf{ICR}}(q_{A,B}, D_{A,B}, \Sigma) \neq \varnothing$.

With Lemma 7.6 in place, we can conclude that

$$\#A > \#B \iff \mathsf{cert}_{\mathsf{ICR}}(q_{A,B}, D_{A,B}, \Sigma) \neq \varnothing$$

for a fixed $\Sigma \in \mathsf{C}_\perp$, for $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, which implies that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$ is Θ_2^P-hard in fp-complexity, and the claim follows. ☐

Theorem 7.5. $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}_\perp)$*, where* $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$*, is* co*NP-hard in* d*-complexity.*

The proof of the above exploits the coNP-hard problem of deciding whether a 3CNF formula is unsatisfiable:

PROBLEM :	$\mathsf{3UNSAT}$
INPUT :	A 3CNF Boolean formula φ.
QUESTION :	Is φ unsatisfiable?

We now proceed with the proof of Theorem 7.5.

Proof of Theorem 7.5. The construction of the database is essentially the same as the one given in the previous proof for the database D_B, with the difference that we have to deal only with one formula and not a set of formulas. Although it is easy to modify the construction given above, we give it here for the sake of completeness.

For a 3CNF formula $\varphi = C_1 \wedge \cdots \wedge C_k$ over the variables $x_1, \ldots, x_n$ with $C_i = (\ell_i^1 \vee \ell_i^2 \vee \ell_i^3)$, the database D_φ stores all the possible truth assignments for a clause of φ. To this end, we use a 4-ary predicate $\mathrm{Clause}^{s_1 s_2 s_3}_{b_1 b_2 b_3}$, where $s_1, s_2, s_3 \in \{\mathsf{p}, \mathsf{n}\}$ and $b_1, b_2, b_3 \in \{0, 1\}$. For example, for $C_i = x_{i_1} \vee x_{i_2} \vee \neg x_{i_3}$, the atom

$$\mathrm{Clause}^{\mathsf{ppn}}_{101}(c_i, x_{i_1}, x_{i_2}, x_{i_3})$$

encodes the clause itself, with the superscript ppn indicating that the variable of the first (resp., second, third) literal appears positively (resp., positively, negatively) in C_i, and at the same time encodes the truth assignment that sets x_{i_1} to true, x_{i_2} to false, and x_{i_3} to true. We proceed to formally define D_φ.

As usual, for a literal ℓ, $\mathsf{sign}(\ell) = \mathsf{p}$ (resp., $\mathsf{sign}(\ell) = \mathsf{n}$) if $\ell = x$ (resp., $\ell = \neg x$) for some variable x. For a clause C_i of φ, we write s_i^j for $\mathsf{sign}(\ell_i^j)$, where $j \in \{1, 2, 3\}$. Recall that $\mathsf{var}(\ell)$ is the variable of the literal ℓ. The database D_φ is

$$\bigcup_{1 \leq i \leq k} \; \bigcup_{b_1, b_2, b_3 \in \{0,1\}} \left\{ \mathrm{Clause}^{s_i^1 s_i^2 s_i^3}_{b_1 b_2 b_3} \left(c_i, \mathsf{var}(\ell_i^1), \mathsf{var}(\ell_i^2), \mathsf{var}(\ell_i^3)\right) \right\}.$$

This completes the definition of D_φ.

We claim that there exists a set Σ of TGDs and NCs such that φ is unsatisfiable iff $\mathsf{cert}_{\mathsf{ICR}}(\mathrm{Unsat}(), D_\varphi, \Sigma) \neq \varnothing$. In particular, $\Sigma = \Sigma_{\mathsf{cons}} \cup \Sigma_{\mathsf{unsat}}$ with Σ_{cons} being a set of NCs that performs a consistency check on the truth assignment for φ, i.e., each variable of φ is assigned exactly one value, and Σ_{unsat} being a set of TGDs that entails the ground atom Unsat() whenever a clause of φ evaluates to false. More precisely, Σ_{cons} consists of the following NCs; a $\star$ symbol in the superscript is a placeholder for p or n, a $\star$ in the subscript is a placeholder for 0 or 1, and, as usual, $\_$ is a "don't care" variable that occurs only once:

