Complexity of Inconsistency-Tolerant Query Answering in Datalog+/–

Thomas Lukasiewicz, María Vanina Martínez, and Gerardo I. Simari

Department of Computer Science, University of Oxford, UK
{thomas.lukasiewicz,vanina.martinez,gerardo.simari}@cs.ox.ac.uk

Abstract. The study of inconsistency-tolerant semantics for query answering in ontological languages has recently gained much attention. In this work, we consider three inconsistency-tolerant semantics recently proposed in the literature, namely: consistent query answering, the intersection (also called IAR) semantics, and the intersection of closed repairs (ICR) semantics. We study the data complexity of conjunctive query answering under these semantics for a wide set of tractable fragments of Datalog+/–. The Datalog+/– family of ontology languages covers several important description logics (DLs), bridging the gap in expressive power between database query languages and DLs as ontology languages, and extending the well-known Datalog language in order to embed DLs. Its properties of decidability of query answering and of tractability of query answering in the (data) complexity make Datalog+/– very useful in modeling real-world applications in which inconsistency-tolerant reasoning is essential.

1 Introduction

The issue of inconsistency handling in AI in general and in knowledge representation (KR) in particular has been widely studied in the last three decades. In the database community, the field of *database repairing* and *consistent query answering* (CQA) has gained much attention since the work of [1], which provided a model-theoretic construct of a database *repair*. The most widely accepted semantics for querying a possibly inconsistent database is that of *consistent answers*, which yields the set of tuples (atoms) that appear in the answer to the query over *every* possible repair. CQA enforces consistency at query time as an alternative to enforcing it at the instance level, as conventional data cleaning techniques do. This allows to focus on a smaller portion of the database for which repairs can be computed more easily. Furthermore, techniques have been developed so that it is not necessary to materialize every possible repair. The work of [14] addresses the basic concepts and results of the area of CQA.

More recently, the advent of the Semantic Web, a vision for the future Web, which is strongly based on ontology languages, has led to a resurgence of interest in this area, specially focusing on the development of efficient inconsistency-tolerant reasoning and query answering in DLs and ontology languages. Lately, several works have focused on inconsistency handling for different classes of DLs, adapting and specializing general techniques previously considered for traditional logics [25, 18, 26, 24]. In [20], the adaptation of CQA for *DL-Lite* ontologies is studied. The *intersection semantics* (*ICons*) is presented as a sound approximation of consistent answers, which

Fragment of Datalog+/–		*Cons*	*ICons*	*ICR*
guarded	(Theorem 9)	co-NP-complete [22]	co-NP-complete [22]	co-NP-complete
linear	(Theorem 10)	co-NP-complete [22]	FO-rewritable [23]	co-NP-complete
multi-linear	(Theorem 10)	co-NP-complete	FO-rewritable	co-NP-complete
frontier-one	(Theorem 11)	co-NP-complete	co-NP-complete	co-NP-complete
frontier-guarded	(Theorem 12)	co-NP-complete	co-NP-complete	co-NP-complete
joint-acyclic	(Theorem 14)	co-NP-complete	co-NP-complete	co-NP-complete
weakly-acyclic	(Theorem 13)	co-NP-complete	co-NP-complete	co-NP-complete
sticky	(Theorem 15)	co-NP-complete	FO-rewritable	co-NP-complete
sticky-join	(Theorem 16)	co-NP-complete	FO-rewritable	co-NP-complete
weakly-sticky	(Theorem 17)	co-NP-complete	co-NP-complete	co-NP-complete
weakly-sticky-join	(Theorem 17)	co-NP-complete	co-NP-complete	co-NP-complete

Table 1. Summary of this paper's data complexity results for BCQ answering in 11 fragments of Datalog+/– under the three different inconsistency-tolerant semantics *Cons*, *ICons*, and *ICR*.

