Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

From Classical to Consistent Query Answering under Existential Rules

Thomas Lukasiewicz[1] **Maria Vanina Martinez**[2] **Andreas Pieris**[3] **Gerardo I. Simari**[2]

[1]Department of Computer Science, University of Oxford, UK
[2]Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur and CONICET, Argentina
[3]Institute of Information Systems, Vienna University of Technology, Austria

thomas.lukasiewicz@cs.ox.ac.uk, {mvm,gis}@cs.uns.edu.ar,
pieris@dbai.tuwien.ac.at

Abstract

Querying inconsistent ontologies is an intriguing new problem that gave rise to a flourishing research activity in the description logic (DL) community. The computational complexity of consistent query answering under the main DLs is rather well understood; however, little is known about existential rules. The goal of the current work is to perform an in-depth analysis of the complexity of consistent query answering under the main decidable classes of existential rules enriched with negative constraints. Our investigation focuses on one of the most prominent inconsistency-tolerant semantics, namely, the AR semantics. We establish a generic complexity result, which demonstrates the tight connection between classical and consistent query answering. This result allows us to obtain in a uniform way a relatively complete picture of the complexity of our problem.

Introduction

An ontology is an explicit specification of a conceptualization of an area of interest. One of the main applications of ontologies is in ontology-based data access (OBDA) (Poggi et al. 2008), where they are used to enrich the extensional data with intensional knowledge. In this setting, description logics (DLs) and rule-based formalisms such as existential rules are popular ontology languages, while conjunctive queries (CQs) form the central querying tool. In real-life applications, involving large amounts of data, it is possible that the data are inconsistent with the ontology. Inconsistencies of this kind may result from automated procedures such as data integration and ontology matching. Since standard ontology languages adhere to the classical FOL semantics, inconsistencies are nothing else than logical contradictions. Thus, the classical inference semantics fails terribly when faced with an inconsistency, since everything follows from a contradiction. This demonstrates the need for developing inconsistency-tolerant semantics for ontological reasoning.

There has been a recent and increasing focus on the development of such semantics for query answering purposes. Consistent query answering, first developed for relational databases (Arenas, Bertossi, and Chomicki 1999) and then generalized as the AR semantics for several DLs (Lembo et al. 2010), is the most widely accepted semantics for querying inconsistent ontologies. 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. Obtaining the set of consistent answers under the AR semantics is known to be a hard problem, even for very simple languages (Lembo et al. 2010). For this reason, several other semantics have been recently developed with the aim of approximating the set of consistent answers (Lembo et al. 2010; Bienvenu 2012; Lukasiewicz, Martinez, and Simari 2012; Bienvenu and Rosati 2013).

It is widely accepted that the variety of ontologies underlying practical applications requires a good understanding of the computational complexity of inconsistency-tolerant semantics. In this work, we are interested in the AR semantics. The complexity of query answering under the AR semantics (and also under several other semantics) when the ontology is described using one of the central DLs is rather well understood. The data and combined complexity were studied in (Rosati 2011) for a wide spectrum of DLs, while the work (Bienvenu 2012) identifies cases for simple ontologies (within the *DL-Lite* family) for which tractable data complexity results can be obtained.

Although the AR semantics has been thoroughly studied for several key DLs, little is known when the ontology is described using existential rules, that is, formulas of the form $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \exists \mathbf{Y}\, p(\mathbf{X}, \mathbf{Y})$, and negative constraints of the form $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \bot$, where $\bot$ denotes the truth constant *false*. An exception are the works (Lukasiewicz, Martinez, and Simari 2012; 2013), where the data complexity of the AR semantics is studied for several classes of existential rules enriched with negative constraints. Notice that existential rules are also known as tuple-generating dependencies (TGDs) and Datalog$^{\pm}$ rules (Calì et al. 2010); henceforth, for brevity, we adopt the term TGDs.

Our main goal in this work is to perform an in-depth analysis of the combined complexity of query answering under the main decidable classes of TGDs, enriched with negative constraints, focusing on the AR semantics. Let us recall that the main (syntactic) conditions on TGDs that guarantee the decidability of CQ answering are guardedness (Calì, Gottlob, and Kifer 2013), stickiness (Calì, Gottlob, and Pieris 2012) and acyclicity. Another important fragment of TGDs, which deserves our attention, is the class of full TGDs, i.e., existential-free TGDs (Abiteboul, Hull, and Vianu 1995). Our second goal is to understand whether the combined complexity of consistent query answering under full TGDs, enriched with negative constraints, is affected or not if we further assume that the given set of TGDs enjoys guardedness, stickiness or acyclicity. Apart from the combined complexity, we would also like to understand how the complexity of our problem is affected when some key parameters are fixed. In particular, we consider the following two variants of the combined complexity: (1) the bounded-arity combined complexity (or simply ba-combined complexity), which is calculated by assuming that the arity of the underlying schema is bounded; and (2) the fixed-program combined complexity (or simply *fp*-combined complexity), which is calculated by considering the set of TGDs and negative constraints as fixed (the set of constraints is usually called program, and hence the term "fixed program"). Notice that, in practice, the arity of the schema is usually small and can be productively assumed to be fixed. Moreover, the components which change quite often over time are the database and the query, while the program remains the same. Hence, the preceding types of complexity are meaningful metrics that deserve to be investigated.