$$\begin{array}{rcl}
\mathrm{Clause}^{\star\star\star}_{1\star\star}(_, x, _, _), \mathrm{Clause}^{\star\star\star}_{0\star\star}(_, x, _, _) & \rightarrow & \perp \\
\mathrm{Clause}^{\star\star\star}_{1\star\star}(_, x, _, _), \mathrm{Clause}^{\star\star\star}_{\star 0\star}(_, _, x, _) & \rightarrow & \perp \\
\mathrm{Clause}^{\star\star\star}_{1\star\star}(_, x, _, _), \mathrm{Clause}^{\star\star\star}_{\star\star 0}(_, _, _, x) & \rightarrow & \perp \\
\mathrm{Clause}^{\star\star\star}_{\star 1\star}(_, _, x, _), \mathrm{Clause}^{\star\star\star}_{0\star\star}(_, x, _, _) & \rightarrow & \perp \\
\mathrm{Clause}^{\star\star\star}_{\star 1\star}(_, _, x, _), \mathrm{Clause}^{\star\star\star}_{\star 0\star}(_, _, x, _) & \rightarrow & \perp \\
\mathrm{Clause}^{\star\star\star}_{\star 1\star}(_, _, x, _), \mathrm{Clause}^{\star\star\star}_{\star\star 0}(_, _, _, x) & \rightarrow & \perp \\
\mathrm{Clause}^{\star\star\star}_{\star\star 1}(_, _, _, x), \mathrm{Clause}^{\star\star\star}_{0\star\star}(_, x, _, _) & \rightarrow & \perp \\
\mathrm{Clause}^{\star\star\star}_{\star\star 1}(_, _, _, x), \mathrm{Clause}^{\star\star\star}_{\star 0\star}(_, _, x, _) & \rightarrow & \perp \\
\mathrm{Clause}^{\star\star\star}_{\star\star 1}(_, _, _, x), \mathrm{Clause}^{\star\star\star}_{\star\star 0}(_, _, _, x) & \rightarrow & \perp.
\end{array}$$

Moreover, the set of TGDs Σ_{unsat} consists of:

$$\begin{array}{rcl}
\mathrm{Clause}^{\mathsf{nnn}}_{111}(_, _, _, _) & \rightarrow & \mathrm{Unsat}() \\
\mathrm{Clause}^{\mathsf{nnp}}_{110}(_, _, _, _) & \rightarrow & \mathrm{Unsat}() \\
\mathrm{Clause}^{\mathsf{npn}}_{101}(_, _, _, _) & \rightarrow & \mathrm{Unsat}() \\
\mathrm{Clause}^{\mathsf{npp}}_{100}(_, _, _, _) & \rightarrow & \mathrm{Unsat}() \\
\mathrm{Clause}^{\mathsf{pnn}}_{011}(_, _, _, _) & \rightarrow & \mathrm{Unsat}()
\end{array}$$

	c-complexity	ba-complexity	fp-complexity	d-complexity
F	ExpTime	NP	NP	PTime
WG	2ExpTime	ExpTime	ExpTime	ExpTime
WA	2ExpTime	2ExpTime	NP	PTime
WS	2ExpTime	2ExpTime	NP	PTime

Table 5: Complexity of $\mathsf{QAns}(\mathsf{C})$, where $\mathsf{C} \in \{\mathsf{F}, \mathsf{WG}, \mathsf{WA}, \mathsf{WS}\}$; these are completeness results.

$$\begin{array}{rcl} \mathrm{Clause}^{\mathsf{pnp}}_{010}(\_,\_,\_,\_) & \rightarrow & \mathrm{Unsat}() \\ \mathrm{Clause}^{\mathsf{ppn}}_{001}(\_,\_,\_,\_) & \rightarrow & \mathrm{Unsat}() \\ \mathrm{Clause}^{\mathsf{ppp}}_{000}(\_,\_,\_,\_) & \rightarrow & \mathrm{Unsat}(). \end{array}$$