for the *DL-Lite* family is easier to compute, as well as the *closed ABox* version, which considers the closure of the set of assertional axioms (ABox, or extensional database) by the terminological axioms (TBox, or intensional database). Later, in [21], the authors explore first-order (FO-) rewritability of *DL-Lite* ontologies under $ICons$. The data and combined complexity of the semantics were studied in [27] for a wide spectrum of DLs. Despite the fact that computing consistent answers is an inherently hard problem ([20] shows co-NP completeness even for ground atomic queries in *DL-Lite*), some works identify cases for simple ontologies (within the *DL-Lite* family) for which tractable results can be obtained [6, 8]. In [27], the complexity for query answering under inconsistency-tolerant semantics is provided for a wide spectrum of DLs, ranging from tractable ones ($\mathcal{EL}$) to very expressive ones ($\mathcal{SHIQ}$). In [22], consistent query answering and query answering under the $ICons$ semantics for linear and guarded Datalog+/– ontologies are studied. An alternative inconsistency-tolerant semantics, called the k-lazy semantics, is also proposed. In [23], FO-rewritability is shown for query answering under the $ICons$ semantics for linear Datalog+/–.

In this work, we analyze the (data) complexity of query answering of Boolean conjunctive queries for a set of fragments of Datalog+/– under the three different recent semantics *consistent query answering* (CQA) [20], the *intersection semantics* ($ICons$) [20], and the *intersection of closed repairs* (ICR) [8]. We focus on the Datalog+/– family of ontology languages [10], particularly on several fragments of Datalog+/– that guarantee termination of query answering procedures in polynomial time in the data complexity or FO-rewritability. Datalog+/– enables a modular rule-based style of knowledge representation, and it represents syntactical fragments of first-order logic so that answering a BCQ Q under a set Σ_T of Datalog+/– rules for an input database D is equivalent to the classical entailment check $D \cup \Sigma_T \models Q$. The following example illustrates how description logic (DL) knowledge bases are expressed in Datalog+/–.

Example 1 A DL knowledge base consists of a *TBox* and an *ABox*. For example, the knowledge that every conference paper is an article and that every scientist is the author of at least one paper can be expressed by two *TBox* axioms: $\mathit{ConfPaper} \sqsubseteq \mathit{Article}$ and $\mathit{Scientist} \sqsubseteq \exists isAuthor$. The fact that Mark is a scientist can be expressed by the *ABox* axiom $\mathit{Scientist}(\mathit{Mark})$. The *TBox* can be encoded in Datalog+/– rules as follows: $\mathit{ConfPaper}(X) \rightarrow \mathit{Article}(X)$ and $\mathit{Scientist}(X) \rightarrow \exists Y\, isAuthor(X, Y)$. The *ABox* is encoded by an identical set of facts in the database. The *TBox* axiom $\mathit{ConfPaper} \sqsubseteq \neg \mathit{JournalPaper}$ saying that conference papers are not journal papers, can be expressed by the following constraint $\mathit{ConfPaper}(X) \wedge \mathit{JournalPaper}(X) \rightarrow \bot$. Finally, a simple Boolean conjunctive query (BCQ) asking if Mark authors a paper is $\exists X\, isAuthor(\mathit{Mark}, X)$.

Datalog+/–'s properties of decidability and data tractability of query answering, along with its expressive power, make it very useful in many real-world application areas, such as ontology querying, Web data extraction, data exchange, ontology-based data access, and data integration. Studying the complexity of consistent query answering for several semantics and languages can provide insights for finding tractable special cases, average cases, and approximation algorithms for each of these combinations, or even inspirations for inconsistency handling semantics with better computational properties. Table 1 summarizes this paper's data complexity results for BCQ answering in 11 fragments of Datalog+/– under the three different inconsistency-tolerant semantics.

This paper is organized as follows. In Section 2, we recall the basic notions of Datalog+/– ontologies. Section 3 provides the definitions for the three inconsistency-tolerant semantics and some results about their relationships. In Section 4, we provide a complete data complexity analysis for 11 fragments of Datalog+/– and the three inconsistency-tolerant semantics (proofs of all results are given in the extended version of this paper). Section 5 summarizes the main results and gives an outlook on future work.

2 Preliminaries on Datalog+/–

In this Section, we briefly recall the basics on Datalog+/– [10], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple- and equality-generating dependencies (TGDs and EGDs, respectively), negative constraints, and Datalog+/– ontologies.