Interestingly, our complexity analysis shows that a systematic and uniform way for transferring complexity results from classical to consistent query answering can be formally established. To briefly summarize the main contributions:

- We present a generic complexity result, which demonstrates the tight connection between classical and consistent query answering (Theorem 3).
- By exploiting our generic theorem, we obtain a (nearly) complete picture of the (*ba*-/*fp*-)combined complexity of consistent query answering (Table 1).
- Finally, as we transition from classical to consistent query answering, several novel complexity results on classical query answering are established (Tables 2 and 3).

Preliminaries

General. Consider the following sets: a set $\mathbf{C}$ of *constants*, a set $\mathbf{N}$ of *labeled nulls*, and a set $\mathbf{V}$ of regular *variables*. A *term* t is a constant, null, or variable. An *atom* has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. Conjunctions of atoms are often identified with the sets of their atoms. An *instance* I is a (possibly infinite) set of atoms $p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of constants and nulls. A *database* D is a finite instance that contains only constants. A *homomorphism* is a substitution $h : \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ that is the identity on $\mathbf{C}$. We assume the reader is familiar with *conjunctive queries (CQs)*. The answer to a CQ q over an instance I is denoted $q(I)$. A Boolean CQ (BCQ) q has a positive answer over I, denoted $I \models q$, if $q(I) \neq \varnothing$.

Dependencies. A *tuple-generating dependency (TGD)* σ is a first-order formula $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \exists \mathbf{Y}\, p(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} \cup \mathbf{Y} \subset \mathbf{V}$, $\varphi(\mathbf{X})$ is a conjunction of atoms, and $p(\mathbf{X}, \mathbf{Y})$ is an atom; $\varphi(\mathbf{X})$ is the *body* of σ, denoted $body(\sigma)$, while $p(\mathbf{X}, \mathbf{Y})$ is the *head* of σ, denoted $head(\sigma)$. For clarity, we consider single-atom-head TGDs; however, our results can be extended to TGDs with a conjunction of atoms in the head. An instance I satisfies σ, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$, then there exists $h' \supseteq h_{|\mathbf{X}}$, where $h_{|\mathbf{X}}$ is the restriction of h on $\mathbf{X}$, such that $h'(p(\mathbf{X}, \mathbf{Y})) \in I$. A *negative constraint (NC)* ν is a first-order formula of the form $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \bot$, where $\mathbf{X} \subset \mathbf{V}$, $\varphi(\mathbf{X})$ is a conjunction of atoms and is called the *body* of ν, denoted $body(\nu)$, and $\bot$ denotes the truth constant $false$. An instance I satisfies ν, written $I \models \nu$, if there is no homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$. Given a set Σ of TGDs and NCs, I satisfies Σ, written $I \models \Sigma$, if I satisfies each TGD and NC of Σ. For brevity, we omit the universal quantifiers in front of TGDs and NCs, and use the comma (instead of $\wedge$) for conjoining body atoms. Given a class of TGDs $\mathbb{C}$, we denote by $\mathbb{C}_\bot$ the formalism obtained by combining $\mathbb{C}$ with arbitrary NCs.

Conjunctive Query Answering. Given a database D and a set Σ of TGDs and NCs, the answers we consider are those that are true in *all* models of D and Σ. Formally, the *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of instances $\{I \mid I \supseteq D \text{ and } I \models \Sigma\}$. The *answer* to a CQ q w.r.t. D and Σ is defined as the set of tuples $ans(q, D, \Sigma) = \bigcap_{I \in mods(D,\Sigma)} \{\mathbf{t} \mid \mathbf{t} \in q(I)\}$. The answer to a BCQ q is *positive*, denoted $D \cup \Sigma \models q$, if $ans(q, D, \Sigma) \neq \varnothing$. The problem of *CQ answering* is defined as follows: given a database D, a set Σ of TGDs and NCs, a CQ q, and a tuple of constants $\mathbf{t}$, decide whether $\mathbf{t} \in ans(q, D, \Sigma)$. It is well-known that CQ answering can be reduced in LOGSPACE to BCQ answering, and we thus focus on BCQs. Henceforth, by CQ, we refer to a BCQ. Following Vardi's taxonomy (1982), the *combined complexity* of CQ answering is calculated by considering all the components, i.e., the database, the set of dependencies, and the query, as part of the input. The *bounded-arity combined complexity* (or simply *ba-combined complexity*) is calculated by assuming that the arity of the underlying schema is bounded by an integer constant. Notice that in the context of description logics, whenever we refer to the combined complexity in fact we refer to the ba-combined complexity since, by definition, the arity of the underlying schema is at most two. The *fixed-program combined complexity* (or simply *fp-combined complexity*) is calculated by considering the set of TGDs and NCs as fixed.

Consistent Query Answering. In the classical setting of CQ answering, given a database D and a set Σ of TGDs and NCs, if $mods(D, \Sigma) = \varnothing$, then every query is entailed since everything is inferred from a contradiction.

Example 1 Consider the database D defined as

$$\{Prof(p), Postdoc(p), Researcher(p), leaderOf(p, g)\},$$

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

$$\begin{array}{rcl} Prof(X) & \rightarrow & Researcher(X) \\ Postdoc(X) & \rightarrow & Researcher(X) \\ Prof(X), Postdoc(X) & \rightarrow & \bot \\ leaderOf(X, Y) & \rightarrow & Prof(X) \\ leaderOf(X, Y) & \rightarrow & Group(Y), \end{array}$$

expressing that professors and postdocs are researchers, professors and postdocs form disjoint sets, and $leaderOf$ has $Prof$ as domain and $Group$ as range. It is easy to see that $mods(D, \Sigma) = \varnothing$, since p violates the disjointness constraint; therefore, for every CQ q, $D \cup \Sigma \models q$. ∎

Clearly, the answers that we obtain from databases that are inconsistent with the given set of TGDs and NCs are not meaningful for practical applications. For this reason, several inconsistency-tolerant semantics have been proposed in the literature. In this work, we focus on one of the central and well-accepted inconsistency-tolerant semantics, that is, the AR semantics. A key notion, which is necessary for defining the AR semantics, is that of repair, which is $\subseteq$-maximal consistent subset of the given database. Fix a database D, a set Σ of TGDs and NCs, and a CQ q.