Observe that Σ_{unsat} falls in C, for each $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$. This completes the definition of Σ. By providing a proof that mimics the one for Lemma 7.5, we can show that φ is unsatisfiable iff $\mathsf{cert}_{\mathsf{ICR}}(\mathrm{Unsat}(), D_\varphi, \Sigma) \neq \varnothing$, and the claim follows. □

Remark. Observe that in the above reduction the query is a ground atomic CQ, that is, Unsat(). This fact, together with Lemma 7.4, implies that φ is unsatisfiable iff $\mathsf{cert}_{\mathsf{AR}}(\mathrm{Unsat}(), D_\varphi, \Sigma) \neq \varnothing$. Thus, the above reduction provides an alternative proof for the fact that $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{C}_\bot)$, where $\mathsf{C} \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, is coNP-hard in d-complexity. However, the proof given in Section 5 shows that the coNP-hardness holds even without TGDs, but it exploits a more complex CQ.

8 Full dependencies and beyond

A central class of TGDs, which is incomparable (at the syntax level) to all the classes that we have seen so far, is the class of *full TGDs*, i.e., TGDs without existentially quantified variables, which we denote by F. Indeed, this class forms a powerful language for modeling ontologies that has been used in several different scenarios. For example, it is known that the logical core of the RL profile of OWL 2, which is aimed at applications that require efficient reasoning without sacrificing too much expressive power, corresponds to full TGDs.[7] Interestingly, the main classes of TGDs that we have seen in the previous sections based on the notions of guardedness, acyclicity and stickiness, come with their "weakly" version that incorporates full TGDs: *weakly-guarded* (WG) [10], *weakly-acyclic* (WA) [21], and *weakly-sticky* (WS), respectively. The definition of all these "weakly" versions follows the same principle: the underlying syntactic condition is relaxed in such a way that only certain "harmful" variables are taken into account; for details we refer the reader to the references given above.

The complexity of $\mathsf{QAns}(\mathsf{C})$, where $\mathsf{C} \in \{\mathsf{F}, \mathsf{WG}, \mathsf{WA}, \mathsf{WS}\}$, is by now well-understood and is summarized in Table 5. The results for $\mathsf{QAns}(\mathsf{F})$ are coming from the Datalog literature since a set of full TGDs is essentially a Datalog program [17]. For all the other classes, we refer the reader to the references mentioned above. But what about the complexity of consistent query answering under the semantics that we have seen so far, when the above classes of TGDs are combined with NCs? It turned out that the analysis performed in the previous sections for the less expressive classes of TGDs allows us to easily complete the picture.

Theorem 8.1. *The* t*-complexity of* $\mathsf{QAns}_{\mathsf{s}}(\mathsf{C}_\bot)$*, where* $\mathsf{t} \in \{\mathsf{c}, \mathsf{ba}, \mathsf{fp}, \mathsf{d}\}$*,* $\mathsf{s} \in \{\mathsf{AR}, \mathsf{IAR}, \mathsf{ICR}\}$*, and* $\mathsf{C} \in \{\mathsf{F}, \mathsf{WG}, \mathsf{WA}, \mathsf{WS}\}$*, is as shown in Table 6.*

Let us briefly summarize how the above complexity results are obtained by exploiting the algorithms and the reductions devised in the previous sections:

AR **semantics.** The upper bounds are obtained via the procedure AlgorithmAR. The 2ExpTime and ExpTime lower bounds are inherited from $\mathsf{QAns}(\mathsf{C})$, while the Π^P_2 and coNP lower bounds are inherited from $\mathsf{QAns}_{\mathsf{AR}}(\mathsf{NC})$.

IAR **semantics.** All the upper bounds, apart from the Θ^P_2 ones, are obtained via AlgorithmIAR1, while the Θ^P_2 upper bounds via AlgorithmIAR2. As above, the 2ExpTime and ExpTime lower bounds are inherited from $\mathsf{QAns}(\mathsf{C})$, while the Π^P_2 one from $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{NC})$. For the Θ^P_2 and coNP lower bounds,