Databases and Queries. We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries, dependencies, and constraints). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geqslant 0$. We assume a *relational schema* $\mathcal{R}$, which is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, null, or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1, ..., t_n)$, where P is an n-ary predicate, and $t_1, ..., t_n$ are terms.

A *database (instance)* D for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. A *conjunctive query (CQ)* over

$\mathcal{R}$ has the form $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms (possibly equalities, but not inequalities) with the variables $\mathbf{X}$ and $\mathbf{Y}$, and possibly constants, but without nulls. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$, often written as the set of all its atoms, without quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu\colon \Delta \cup \Delta_N \cup \mathcal{V} \rightarrow \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $Q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu\colon \mathbf{X} \cup \mathbf{Y} \rightarrow \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$ (without nulls), called the *body* and the *head* of σ, denoted $\mathit{body}(\sigma)$ and $\mathit{head}(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D, there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. All sets of TGDs are finite here. Since TGDs can be reduced to TGDs with only single atoms in their heads, in the sequel, every TGD has w.l.o.g. a single atom in its head.

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $\mathit{mods}(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $\mathit{ans}(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in \mathit{mods}(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $\mathit{ans}(Q, D, \Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [5], even when the schema and TGDs are fixed [9]. Both problems of CQ and BCQ evaluation under TGDs are LOGSPACE-equivalent [17, 16]. Moreover, the query output tuple (QOT) problem (as a decision version of CQ evaluation) and BCQ evaluation are AC$_0$-reducible to each other. Henceforth, we focus only on BCQ evaluation, and any complexity results carry over to the other problems.

Negative constraints (or constraints) υ are first-order formulas $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ (called the *body* of υ) is a conjunction of atoms (without nulls). Under the standard semantics of query answering of BCQs in Datalog+/– with TGDs for each constraint $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, we only have to check that the BCQ $\Phi(\mathbf{X})$ evaluates to false in D under Σ; if one of these checks fails, then the answer to the original BCQ Q is false, otherwise the constraints can simply be ignored when answering the BCQ Q.

Equality-generating dependencies (or *EGDs*) σ, are first-order formulas $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of σ, denoted $\mathit{body}(\sigma)$, is a (without nulls) conjunction of atoms, and X_i and X_j are variables from $\mathbf{X}$. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. In this work, we assume *non-conflicting* [10] EGDs; this guarantees that EGDs have no impact on the chase with respect to query answering. We usually omit the universal quantifiers in TGDs, negative constraints, and EGDs, and assume that all sets of dependencies and/or constraints are finite.

Datalog+/– Ontologies. A *Datalog+/– ontology* $KB = (D, \Sigma)$, where $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$, consists of a finite database D, a set of TGDs Σ_T, a set of non-conflicting EGDs Σ_E, and a set of negative constraints Σ_{NC}.

3 Inconsistency-Tolerant Query Answering

Different works on inconsistency handling in description logics (DLs) allow for inconsistency to occur for different reasons. Depending on the expressive power of the underlying formalism, some works allow for both terminological axioms (TBox) and assertional axioms (ABox) to be inconsistent. In this work, following the idea from database theory in which formulas in Σ are interpreted as integrity constraints expressing the semantics of the data contained in D, we assume that Σ is itself consistent, and inconsistencies can only arise when D and Σ are considered together.

Definition 2 (Consistency) A Datalog+/– ontology $KB = (D, \Sigma)$ is *consistent* iff $mods(D, \Sigma) \neq \emptyset$.

We recall now the notion of *data repairs* of a Datalog+/– ontology from [22]; intuitively these are minimal (in the set-inclusion sense) consistent subsets of D.

Definition 3 (Data Repair) A *data repair* of a Datalog+/– ontology $KB = (D, \Sigma)$ is a set D' such that (i) $D' \subseteq D$, (ii) $mods(D', \Sigma) \neq \emptyset$, and (iii) there is no other set $D'' \subseteq D$ such that $D' \subset D''$ and $mods(D'', \Sigma) \neq \emptyset$. We denote by $DRep(KB)$ the set of all data repairs for KB.

Data repairs play a central role in *consistent answers* for a query to an ontology, which are intuitively the answers relative to each ontology built from a data repair.