Definition 1 A *repair* of D and Σ is some $D' \subseteq D$ such that (i) $mods(D', \Sigma) \neq \varnothing$; and (ii) there is no $\underline{a} \in D \setminus D'$ for which $mods(D' \cup \{\underline{a}\}, \Sigma) \neq \varnothing$. We denote by $drep(D, \Sigma)$ the set of repairs of D and Σ.

Example 2 Consider the database D and the set Σ of TGDs and NCs given in Example 1. The set of repairs of D and Σ consists of the following subsets of D:

$$\begin{array}{rcl} D_1 & = & \{Prof(p), Researcher(p), leaderOf(p, g)\} \\ D_2 & = & \{Postdoc(p), Researcher(p)\}. \end{array}$$

To obtain D_1 it suffices to remove the atom $Postdoc(p)$ from D. However, to obtain D_2, apart from eliminating $Prof(p)$, we also need to remove the atom $leaderOf(p, g)$, which, together with the TGD $leaderOf(X, Y) \rightarrow Prof(X)$, implies the atom $Prof(p)$. ∎

The AR semantics (Lembo et al. 2010) is based on the idea that a query can be considered to hold if it can be inferred from each of the repairs.

Definition 2 The query q is entailed by D and Σ under the *AR semantics*, written $D \cup \Sigma \models_{AR} q$, if $D' \cup \Sigma \models q$, for every $D' \in drep(D, \Sigma)$.

Example 3 Consider the database D and the set Σ of TGDs and NCs given in Example 1, and also the CQs

$$\begin{array}{rcl} q_1 & = & \exists X\, Researcher(X) \\ q_2 & = & \exists X \exists Y\, Researcher(X) \wedge leaderOf(X, Y). \end{array}$$

The former asks whether a researcher exists, while the latter asks whether a researcher, who is also the leader of a group, exists. Assume that D_1 and D_2 are the repairs of D and Σ, as given in Example 2. It easy to verify that $D_i \cup \Sigma \models q_1$, for each $i \in \{1, 2\}$, and thus $D \cup \Sigma \models_{AR} q_1$. On the other hand, although $D_1 \cup \Sigma \models q_2$, $D_2 \cup \Sigma \not\models q_2$, which implies that $D \cup \Sigma \not\models_{AR} q_2$. ∎

We refer to consistent CQ answering under the AR semantics as AR-CQ answering.

Complexity Classes. In our later complexity analysis, beside the standard complexity classes NP, PSPACE, EXPTIME, NEXPTIME and 2EXPTIME, we will also mention the following classes of the polynomial hierarchy: (i) Σ_2^p, the class of problems that can be solved in non-deterministic polynomial time using an NP-oracle; and (ii) Π_2^p, the complement of Σ_2^p. Another hierarchy of classes, which is relevant for our complexity analysis, is the strong exponential hierarchy (SEH) (Hemachandra 1989). A key class is P^{NE}, that is, the class of problems that can be solved in polynomial time using an NE-oracle, which is the Δ_2 level of the SEH. Recall that $\text{NE} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(2^{kn})$, i.e., the class of problems that can be solved in non-deterministic exponential time with linear exponent.

AR-CQ Answering: An Overview

As said, the main objective of the current work is to investigate the (*ba-*/*fp-*)combined complexity of AR-CQ answering under the main decidable classes of TGDs, enriched with arbitrary NCs. But let us first briefly recall those classes.

Decidability Paradigms. The main (syntactic) conditions on TGDs that guarantee the decidability of CQ answering are guardedness (Calì, Gottlob, and Kifer 2013), stickiness (Calì, Gottlob, and Pieris 2012) and acyclicity. Interestingly, each one of those conditions has its "weakly" counterpart: weak-guardedness (Calì, Gottlob, and Kifer 2013), weak-stickiness (Calì, Gottlob, and Pieris 2012) and weak-acyclicity (Fagin et al. 2005), respectively.

A TGD σ is called *guarded* if there exists an atom $\underline{a} \in body(\sigma)$ which contains (or "guards") all the body variables of σ. The class of guarded TGDs, denoted G, is defined as the family of all possible sets of guarded TGDs. A key subclass of guarded TGDs are the so-called linear TGDs with just one body atom (which is automatically a guard), and the corresponding class is denoted L. *Weakly-guarded* TGDs extend guarded TGDs by requiring only "harmful" body variables to appear in the guard, and the associated class is denoted WG. It is easy to verify that $\mathsf{L} \subset \mathsf{G} \subset \mathsf{WG}$.

Stickiness is inherently different from guardedness, and its central property can be described as follows: variables that appear more than once in a body (i.e., join variables) are always propagated (or "stick") to the inferred atoms. A set of TGDs that enjoys the above property is called *sticky*, and the corresponding class is denoted S. Weak-stickiness is a relaxation of stickiness where only "harmful" variables are taken into account. A set of TGDs which enjoys weak-stickiness is *weakly-sticky*, and the associated class is denoted WS. Observe that $\mathsf{S} \subset \mathsf{WS}$.