[7] https://www.w3.org/TR/owl2-profiles/#OWL_2_RL

		c-complexity	ba-complexity	fp-complexity	d-complexity
$\mathsf{F}_\perp$	AR	ExpTime	Π_2^P	Π_2^P	coNP
	IAR	ExpTime	Π_2^P	Θ_2^P	coNP
	ICR	ExpTime	Π_2^P	Θ_2^P	coNP
$\mathsf{WG}_\perp$	AR	2ExpTime	ExpTime	ExpTime	ExpTime
	IAR	2ExpTime	ExpTime	ExpTime	ExpTime
	ICR	2ExpTime	ExpTime	ExpTime	ExpTime
$\mathsf{WA}_\perp$	AR	2ExpTime	2ExpTime	Π_2^P	coNP
	IAR	2ExpTime	2ExpTime	Θ_2^P	coNP
	ICR	2ExpTime	2ExpTime	Θ_2^P	coNP
$\mathsf{WS}_\perp$	AR	2ExpTime	2ExpTime	Π_2^P	coNP
	IAR	2ExpTime	2ExpTime	Θ_2^P	coNP
	ICR	2ExpTime	2ExpTime	Θ_2^P	coNP

Table 6: Complexity of $\mathsf{QAns}_\mathsf{s}(\mathsf{C}_\perp)$, where $\mathsf{C} \in \{\mathsf{F}, \mathsf{WG}, \mathsf{WA}, \mathsf{WS}\}$. For each class, the first (resp., second, third) row corresponds to AR (resp., IAR, ICR); these are completeness results.

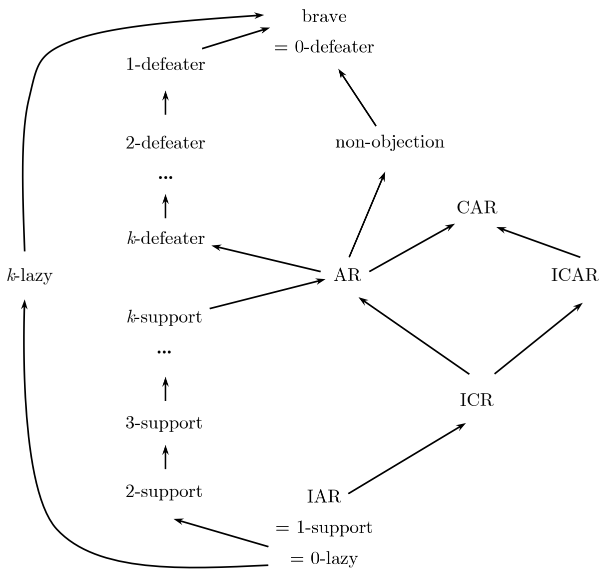


Figure 1: Complete picture of the relationships among inconsistent-tolerant semantics [4]. The semantics to which the arrow points is a complete approximation of the semantics from where the arrow starts; e.g., both AR and ICAR entail all the ICR answers, and ICR all the IAR answers.

it suffices to observe that the proof for the fact that $\mathsf{QAns}_{\mathsf{IAR}}(\mathsf{G}_\perp)$ is Θ_2^P-hard in fp-complexity and coNP-hard in d-complexity exploits only full TGDs.

ICR **semantics.** The c-complexity upper bounds are obtained via AlgorithmICR1. The ba- and d-complexity upper bounds, as well as the fp-complexity upper bound in the case of $\mathsf{WG}_\perp$, are obtained via AlgorithmICR3. The Θ_2^P upper bounds are established by using AlgorithmICR4. The 2ExpTime and ExpTime lower bounds are inherited from $\mathsf{QAns}(\mathsf{C})$, while the Π_2^P one from $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{NC})$. Finally, for the Θ_2^P and coNP lower bounds, observe that the proofs for showing that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{C}'_\perp)$ is Θ_2^P-hard in fp-complexity and coNP-hard in d-complexity, for $\mathsf{C}' \in \{\mathsf{L}, \mathsf{A}, \mathsf{S}\}$, use only full TGDs.