Definition 4 (Consistent Answers) Let $KB = (D, \Sigma)$ be a Datalog+/– ontology, and Q be a BCQ. Then, *Yes* is a *consistent answer* for Q to KB, denoted $KB \models_{Cons} Q$, iff it is an answer for Q to each $KB' = (D', \Sigma)$ with $D' \in DRep(KB)$.

The problem of determining if an answer is a consistent answer has been shown to be hard even for simple ontological languages such as *DL-Lite*$_{Core}$ [20] even for more restrictive sublanguages [6]. To avoid intractability, approximations to consistent answers have been developed. The first we consider only takes into account the atoms that are in the *intersection* of all data repairs; it was presented as the *IAR* semantics for *DL-Lite* in [20]. The *intersection semantics* (or *ICons* semantics), as called in a generalization for Datalog+/– ontologies in [22], yields a unique way of repairing inconsistency, and the consistent answers are intuitively the answers that can be obtained from that unique set. This semantics is also called *cautious query answering* in [6].

Definition 5 (Intersection Semantics) Let $KB = (D, \Sigma)$ be a Datalog+/– ontology, and Q be a BCQ. Then, *Yes* is a *consistent answer* for Q to KB *under the intersection semantics*, denoted $KB \models_{ICons} Q$, iff it is an answer for Q to $KB_I = (D_I, \Sigma)$, where $D_I = \bigcap \{D' \mid D' \in DRep(KB)\}$.

A finer approximation to consistent answers is proposed by [7] which, adapted to Datalog+/– , corresponds to closing repairs with respect to Σ_T before intersecting them. In the next definition, $Cn(D', \Sigma_T)$ denotes the set of all ground atoms entailed by a set of atoms D' and a set of TGDs Σ_T. Note that even though the set of constants may be infinite, we restrict the closure of D with respect to Σ_T to the active domain of D (i.e., the constants that appear in the database instance).

Definition 6 (Intersection of Closed Repairs) Let $KB = (D, \Sigma)$ be a Datalog+/– ontology, and Q be a BCQ. Then, *Yes* is a *consistent answer* for Q to KB *under the ICR semantics*, denoted $KB \models_{ICR} Q$, iff it is an answer for Q to $KB_I = (D_I, \Sigma)$, where $D_I = \bigcap \{ Cn(D', \Sigma_T) \mid D' \in DRep(KB)\}$.

Soundness for this semantics is shown in [7]; that is, for a given BCQ Q and *DL-Lite*$_{core}$ ontology KB, if $KB \models_{ICR} Q$ then $KB \models_{Cons} Q$. The following proposition shows the equivalence between the two semantics whenever Q is an atomic ground BCQ. Note that the result holds for arbitrary Datalog+/– ontologies.

Proposition 7 *Let KB be a Datalog+/– ontology, and Q be a ground atomic BCQ. We have that $KB \models_{Cons} Q$ iff $KB \models_{ICR} Q$.*

Proof. We want to show that $KB \models_{ICR} Q$ iff $KB \models_{Cons} Q$, that is the same as showing that $(\bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)) \models Q$ iff for every $r \in DRep(KB)$, $(r, \Sigma_T) \models Q$.

$(\Rightarrow)$ If $(\bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)) \models Q$ then for every data repair r, either $Q \in \bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)$ or there exists a set $B \subseteq \bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)$ such that $(B, \Sigma_T) \models Q$. This means that $(r, \Sigma_T) \models Q$ for every $r \in DRep(KB)$.

$(\Leftarrow)$ If for every $r \in DRep(KB)$, we have that $(r, \Sigma_T) \models Q$ then, for every r, $Cn(r, \Sigma_T) \models Q$ and therefore $Q \in Cn(r, \Sigma_T)$, then $Q \in \bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)$ and thus $KB \models_{ICR} Q$. □

In the next section we analyze the data complexity of query answering under the three different inconsistency-tolerant semantics for a variety of fragments of Datalog+/– for which classical query answering of BCQs has already been shown to be tractable. The following proposition helps us prove some of the complexity results; in showing the data complexity in some of the cases; it states that the property of FO-rewritability for classical query answering of BCQs still holds under the intersection semantics.