A set Σ of TGDs is called *acyclic* if its predicate graph is acyclic, and the underlying class is denoted A. In fact, an acyclic set of TGDs can be seen as a nonrecursive set of TGDs. Σ is called *weakly-acyclic* if its dependency graph enjoys a certain acyclicity condition, which actually guarantees the existence of a finite canonical model; the associated class is denoted WA. Clearly, $\mathsf{A} \subset \mathsf{WA}$.

Another key fragment of TGDs, which deserves our attention, are the so-called *full* TGDs, i.e., TGDs without existentially quantified variables, and the corresponding class is denoted F. If we further assume that full TGDs enjoy linearity, guardedness, stickiness, or acyclicity, then we obtain the classes LF, GF, SF, and AF, respectively.

	Comb.	ba-comb.	fp-comb.
$\mathsf{L(F)}_\perp$, $\mathsf{AF}_\perp$	PSPACE	Π_2^p	Π_2^p
$\mathsf{G}_\perp$	2EXP	EXP	Π_2^p
$\mathsf{WG}_\perp$	2EXP	EXP	EXP
$\mathsf{F}_\perp$, $\mathsf{GF}_\perp$, $\mathsf{S(F)}_\perp$	EXP	Π_2^p	Π_2^p
$\mathsf{A}_\perp$	NEXP - P^{NE}	NEXP - P^{NE}	Π_2^p
$\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$	2EXP	2EXP	Π_2^p

Table 1: Complexity of AR-CQ answering. A single complexity class in a cell refers to a completeness result, while two classes $\mathcal{C}_1$-$\mathcal{C}_2$ refer to $\mathcal{C}_1$-hardness and $\mathcal{C}_2$-membership.

Complexity Results. The (*ba*-/*fp*-)combined complexity of AR-CQ answering under the classes of TGDs introduced above, enriched with NCs, is given in Table 1. Observe that for the cases where classical CQ answering is already very complex, namely, PSPACE and above, dealing with inconsistency comes for free; for the complexity of classical CQ answering see Tables 2 and 3. Of course, this is not true for the class of acyclic TGDs for which we have a complexity gap; let us briefly comment on this gap.

It is well known that P^{NE} and $\mathrm{NEXPTIME}^{\mathrm{NP}}$ are strongly related complexity classes. As shown in (Hemachandra 1989), $\mathrm{NEXPTIME}^{\mathrm{NP}}$ is a delicate class, and if we restrict its oracle access too much, it is weakened to the point of collapsing to P^{NE}. For example, following the notation of (Hemachandra 1989), P^{NE} coincides with $\mathrm{NEXPTIME}^{\mathrm{NP}[poly]_{tree}}$, where only polynomially many NP-oracle calls are allowed throughout the computation tree of the Turing machine. It is evident that the current complexity gap for AR-CQ answering under acyclic TGDs and NCs is not so wide. Nevertheless, the task of bridging this gap is apparently very challenging.

Now, for the cases where the complexity of CQ answering is in NP, dealing with inconsistency comes at a price; it increases to Π_2^p. Interestingly, the transition from CQ to AR-CQ answering follows a certain pattern. In particular, the cases where CQ answering is $\mathcal{C}$-complete, with $\mathcal{C} \supseteq$ PSPACE being a deterministic complexity class, AR-CQ answering remains $\mathcal{C}$-complete, while for the cases where CQ answering is in NP, AR-CQ answering becomes Π_2^p-complete. Notably, a systematic way for transferring results from CQ to AR-CQ answering under arbitrary TGDs and NCs can be established. This will be the subject of the next section.

Before we proceed further, we would like to say that the same (*ba*-)combined complexity for AR-CQ answering under linear TGDs with negative constraints, have been established independently by (Bienvenu and Rosati 2014).

A Generic Complexity Result

We present a generic complexity result which demonstrates the tight connection between classical and consistent CQ answering. This result will automatically provide us with a (nearly) complete picture of the (*ba*-/*fp*-)combined complexity of AR-CQ answering under the main classes of TGDs, enriched with NCs, assuming that the complexity of CQ answering is already known.

Theorem 3 *Assume that CQ answering under a class $\mathbb{C}$ of TGDs is $\mathcal{C}$-complete in (X-)combined complexity, where $X \in \{ba, fp\}$. Then, the (X-)combined complexity of AR-CQ answering under $\mathbb{C}_\perp$ is*

1. *$\mathcal{C}$-complete, if $\mathcal{C} \supseteq$ PSPACE is a deterministic class;*
2. *in P^{NE} and NEXPTIME-hard, if $\mathcal{C} =$ NEXPTIME; and*
3. *Π_2^p-complete, if $\mathcal{C} =$ NP.*

ALGORITHM 1: The algorithm $\mathsf{ARCQAns}$

Input: database D, set $\Sigma \in \mathbb{C}_\perp$, CQ q
Output: $accept$ if $D \cup \Sigma \not\models_{AR} q$; otherwise, $reject$

```
Guess an instance D' ⊆ D;
if there exists ν ∈ Σ_N such that D' ∪ Σ_T ⊨ q_ν then
  | return reject
end
foreach a ∈ D \ D' do
  | if there is no ν ∈ Σ_N such that D' ∪ {a} ∪ Σ_T ⊨ q_ν
  | then
  |   | return reject
  | end
end
if D' ∪ Σ ⊭ q then
  | return accept
else
  | return reject
end
```

In what follows, we establish the above key result.