9 Related work

There has been an extensive body of work on querying inconsistent knowledge bases in the context of DL and existential rule languages. Arguably, as discussed in the introduction, the AR, IAR, and ICR semantics have been the most prominent inconsistency-tolerant semantics. The AR semantics (known in the database literature as consistent query answering) was first developed for relational databases in [1], and then applied to several DLs in [28, 31]. Intuitively, the AR semantics entails the set of answers that are classically entailed in every possible repair. The intractability of the AR semantics was first established in [31], which showed that ontological UCQ answering is coNP-complete in data complexity. This result was then strengthened in [28], which showed that the coNP-hardness holds even for ground atomic queries and when the knowledge base is expressed in $\mathit{DL\text{-}Lite}_{core}$ (the least expressive logic in the *DL-Lite* family). The work of [44] studied both the data and the combined complexity for a wide spectrum of DLs, while [3] identified cases for simple ontologies (within the *DL-Lite* family) for which tractable data complexity results can be obtained. In [37, 38, 40], the data and different types of combined complexity of the AR semantics have been studied for ontologies modeled via existential rules and negative constraints.

The IAR semantics was introduced in [28] as a sound (under-)approximation of AR, as it entails the set of answers that are classically entailed from the intersection of all repairs. The work of [28] showed that ontological UCQ answering is in PTime in data complexity for $\mathit{DL\text{-}Lite}_{\mathcal{A}}$. On the other hand, [29] showed that ontological CQ answering under the IAR semantics is first-order rewritable for DLs of the *DL-Lite* family. The combined complexity of the IAR semantics for ontology languages of the *DL-Lite* family was investigated in [9]. The work of [6] analyzed the data and combined complexity of ontological query answering under the AR and IAR semantics for different notions of maximal repairs focusing on the lightweight logic $\mathit{DL\text{-}Lite}_{\mathcal{R}}$. Practical implementations of the AR and IAR semantics have been developed in [7, 30].

The ICR semantics was introduced in [3], where it was also shown that ontological CQ answering is in PTime in data complexity for simple DL ontologies. The ICR semantics entails the set of classical answers obtained from the intersection of the logical closure of all possible repairs, and it is an over-approximation of IAR (i.e., IAR answers are ICR answers, but the reverse does not hold) and an under-approximation of AR. The complexity of ontological query answering under the IAR and ICR semantics for a wide range of existential rule languages and for different complexity measures has been investigated in [34]. The work of [36] investigated the complexity of ontological query answering under the AR, IAR, and ICR semantics for several existential rule languages and complexity measures when repairs are cardinality-maximal.

The work of [28] also introduces other semantics that under and over approximate AR, namely CAR and ICAR, which stand for *Closed ABox Repairs* and *Intersection of Closed ABox Repairs*, respectively. The rationale for the CAR semantics is that AR is dependant of the syntax of the database, which means that logically equivalent knowledge bases may yield different answers under the AR semantics. CAR is an over-approximation of AR based on repairs that are computed from the consistent closure of the database with respect to the (DL-based or rule-based) ontology Σ; intuitively, the consistent closure is the set of all atoms that can be consistently derived from the database and Σ, i.e., such that no negative axiom is violated in the derivation. A repair is now any subset of the consistent closure that "maximally preserves" the content of the original database. The CAR semantics corresponds then to the set of answers that are classically entailed from every closed repair, and contains all AR answers. Therefore, it is a complete approximation, but there are answers that are true under the CAR semantics that are not true under the AR semantics. In [3, 44] it is shown that ontological query answering for $\mathit{DL\text{-}Lite}_{core}$ under CAR is in PTime in data complexity when we focus on atomic queries, and coNP-complete for UCQs. It has been also shown that for the DL *EL*, ontological UCQ answering under CAR is DP-complete in data complexity. Analogously to IAR, [28] defines the ICAR semantics, a sound approximation to CAR that computes the answers from the intersection of all closed repairs.

Following these families of under and over approximations to the AR semantics, further semantics where developed trying to formalize more granular conflict resolution techniques. The notion of k-lazy consistent answers, proposed in [38], provides an alternative semantics that offers a compromise between quality of answers and computation time. Lazy answers are based on a "budget" (the parameter k) that restricts the size of removals that need to be made in an inconsistent set of facts in order to make it consistent; if the budget is large enough, then all possible ways of resolving the conflicts within the budget are considered, but if it is not enough then the whole inconsistent set is removed. If we think of the problem of querying inconsistent KBs as a reasoning task for an intelligent agent, then the value of the budget would be a bound on its reasoning capabilities (more complex reasoning can thus be afforded with higher budgets). The k-lazy semantics is non-monotonic with respect to k in the sense that the answers obtained under the semantics with parameter k may not be a subset of those obtained with $k+1$; nevertheless, the union-k-lazy extension proposed in [41] allows to monotonically expand the set of consistent answers. Although the k-lazy approach is not strictly based on the same notion of repair, it was shown that there always exists a value k for which k-lazy and AR coincide.