Proposition 8 *Let KB be a F-Datalog+/– ontology. If query answering is FO-rewritable for the fragment F, then so is query answering under the $ICons$ semantics.*

Proof (sketch). The rewriting algorithm for linear ontologies from [23] can be used to show FO-rewritability of any fragment of Datalog+/– for which query answering is FO-rewritable, since this is the only assumption made in the proofs of the results presented in that paper. □

4 Landscape of Datalog+/– Classes

The problem of entailment is known to be undecidable in the presence of general TGDs [5, 13]. However, in the recent years several decidable and even tractable classes have been defined based on different syntactic properties of the TGDs and abstract properties related to the behavior of reasoning mechanisms [3, 2, 12, 4, 10]. In this Section we recall several Datalog+/– sublanguages and analyze the data complexity of query answering under the different inconsistency-tolerant semantics described in Section 3. Figure 1 shows the summary of the results obtained in this section.

4.1 Guarded Sets of TGDs

We first recall the notion of *guardedness* from [10]. A TGD σ is said to be *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. The leftmost such atom is the *guard atom* (or *guard*) of σ.

Computing the consisten answers and query answering under the $ICons$ semantics for guarded Datalog+/– was shown to be co-NP-complete in [22]. To establish membership in co-NP for consistent query answering, we can show a witness that consists of a $D' \in DRep(KB)$ for which $(D', \Sigma_T) \not\models Q$. Note that D' can be verified to be a repair in polynomial time, since atoms from D can be added in turn and verified to give rise to an inconsistency, as can $(D', \Sigma_T) \not\models Q$ be checked in polynomial time in the guarded case.

Theorem 9 *Let KB be a guarded Datalog+/– ontology, and Q be a BCQ. Deciding if $KB \models_{ICR} Q$ is co-NP-complete.*

Proof (sketch). Membership in co-NP for query answering under the ICR semantics can be shown by guessing and verifying some $D^\star \subseteq Cn(D, \Sigma_T)$ with $(D^\star, \Sigma_T) \not\models Q$, and, for every $\alpha \in Cn(D, \Sigma_T) - D^\star$, some $D'_\alpha \subseteq \mathit{DRep}(KB)$ such that $\alpha \notin Cn(D'_\alpha, \Sigma_T)$. Note that the latter implies that $\bigcap_{D' \in DRep(KB)} Cn(D', \Sigma_T) \subseteq D^\star$. Hence, $(D^\star, \Sigma) \not\models Q$ implies $KB \not\models_{ICR} Q$. The above guessing and verifying can be done in polynomial time in the data complexity in the guarded case. Membership in co-NP for the $ICons$ semantics can be shown analogously (without computing the closure of the repairs).

Finally, co-NP hardness for consistent query answering is shown in [20] even for ground atomic queries; by Proposition 7, this means that it is also hard for *DL-Lite*$_{core}$ under the ICR semantics, and since this language is a subset of guarded Datalog+/– we can conclude that query answering under the ICR semantics is also co-NP hard in the guarded case. □

A TGD *linear* iff it contains only a single atom in its body. Furthermore, sets of multi-linear TGDs correspond to TGDs whose bodies contain only guards. For linear Datalog+/– co-NP completeness for consistent query answering was shown in [22] and FO-rewritability for linear Datalog+/– ontologies under the $ICons$ semantics was shown in [23]. For multi-linear sets of TGDs we have the following results.

Theorem 10 *Let KB be a multi-linear Datalog+/– ontology, and Q be a BCQ. Deciding if (a) $KB \models_{Cons} Q$ and if (b) $KB \models_{ICR} Q$ are both co-NP-complete. Also, (c) query answering for BCQs under $ICons$ is FO-rewritable.*

Proof (sketch). Since linear is a sublanguage of multi-linear Datalog+/– [10], we can easily show that consistent query answering is co-NP-hard for the multi-linear case. Membership in co-NP can be shown analogously to the linear/guarded case. Furthermore, by Proposition 8 we now that query answering under the $ICons$ semantics is also FO-rewritable for the multi-linear case. Finally, following the same argument made for guarded Datalog+/–, by Proposition 7 we conclude that query answering under the ICR semantics is also co-NP hard in both the linear and multi-linear case. □

4.2 Frontier-Restricted Sets of TGDs

The frontier of a TGD σ is the set of variables that occur both in the head and body of σ. A TGD σ is *frontier-one* if its frontier is of size one.