Upper Bounds. Fix a database D, a set $\Sigma \in \mathbb{C}_\perp$ of TGDs and NCs, and a CQ q. It is easy to see that the problem of deciding whether $D \cup \Sigma \not\models_{AR} q$ can be solved via the algorithm $\mathsf{ARCQAns}$. In the definition of $\mathsf{ARCQAns}$, Σ_T (resp., Σ_N) denotes the set of TGDs (resp., NCs) occurring in Σ. Moreover, given a NC ν of the form $\varphi(\mathbf{X}) \rightarrow \perp$, by q_ν we refer to the CQ $\exists \mathbf{X}\, \varphi(\mathbf{X})$. The first if-then statement of $\mathsf{ARCQAns}$ checks whether $mods(D', \Sigma) \neq \varnothing$, while the foreach-do statement verifies that D' is a $\subseteq$-maximal consistent subset of D. In other words, the above two statements verify that $D' \in drep(D, \Sigma)$. Finally, the last if-then-else statement checks whether D' is a counterexample for the given query q. The next technical lemma is established by analyzing the complexity of $\mathsf{ARCQAns}$.

Lemma 4 *The complement of AR-CQ answering under $\mathbb{C}_\perp$ is in $\mathrm{NP}^{\mathcal{C}}$ in (X-)combined complexity, where $X \in \{ba, fp\}$.*

Having the above lemma in place, we can show the upper bounds in Theorem 3: (1) If $\mathcal{C} \supseteq$ PSPACE is a deterministic class, then, by Lemma 4, AR-CQ answering under $\mathbb{C}_\perp$ is in $\mathcal{C}$ since $\mathrm{NP}^{\mathcal{C}}$ coincides with $\mathcal{C}$ and co$\mathcal{C} = \mathcal{C}$; (2) If $\mathcal{C} =$ NEXPTIME, then Lemma 4 implies a $\mathrm{coNP}^{\mathrm{NEXPTIME}}$ upper bound. The complexity class $\mathrm{NP}^{\mathrm{NEXPTIME}}$ lies at a higher level of the strong exponential hierarchy. However, we know by the work (Hemachandra 1989) that the strong exponential hierarchy collapses to its Δ_2 level, which implies that $\mathrm{NP}^{\mathrm{NEXPTIME}} = \mathrm{P}^{\mathrm{NE}}$, and thus we obtain a $\mathrm{coP}^{\mathrm{NE}}$ upper bound. Observe that the class P^{NE} is a deterministic one, since the oracle machines in terms of which it is defined are deterministic, and therefore $\mathrm{coP}^{\mathrm{NE}} = \mathrm{P}^{\mathrm{NE}}$. Consequently, AR-CQ answering under $\mathbb{C}_\perp$ is in P^{NE}; (3) Finally, if $\mathcal{C} =$ NP, then we get a Π_2^p upper bound, since $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^p$ and co$\Sigma_2^p = \Pi_2^p$.

Lower Bounds. We now proceed with the lower bounds. Clearly, the $\mathcal{C}$-hardness results when $\mathcal{C}$ is NEXPTIME, or a deterministic class above PSPACE, follow immediately, since CQ answering under $\mathbb{C}$ is a special case of AR-CQ answering under $\mathbb{C}_{\perp}$. The non-trivial result is the Π_2^p-hardness. Interestingly, a strong lower bound, which implies all the necessary Π_2^p-hardness results, can be established by a reduction from the validity problem of 2QBF formulas:

Proposition 5 *AR-CQ answering under a single negative constraint $\varphi(\mathbf{X}) \rightarrow \perp$, where φ consists of two atoms and it uses a single ternary predicate, while the database and the CQ use only binary and ternary predicates, is Π_2^p-hard.*

Proof. We proceed by a reduction from the validity problem of 2QBF formulas. Let φ be a 2QBF formula of the form $\forall X_1 \ldots \forall X_n \exists Y_1 \ldots \exists Y_m\, \psi$, where $\psi = C_1 \wedge \ldots \wedge C_k$ is a 3CNF formula such that C_i is a clause of the form $(\ell_i^1 \vee \ell_i^2 \vee \ell_i^3)$. Let $var(\ell_i^j)$ be the variable of ℓ_i^j. In the sequel, let $\mathbf{T} = \{X_1, \ldots, X_n, Y_1, \ldots, Y_m\}$, i.e., the variables in φ. We proceed with the construction of D, Σ and q such that $D \cup \Sigma \models_{AR} q$ iff φ is satisfiable.

The Database D. Intuitively speaking, in D we store, for each clause C_i, all the valuations which make C_i true. A valuation for $\mathbf{T}$ is a function $f : \mathbf{T} \rightarrow \{0, 1\}$. Given a literal $\ell = Z$ (resp., $\ell = \neg Z$), $f(\ell) = f(Z)$ (resp., $f(\ell) = \neg f(Z)$). A valuation f satisfies a clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$ if $(f(\ell_1) \vee f(\ell_2) \vee f(\ell_3)) = 1$. For a clause C, let F_C be the set of all valuations for $\mathbf{T}$ which make C true. The database D is defined as follows: $\{p_i^j(c_i, f(var(\ell_i^j)))\}_{i \in [k], f \in F_{C_i}, j \in [3]} \cup \{s(0, 1, d_i), s(1, 0, d_i)\}_{i \in [n]}$; the purpose of the auxiliary s-atoms will be clarified soon.