The k-support semantics [9] increasingly produces more fine-grained under (sound) approximations of the AR semantics. On the other hand, [9] also proposed the k-defeater semantics, which provides increasingly tighter upper (complete) approximations of the AR semantics. The k-support semantics restricts the number of distinct supports (i.e., consistent derivations for an answer) that can cover all the repairs; with $k = 1$, the same support must be present in every repair, so it coincides with IAR – increasing parameter k yields larger sets of answers until AR is reached. On the other hand, in the k-defeater semantics, only sets of size k that create a contradiction w.r.t. every minimal support for an answer are considered; clearly, when $k = 0$ the semantics coincides with the brave semantics, i.e., the set of answers that can be obtained from some repair. As k increases, larger defeater sets are considered and the set of answers are incrementally reduced until the set of AR answers is reached. Both semantics enjoy desirable computational properties; however, note that k-defeater may entail answers that are conflicting among each other (even for the same value of k). Figure 1, reproduced from [4], summarizes the relationship among the main inconsistency-tolerant semantics defined so far in the literature. We refer the reader to [4] for more details on the semantics and complexity results for several families of DLs.

The AR semantics was extended to the *generalized repair (GR)* semantics, and its computational complexity was analyzed in [20]. In the GR semantics, not only atoms from the database, but also ontological axioms may be removed and considered as part of the repairs; notice, however, that some database atoms and axioms may be specified to be non-removable. The generalized repair semantics was applied to the IAR and ICR semantics in [34], where its complexity was analyzed for different existential rule languages and complexity measures.

Recently, inconsistency-tolerant semantics for ontological query answering has also been considered from an explanation perspective. A (minimal) explanation for an ontological query can be defined in different ways. However, the literature has lately focused on a quite natural definition in which a (minimal) explanation for an ontological query is a (minimal) set of facts that, together with the ontology, entail the query (see [7, 15] for the DL setting, and [14] for the existential rule setting, and the references therein). This concept of explanation has been extended to the inconsistency-tolerant semantics, in which, intuitively, an explanation in terms of (sets of) facts is provided to justify why a query is entailed under the AR, IAR, ICR, semantics (see [7] for the DL setting, and [35] for the existential rule setting).

10 Conclusion

We performed a thorough complexity analysis of consistent query answering under the main classes of TGDs based on the notions of guardedness, linearity, acyclicity, and stickiness, and extensions thereof. In our analysis, we focused on the standard inconsistency-tolerant semantics (AR semantics), as well as the main sound approximations of it (IAR and ICR semantics), and we considered different complexity measures with the aim of understanding how the complexity is affected when key parameters of the input are considered to be fixed.

Another goal of our analysis, apart from clarifying the complexity landscape, was to understand whether the IAR and ICR semantics have the desired effect on the data complexity of our problem, i.e., whether they lead to tractability. It turned out that this is not the case, as the problem remains coNP-hard in most of the cases. The only exceptions are the classes based on linearity, acyclicity, and stickiness when we focus on the IAR semantics. In these cases, we show that the problem is in AC_0 in data complexity. This is established via FO-rewritability, which in turn relies on the fact that the classes in question are UCQ-rewritable.

As for future work, apart from performing a complexity analysis with other semantics, we believe that it is important to empirically evaluate the performance of the various inconsistency-tolerant semantics with respect to their expressive power. Many of the semantics proposed in the literature have been designed to trade off expressive power for computational tractability. Many also maintain soundness with respect to the AR semantics as a kind of quality guarantee that is difficult to evaluate in practice. One way to compare the performance of the various alternative semantics is therefore to design quality metrics that yield objective comparisons of expressive power in practice. This would also shed light on the performance of semantics that go beyond the classical concept of data repair, which are typically not designed to be sound with respect to the AR semantics. The main challenges in this line of work involve selecting adequate real-world datasets, as well as designing well-founded methods to synthetically generate datasets.