Theorem 11 *Let KB be a frontier-one Datalog+/– ontology, and Q be a BCQ. Deciding if (a) $KB \models_{Cons} Q$, (b) $KB \models_{ICons} Q$, and (c) $KB \models_{ICR} Q$ are co-NP-complete.*

Proof (sketch). Since classical query answering can be performed in polynomial time in the data complexity for frontier-one ontologies [4], membership in co-NP for the three problems can be shown similar to the way it is done for the guarded case. Hardness can be shown analogously to the hardness proof for the guarded case in [22], with a slight modification in the definition of the set of TGDs to ensure using frontier-one TGDs. Query answering under the $ICons$ semantics can be proved by reduction from the k-clique problem of undirected graphs: the set of TGDs contains the characterization of a clique of size k. Hardness in co-NP under the ICR semantics follows from the fact that consistent query answering for frontier-one Datalog+/– is shown to be co-NP-hard even for ground atomic queries and Proposition 7. □

By noticing that the "shape" of derived facts depends only on how the frontier of the TGDs is mapped (and not on how the whole body is mapped, since only the images of the frontier are used to apply a TGD), one obtains a generalization of both fr1- and guarded-rules: a TGD σ is frontier-guarded (fg) if there is an atom a in its body that contains all variables in its frontier, i.e., $vars(fr(\sigma)) \subseteq vars(a)$.

Theorem 12 *Let KB be a frontier-guarded Datalog+/– ontology, and Q be a BCQ. Deciding if (a) $KB \models_{Cons} Q$, (b) $KB \models_{ICons} Q$, and (c) $KB \models_{ICR} Q$ are co-NP-complete.*

Proof (sketch). Given that classical query answering can be performed in polynomial time in the data complexity for frontier-guarded ontologies [4], membership in co-NP for the three problems can easily be shown similarly to the guarded case. Furthermore, hardness for the consistent query answering problem and query answering under the $ICons$ semantics can be proved by reduction from the corresponding problems for frontier-one ontologies, whose data complexity was shown above. Hardness for query answering under the ICR semantics follows from the fact that consistent query answering for frontier-guarded is shown to be co-NP-hard even for ground atomic queries and Proposition 7. □

4.3 Weakly- and Joint-Acyclic Sets of TGDs

Other tractable cases are obtained by restricting possible interactions between TGDs. These interactions have been encoded in different directed graphs that represent encodings of variable sharing between positions in predicates or the encoding of dependencies between TGDs. Here we briefly recall the notion of *(position) dependency graph* [17]. The nodes in a dependency graph represent positions in predicates, i.e., the node (p, i) represents a position i in predicate p. For each TGD σ and each variable X in $body(\sigma)$ occurring in position i, edges with origin (p, i) are built as follows: if X is in the frontier of σ, there is an edge from (p, i) to each position of X in $head(\sigma)$; furthermore, for each existential variable Y in $head(\sigma)$ occurring in position (q, j), there is a *special* edge from (p, i) to (q, j). A set of rules is said to be *w*eakly-acyclic if its dependency graph has no cycle passing through a special edge. Weakly-acyclic sets of TGDs strictly include both sets of full TGDs and acyclic sets of inclusion dependencies [17].

Theorem 13 *Let KB be a weakly-acyclic Datalog+/– ontology, and Q be a BCQ. Deciding if (a) KB $\models_{Cons}$ Q, (b) KB $\models_{ICons}$ Q, and (c) KB $\models_{ICR}$ Q are co-NP-complete.*

Proof (sketch). Membership in co-NP for the three problems can be obtained in the same way that is done for the guarded case since query answering for weakly-acyclic sets of TGDs is in PTIME [15, 17]. As for co-NP hardness for consistent query answering, the reduction provided in [22] to show co-NP hardness can be used in this case since the set of TGDs generated in the reduction is trivially a set of weakly-acyclic set of TGDs. Co-NP hardness can be shown for the $ICons$ by slightly modifying the reduction in Theorem 11. Finally, co-NP hardness for the ICR semantics follows directly from the fact that consistent query answering for weakly-acyclic is shown to be co-NP-hard even for ground atomic queries and Proposition 7. □

Joint-acyclicity [19]. This class is obtained by simply shifting the focus from positions to existential variables; that is, replacing the position dependency graph by the existential dependency graph, where the nodes are the existential variables occurring in TGDs. This yields a finer analysis of potentially infinite creations of existential variables.