The Set Σ. This set contains the single negative constraint

$$s(X, Y, Z), s(W, X, Z) \;\rightarrow\; \perp.$$

Note that the above constraint uses only one 3-ary predicate.

The CQ q. Finally, the conjunctive query q is defined as follows: $\wedge_{i=1}^{k} \wedge_{j=1}^{3} p_i^j(Z_i, var(\ell_i^j)) \;\wedge\; \wedge_{i=1}^{n} s(X_i, W_i, d_i)$, where all the variables are existentially quantified variables. This completes our construction.

It is not difficult to show that indeed $D \cup \Sigma \models_{AR} q$ iff φ is satisfiable. Roughly speaking, the single negative constraint forces us to consider all the possible subsets of D which can be obtained by removing either the atom $s(0, 1, d_i)$ or the atom $s(1, 0, d_i)$, for each $i \in [n]$; this holds since there are no TGDs in Σ. Each such subset D' of D corresponds to a possible assignment μ of values to the universally quantified variables of φ. Finally, by evaluating the query q over D' in fact we ask whether there exists a valuation which is compatible with μ that makes φ true. $\square$

By Proposition 5, for every class $\mathbb{C}$ of TGDs, AR-CQ answering under $\mathbb{C}_{\perp}$ is Π_2^p-hard in *fp*-combined complexity, which in turn implies the Π_2^p-hardness results in Theorem 3.

From Classical to AR-CQ Answering

In this section, we focus on the non-full classes of TGDs introduced above, and we show that the complexity of AR-CQ answering can be obtained in a uniform way by exploiting our generic complexity theorem. To this aim, it suffices to identify the complexity of (classical) CQ answering, which is summarized in Table 2. Clearly, by combining Table 2 with Theorem 3, we get that:

	Comb.	ba**-comb.**	fp**-comb.**
L	PSPACE	NP	NP
G	2EXP	EXP	NP
WG	2EXP	EXP	EXP
S	EXP	NP*	NP
A	NEXP*	NEXP*	NP
WS, WA	2EXP	2EXP	NP

Table 2: Complexity of CQ answering. All results are completeness results. The symbol $\star$ refers to novel results.

Theorem 6 *The (X-)combined complexity of AR-CQ answering under $\mathbb{C}_{\perp}$, where $X \in \{ba, fp\}$ and $\mathbb{C} \in \{$L, (W)G, (W)S, (W)A$\}$, is as shown in Table 1.*

Let us now focus on classical query answering.

Classical CQ Answering

Although for several cases the complexity of CQ answering is already known, there are some interesting cases that are still open. Surprisingly, the (*ba*-)combined complexity of CQ answering under acyclic TGDs has to our knowledge never been explicitly studied. Furthermore, the *ba*-combined complexity for sticky sets of TGDs is not known. In what follows, we close the above open problems.

Our main tool is the *chase procedure*, which works on an instance through the *chase rule*. Answering a CQ against a database D and a set Σ of TGDs is equivalent to evaluating the same query over the chase-expansion of D according to Σ, denoted $chase(D, \Sigma)$; this expansion can be obtained via the chase procedure. Informally, the chase adds new atoms to D (possibly involving null values) until the final result satisfies Σ; for the details, see, e.g., (Calì, Gottlob, and Pieris 2012). We now proceed with our open questions.

Stickiness. We start with the *ba*-combined complexity for sticky sets of TGDs, and show that it is NP-complete. The key property of stickiness that we are going to exploit is the so-called *polynomial witness property (PWP)* (Gottlob and Schwentick 2012). Roughly, a class of TGDs $\mathbb{C}$ enjoys the PWP if, whenever a CQ q is entailed by a database D and a set $\Sigma \in \mathbb{C}$, then q is already satisfied by a finite part of $chase(D, \Sigma)$, the witness, of polynomial size in q and Σ. In fact, there exists a polynomial f such that the witness can be constructed after $f(q, \Sigma)$ applications of the chase rule.

It is not difficult to show that the PWP implies an NP upper bound for CQ answering. One can apply chase steps non-deterministically until the CQ q is entailed, in which case the algorithm accepts; if after $f(q, \Sigma)$ chase steps q is not entailed, then our algorithm rejects. Clearly, the above procedure runs in polynomial time, and the claim follows. It has been recently shown that sticky sets of TGDs enjoy the PWP when the arity is bounded by an integer constant (Gottlob, Manna, and Pieris 2014), and the desired upper bound follows. The NP-hardness is inherited from CQ containment (which is LOGSPACE-equivalent to CQ answering) without constraints (Chandra and Merlin 1977). Thus, we get that:

	Comb.	*ba*-comb.	*fp*-comb.
F	EXPTIME	NP	NP
LF	PSPACE*	NP	NP
GF	EXPTIME*	NP	NP
SF	EXPTIME$^\diamondsuit$	NP	NP
AF	PSPACE	NP	NP

Table 3: Complexity of CQ answering. All results are completeness results. The symbol $\star$ refers to novel results, while the symbol $\diamondsuit$ to results that are derivable from existing ones.