Acknowledgments

We would like to thank Marco Calautti for bringing to our attention Corollary 4 from [46], which was crucial for showing that $\mathsf{QAns}_{\mathsf{ICR}}(\mathsf{A}_{\bot})$ is in $\mathrm{P}^{\mathrm{NExpTime}}$ in c-complexity. This work was supported by the UK EPSRC grants EP/J008346/1, EP/R013667/1, EP/L012138/1, and EP/M025268/1, by the AXA Research Fund, and by the Alan Turing Institute under the EPSRC grant EP/N510129/1. M.V. Martinez and G.I. Simari were partially supported by Proyecto PIP-CONICET 112-201101-01000. A. Pieris was also supported by the EPSRC grant EP/S003800/1.

References

[1] M. Arenas, L. E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: PODS, 1999, pp. 68–79.

[2] C. Beeri, M. Y. Vardi, The implication problem for data dependencies, in: ICALP, 1981, pp. 73–85.

[3] M. Bienvenu, On the complexity of consistent query answering in the presence of simple ontologies, in: AAAI, 2012, pp. 705—711.

[4] M. Bienvenu, Inconsistency handling in ontology-mediated query answering: A progress report, in: DL, 2019.

[5] M. Bienvenu, C. Bourgaux, Inconsistency-tolerant querying of description logic knowledge bases, in: Reasoning Web, 2016, pp. 156–202.

[6] M. Bienvenu, C. Bourgaux, F. Goasdoué, Querying inconsistent description logic knowledge bases under preferred repair semantics, in: AAAI, 2014, pp. 996–1002.

[7] M. Bienvenu, C. Bourgaux, F. Goasdoué, Computing and explaining query answers over inconsistent dl-lite knowledge bases, J. Artif. Intell. Res. 64 (2019) 563–644.

[8] M. Bienvenu, R. Rosati, New inconsistency-tolerant semantics for robust ontology-based data access, in: DL, 2013, pp. 53–64.

[9] M. Bienvenu, R. Rosati, Tractable approximations of consistent query answering for robust ontology-based data access, in: IJCAI, 2013, pp. 775–781.

[10] A. Calì, G. Gottlob, M. Kifer, Taming the infinite chase: Query answering under expressive relational constraints, J. Artif. Intell. Res. 48 (2013) 115–174.

[11] A. Calì, G. Gottlob, T. Lukasiewicz, A general Datalog-based framework for tractable query answering over ontologies, J. Web Sem. 14 (2012) 57–83.

[12] A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, A. Pieris, Datalog+/-: A family of logical knowledge representation and query languages for new applications, in: LICS, 2010, pp. 228–242.

[13] A. Calì, G. Gottlob, A. Pieris, Towards more expressive ontology languages: The query answering problem, Artif. Intell. 193 (2012) 87–128.

[14] İ. İ. Ceylan, T. Lukasiewicz, E. Malizia, A. Vaicenavičius, Explanations for query answers under existential rules, in: IJCAI, 2019, pp. 1639–1646.

[15] İ. İ. Ceylan, T. Lukasiewicz, E. Malizia, A. Vaicenavičius, Explanations for ontology-mediated query answering in description logics, in: ECAI, 2020, pp. 672–679.

[16] J. Chomicki, J. Marcinkowski, Minimal-change integrity maintenance using tuple deletions, Inf. Comput. 197 (2005) 90–121.

[17] E. Dantsin, T. Eiter, G. Gottlob, A. Voronkov, Complexity and expressive power of logic programming, ACM Comput. Surv. 33 (2001) 374–425.

[18] E. Dantsin, A. Voronkov, Complexity of query answering in logic databases with complex values, in: LFCS, 1997, pp. 56–66.

[19] A. Deutsch, A. Nash, J. B. Remmel, The chase revisited, in: PODS, 2008, pp. 149–158.

[20] T. Eiter, T. Lukasiewicz, L. Predoiu, Generalized consistent query answering under existential rules, in: KR, 2016, pp. 359–368.