Theorem 14 *Let KB be a joint-acyclic Datalog+/– ontology, and Q be a BCQ. Deciding if KB $\models_{Cons}$ Q, KB $\models_{ICons}$ Q, and KB $\models_{ICR}$ Q are co-NP-complete.*

Proof (sketch). Membership in co-NP for all three problems can be obtained in the same way that is done for the guarded Datalog+/– since classical query answering for joint-acyclic sets of TGDs is in PTIME [19]. co-NP hardness for all three problems follows directly from the fact that joint-acyclic sets of TGDs strictly generalize weakly-acyclic sets of TGDs. □

4.4 Sticky Sets of TGDs

The stickiness property restricts multiple occurrences of variables (in the same atom or in distinct atoms, i.e., in joins) in the TGD bodies. The class of *sticky* sets of TGDs is defined in [11] by a syntactic criterion that is easily testable, which is as follows. For every database D, assume that during the chase (or forward chaining processing) of D regarding a set Σ of TGDs, we apply a TGD $\sigma \in \Sigma$ which has a variable V appearing more than once in its body. Assume also that V maps (via a homomorphism) on the symbol z, and that by virtue of this application the atom a is generated. In this case, for each atom $b \in \mathit{body}(\sigma)$ we say that a is derived from b. Then, z appears in a, and in all atoms resulting from some derivation sequence starting from a, "sticking" to them (hence the name "sticky" sets of TGDs). We mark the variables that occur in the body of the TGDs of Σ_T according to the following marking procedure. First, for each TGD $\sigma \in \Sigma_T$ and for each variable V in $\mathit{body}(\sigma)$, if there exists an atom a in $\mathit{head}(\sigma)$ such that V does not appear in a, then we mark each occurrence of V in $\mathit{body}(\sigma)$. Now, we apply exhaustively (i.e., until a fixpoint is reached) the following step: for each TGD $\sigma \in \Sigma_T$, if a marked variable in $\mathit{body}(\sigma)$ appears at position π, then for every TGD $\sigma' \in \Sigma_T$ (including the case $\sigma = \sigma'$), we mark each occurrence of the variables in $\mathit{body}(\sigma')$ that appear in $\mathit{head}(\sigma')$ at the same position π. We say that Σ_T is sticky if there is no TGD $\sigma \in \Sigma_T$ such that a marked variable occurs in $\mathit{body}(\sigma)$ more than once.

Theorem 15 *Let KB be a sticky Datalog+/– ontology, and Q be a BCQ. Deciding if (a) $KB \models_{Cons} Q$ and if (b) $KB \models_{ICR} Q$ are both co-NP-complete. Furthermore, (c) query answering for BCQs under ICons is FO-rewritable.*

Proof (sketch). Hardness in co-NP for consistent query answering follows from the fact that sticky Datalog+/– generalizes *DL-Lite*$_{core}$ (the translation of a *DL-Lite*$_{core}$ *TBox* yields sticky sets of TGDs), a DL for which the problem is co-NP-complete [20]. Membership can be shown analogously to the guarded case since query answering for sticky sets of TGDs is in PTIME [12]. FO-rewritability for the $ICons$ semantics follows from Proposition 8. Finally, consistent query answering is co-NP hard even for ground atomic queries [20]; by Proposition 7, this means that it is also hard for *DL-Lite*$_{core}$ under the ICR semantics. Therefore, query answering under the ICR semantics is also co-NP hard in the sticky case. □

Finally, sticky sets of TGDs are not expressive enough to model simple cases such as the following linear TGD $p(X, Y, X) \rightarrow \exists Z q(Y, Z)$; variable X is marked and therefore the stickiness condition violated. The class of *sticky-join* TGDs extends both linear and sticky TGDs preserving tractability (FO-rewritability) for query answering [12]. As in the case of sticky, sticky-join sets of TGDs are defined by a testable condition based on variable-marking, though a more sophisticated one. For lack of space we omit the formal definition of such marking (cf. [12] for more details on this class).