Theorem 7 *CQ answering under* S *is* NP*-complete in ba-combined complexity.*

Acyclicity. Let us now proceed with acyclic TGDs. Since acyclicity guarantees the termination of the chase procedure, an obvious query answering algorithm is to explicitly construct the chase, and then evaluate the given query over the obtained (finite) instance. However, this naive approach does not provide us with an optimal upper bound, since the chase procedure under acyclic TGDs, in general, terminates after double-exponentially many steps. Luckily, the desired NEXPTIME upper bound can be obtained by exploiting a result in (Dantsin and Voronkov 1997), where the complexity of nonrecursive logic programs with complex values is investigated. The problem of deciding whether a fact is entailed by a (positive) nonrecursive logic program is in NEXPTIME. Our problem can be effectively reduced, via skolemization, to the above problem. For the lower bound, we use a classical NEXPTIME-hard problem, namely TILING (Fürer 1983), and we show that CQ answering under acyclic TGDs is NEXPTIME-hard, even for atomic queries and predicates of arity at most six. From the above discussion, we get that:

Theorem 8 *CQ answering under* A *is* NEXPTIME*-complete in (ba-)combined complexity.*

Full Dependencies: A Closer Look

We now focus on the key class of full TGDs. Notice that important database dependencies, e.g., join and multivalued dependencies, are expressible via full TGDs (Abiteboul, Hull, and Vianu 1995), and also a plain Datalog program can be seen as a set of full TGDs. It is evident that full TGDs are of great importance and they deserve further investigation. We would like to better understand whether the complexity of AR-CQ answering under full TGDs, enriched with NCs, is affected or not if we further assume that the given set of TGDs enjoys linearity, guardedness, stickiness, or acyclicity. Observe that the existential-free version of the "weakly" classes of TGDs considered in this work coincide with full TGDs; this is why we consider only the classes LF, GF, SF and AF. We show that:

Theorem 9 *The (X-)combined complexity of AR-CQ answering under* $\mathbb{C}_\perp$*, where* $X \in \{ba, fp\}$ *and* $\mathbb{C} \in \{\mathsf{F}, \mathsf{LF}, \mathsf{GF}, \mathsf{SF}, \mathsf{AF}\}$*, is as shown in Table 1.*

In the sequel, we establish the above theorem. To this aim, it suffices to pinpoint the complexity of classical CQ answering under the relevant classes of TGDs, which is shown in Table 3, and then apply our generic complexity result.

Classical CQ Answering

It is easy to show that CQ answering under full TGDs is EXPTIME-complete in combined complexity and NP-complete in *ba-/fp*-combined complexity. The upper bounds follow from the fact that a universal model can be constructed in exponential (resp., nondeterministic polynomial) time via the chase procedure. The EXPTIME-hardness is inherited from plain Datalog (Dantsin et al. 2001), while the NP-hardness is inherited from CQ containment without constraints.

It is straightforward to see that the *ba-/fp*-combined complexity of CQ answering under the relevant fragments of full TGDs remain NP-complete. In what follows, more details about the combined complexity of our problem are given.

Linearity. A PSPACE upper bound for linear full TGDs is obtained from the fact that CQ answering under linear TGDs is feasible in polynomial space. Regarding the lower bound, we can easily simulate the behavior of a deterministic polynomial space machine, and get the following result.

Proposition 10 *CQ answering under* LF *is* PSPACE*-hard in combined complexity, even for atomic CQs.*

Notice that the above result is implicit in (Gottlob and Papadimitriou 2003; Calì, Gottlob, and Pieris 2012). However, the existing proofs heavily use constants in the set of TGDs and the query, while our construction employs constant-free TGDs and a query consisting of a single propositional atom.

Guardedness. Unfortunately, guardedness has no positive impact on our problem. In fact, we show a stronger result, i.e., even if the TGDs are *strongly-guarded*, i.e., each body-atom contains all the body-variables, our problem remains EXPTIME-hard. This is a surprising result as one expects that such a strong condition would force the TGDs to behave like linear TGDs, and thus reduce the complexity to PSPACE. The key idea underlying our proof is to simulate a Datalog program, over the domain $\{0, 1\}$ (Dantsin et al. 2001), using a strongly-guarded full set of TGDs. Notice that the fact inference problem for Datalog programs over the domain $\{0, 1\}$ is already EXPTIME-hard (Dantsin et al. 2001).

Proposition 11 *CQ answering under* GF *is* EXPTIME*-hard in combined complexity, even for strongly-guarded TGDs and atomic CQs.*

Proof. Consider a database D, where $dom(D) = \{0, 1\}$, a Datalog program P, and a ground atom $\underline{a}$. We are going to construct a database $\hat{D}$, a set Σ of strongly-guarded TGDs, and an atomic CQ q such that $\underline{a} \in P(D)$ iff $\hat{D} \cup \Sigma \models q$.

The Database $\hat{D}$. The database $\hat{D}$ is obtained from D by simply extending the arity of each predicate occurring in D by two, and adding to the first two positions the constants 0 and 1. Formally, $\hat{D} = \{\bar{p}(0, 1, \mathbf{t}) \mid p(\mathbf{t}) \in D\}$.

The Set Σ. First, we transform the program P into a set Σ_1 of strongly-guarded TGDs: for each $\rho \in P$ of the form $p_1(\mathbf{X}_1), \ldots, p_n(\mathbf{X}_n) \rightarrow p_0(\mathbf{X}_0)$, we add in Σ_1 the strongly-guarded TGD $\bar{p}_1^\rho(Z, O, \mathbf{X}_1, Y_1, \ldots, Y_m), \ldots, \bar{p}_n^\rho(Z, O, \mathbf{X}_n, Y_1, \ldots, Y_m) \rightarrow \bar{p}_0(Z, O, \mathbf{X}_0)$, where $Y_1, \ldots, Y_m$ are the variables occurring in ρ. Let us say that the variables Z and O are auxiliary variables that will allow us to have access to the constants 0 and 1, respectively, without explicitly mentioning them in the body of the TGDs.