[21] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa, Data exchange: Semantics and query answering, Theor. Comput. Sci. 336 (2005) 89–124.

[22] G. Gottlob, E. Malizia, Achieving new upper bounds for the hypergraph duality problem through logic, in: CSL-LICS, 2014, pp. 43:1–43:10.

[23] G. Gottlob, M. Manna, A. Pieris, Polynomial combined rewritings for existential rules, in: KR, 2014, pp. 268–277.

[24] G. Gottlob, G. Orsi, A. Pieris, Query rewriting and optimization for ontological databases, ACM Trans. Database Syst. 39 (2014) 25:1–25:46.

[25] G. Greco, E. Malizia, L. Palopoli, F. Scarcello, On the complexity of core, kernel, and bargaining set, Artif. Intell. 175 (2011) 1877–1910.

[26] L. A. Hemachandra, The strong exponential hierarchy collapses, J. Comput. Syst. Sci. 39 (1989) 299–322.

[27] D. S. Johnson, A catalog of complexity classes, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science (Vol. A), Elsevier, 1990, pp. 67–161.

[28] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, D. F. Savo, Inconsistency-tolerant semantics for description logics, in: RR, 2010, pp. 103–117.

[29] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, D. F. Savo, Query rewriting for inconsistent dl-lite ontologies, in: RR, 2011, pp. 155–169.

[30] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, D. F. Savo, Inconsistency-tolerant query answering in ontology-based data access, J. Web Semant. 33 (2015) 3–29.

[31] D. Lembo, M. Ruzzi, Consistent query answering over description logic ontologies, in: RR, 2007, pp. 194–208.

[32] T. Lukasiewicz, E. Malizia, On the complexity of mCP-nets, in: AAAI, 2016, pp. 558–564.

[33] T. Lukasiewicz, E. Malizia, A novel characterization of the complexity class θ_k^P based on counting and comparison, Theor. Comput. Sci. 694 (2017) 21–33.

[34] T. Lukasiewicz, E. Malizia, C. Molinaro, Complexity of approximate query answering under inconsistency in Datalog+/-, in: IJCAI, 2018, pp. 1921–1927.

[35] T. Lukasiewicz, E. Malizia, C. Molinaro, Explanations for inconsistency-tolerant query answering under existential rules, in: AAAI, 2020, pp. 2909–2916.

[36] T. Lukasiewicz, E. Malizia, A. Vaicenavicius, Complexity of inconsistency-tolerant query answering in datalog+/- under cardinality-based repairs, in: AAAI, 2019, pp. 2962–2969.

[37] T. Lukasiewicz, M. V. Martinez, A. Pieris, G. I. Simari, From classical to consistent query answering under existential rules, in: AAAI, 2015, pp. 1546–1552.

[38] T. Lukasiewicz, M. V. Martinez, G. I. Simari, Inconsistency handling in Datalog+/- ontologies, in: ECAI, 2012, pp. 558–563.

[39] T. Lukasiewicz, M. V. Martinez, G. I. Simari, Inconsistency-tolerant query rewriting for linear Datalog+/-, in: Datalog 2.0, 2012, pp. 123–134.

[40] T. Lukasiewicz, M. V. Martinez, G. I. Simari, Complexity of inconsistency-tolerant query answering in Datalog+/-, in: ODBASE, 2013, pp. 488–500.

[41] M. V. Martinez, G. I. Simari, Explanation-friendly query answering under uncertainty, in: Reasoning Web, 2019, pp. 65–103.

[42] C. H. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

[43] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. Data Semantics 10 (2008) 133–173.

[44] R. Rosati, On the complexity of dealing with inconsistency in description logic ontologies, in: IJCAI, 2011, pp. 1057–1062.

[45] M. Schaefer, Graph ramsey theory and the polynomial hierarchy, J. Comput. Syst. Sci. 62 (2001) 290–322.

[46] U. Schöning, K. W. Wagner, Collapsing oracle hierarchies, census functions and logarithmically many queries, in: STACS, 1988, pp. 91–97.

[47] M. Y. Vardi, On the complexity of bounded-variable queries, in: PODS, 1995, pp. 266–276.