Now, we have to guarantee that all the necessary atoms with a predicate of the form $\bar{p}^{\rho}$ will eventually be inferred. To this aim, we are going to construct a set Σ_2 of linear TGDs of polynomial size. For each n-ary predicate p occurring in P, and for each rule $\rho \in P$ such that p occurs in the body of ρ, assuming that ρ contains m variables, we add in Σ_2 the linear TGD $\bar{p}(Z, O, \mathbf{X}) \rightarrow \bar{p}^{\rho}(Z, O, \mathbf{X}, Z, \ldots, Z)$, where we have m occurrences of Z, and the linear TGDs

$$\{\bar{p}^{\rho}(Z, O, \mathbf{X}, \mathbf{Y}^{m-i}, Z, O^{i-1}) \rightarrow \bar{p}^{\rho}(Z, O, \mathbf{X}, \mathbf{Y}^{m-i}, O, Z^{i-1})\}_{i \in [m]},$$

where $\mathbf{X}$ is the tuple of variables $X_1, \ldots, X_n$, $\mathbf{Y}^k$ is the tuple of variables $Y_1, \ldots, Y_k$ and V^k is the k-tuple $V, \ldots, V$, with $k \geqslant 1$ and $V \in \{Z, O\}$. Let $\Sigma = \Sigma_1 \cup \Sigma_2$.

The Query q. Assuming that $\underline{a} = p(\mathbf{t})$, the atomic CQ q is simply defined as $\bar{p}(0, 1, \mathbf{t})$. □

Stickiness. Unluckily, also stickiness has no positive impact on the combined complexity of CQ answering under full TGDs. This fact is easily derivable from (Calì, Gottlob, and Pieris 2012), where it is shown that every Datalog program over the domain $\{0, 1\}$ can be converted in polynomial time into a set of full TGDs which enjoy the following property: each variable in the body occurs also in the head atom, which in turn implies that stickiness is trivially satisfied.

Acyclicity. Finally, acyclicity reduces the combined complexity of our problem to PSPACE. This follows from the fact that nonrecursive Datalog is PSPACE-complete (Vorobyov and Voronkov 1998).

Having the above complexity results in place for classical CQ answering, it is easy to verify that our generic theorem implies the results for AR-CQ answering shown in Table 1.

Conclusions

In this work, we performed an in-depth complexity analysis of the problem of consistent query answering under the main decidable classes of TGDs, focussing on the AR semantics. Notably, a generic complexity result has been established, which allowed us to obtain a (nearly) complete picture of the complexity of our problem in a systematic and uniform way. Regarding future work, apart from bridging the complexity gap for acyclic TGDs, we intend to perform a similar complexity analysis for other important semantics such as the IAR semantics, that is, a sound approximation of the AR semantics (Lembo et al. 2010).

Acknowledgements. This work has received funding from the EPSRC grant EP/J008346/1. M.V. Martinez and G.I. Simari are partially supported by Proyecto PIP-CONICET 112-201101-01000. A. Pieris is also supported by the Austrian Science Fund (FWF): P25207-N23 and Y698.

References

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

Arenas, M.; Bertossi, L. E.; and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *PODS*, 68–79.

Bienvenu, M., and Rosati, R. 2013. Tractable approximations of consistent query answering for robust ontology-based data access. In *IJCAI*.

Bienvenu, M., and Rosati, R. 2014. Personal communication.

Bienvenu, M. 2012. On the complexity of consistent query answering in the presence of simple ontologies. In *AAAI*.

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

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48:115–174.

Calì, A.; Gottlob, G.; and Pieris, A. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193:87–128.

Chandra, A. K., and Merlin, P. M. 1977. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, 77–90.

Dantsin, E., and Voronkov, A. 1997. Complexity of query answering in logic databases with complex values. In *LFCS*, 56–66.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv.* 33(3):374–425.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci.* 336(1):89–124.

Fürer, M. 1983. The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems). In *Logic and Machines*, 312–319.

Gottlob, G., and Papadimitriou, C. H. 2003. On the complexity of single-rule Datalog queries. *Inf. Comput.* 183(1):104–122.

Gottlob, G., and Schwentick, T. 2012. Rewriting ontological queries into small nonrecursive Datalog programs. In *KR*.

Gottlob, G.; Manna, M.; and Pieris, A. 2014. Polynomial combined rewritings for existential rules. In *KR*.

Hemachandra, L. A. 1989. The strong exponential hierarchy collapses. *J. Comput. Syst. Sci.* 39(3):299–322.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2010. Inconsistency-tolerant semantics for description logics. In *RR*, 103–117.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2012. Inconsistency handling in Datalog+/- ontologies. In *ECAI*, 558–563.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013. Complexity of inconsistency-tolerant query answering in Datalog+/-. In *ODBASE*, 488–500.

Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. Data Semantics* 10:133–173.

Rosati, R. 2011. On the complexity of dealing with inconsistency in description logic ontologies. In *IJCAI*, 1057–1062.

Vardi, M. Y. 1982. The complexity of relational query languages (extended abstract). In *STOC*, 137–146.

Vorobyov, S. G., and Voronkov, A. 1998. Complexity of nonrecursive logic programs with complex values. In *PODS*, 244–253.