Theorem 16 *Let KB be a sticky-join Datalog+/– ontology, and Q be a BCQ. Deciding if (a) $KB \models_{Cons} Q$ and if (b) $KB \models_{ICR} Q$ are both co-NP-complete. Also, (c) query answering for BCQs under ICons is FO-rewritable.*

Proof (sketch). Membership in co-NP for consistent query answering and query answering under the ICR semantics can be shown analogously to the guarded case, since query answering for weakly-sticky sets of TGDs is in PTIME [12]. Hardness for co-NP in both cases follows directly from the fact that sticky-join Datalog+/– generalizes linear Datalog+/–, a fragment for which it is already shown that the problems are co-NP-complete (cf. Theorem 10). FO-rewritability for the $ICons$ semantics follows from Proposition 8. □

The class of *weakly-sticky* [12] TGDs generalizes both weakly-acyclic and sticky sets of TGDs. Intuitively, in a weakly-sticky set of TGDs, the variables that appear more than once in the body of a TGD are either non-marked (as defined above for sticky TGDs), or occur at positions where a finite number of distinct values can appear during the chase. Lastly, we look at weakly-sticky-join [12] sets of TGDs, which generalize both weakly-sticky sets and sticky-join sets.

Theorem 17 *Let KB be a weakly-sticky(-join) Datalog+/– ontology, and Q be a BCQ. Deciding if (a) $KB \models_{Cons} Q$, (b) $KB \models_{ICons} Q$, and (c) $KB \models_{ICR} Q$ are co-NP-complete.*

5 Summary and Outlook

In this work, we have studied the problem of inconsistency-tolerant query answering of BCQs for a wide range of tractable fragments of Datalog+/–. For each of the 11 languages chosen, we have determined the data complexity of consistent query answering and of query answering under two of its approximations proposed recently in the literature: $ICons$ (intersection of repairs) and ICR (intersection of closed repairs) semantics. As has been shown in previous work, the problem of computing consistent answers is a hard problem even for very simple languages; therefore, the results obtained for this semantics are not surprising, though the tractability of query answering in these languages at least yields non-deterministic polynomial time upper bounds. On the other hand, we see that for the $ICons$ semantics, there is a large gap in the data complexity, when going from FO-rewritable fragments of Datalog+/– to those for which classical query answering is *PTIME*-complete. Finally, for the chosen languages, the ICR semantics proved to be as hard as the consistent answers to compute. These results are suggestive, showing that it is still necessary to look for alternative inconsistency-tolerant semantics that provide a better tradeoff between quality of answers and efficiency.

The present work can be continued along several lines. For instance, there are several more inconsistency-tolerant semantics that would be interesting to analyze for this set of languages. Furthermore, it would be very important for practical purposes to search for other fragments of Datalog+/– that allow for tractable query answering under any of the current semantics existing in the literature. The complexity results of this paper provide us with a direction for this search.

Acknowledgments. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J008346/1 “PrOQAW: Probabilistic Ontological Query Answering on the Web”, the European Research Council (FP7/2007-2013)/ERC grant 246858 (“DIADEM”), and by a Yahoo! Research Fellowship.

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proc. PODS-1999. pp. 68–79. ACM Press (1999)
2. Baget, J.F., Leclère, M., Mugnier, M.L.: Walking the decidability line for rules with existential variables. In: Proc. KR-2010. pp. 466–476. AAAI Press (2010)
3. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. In: Proc. IJCAI-2009. pp. 677–682 (2009)
4. Baget, J.F., Mugnier, M.L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: Proc. IJCAI-2011. pp. 712–717. IJCAI/AAAI Press (2011)
5. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. ICALP-1981. LNCS, vol. 115, pp. 73–85. Springer (1981)
6. Bienvenu, M.: First-order expressibility results for queries over inconsistent *DL-Lite* knowledge bases. In: Proc. DL-2011. CEUR Workshop Proceedings, vol. 745. CEUR-WS.org (2011)
7. Bienvenu, M.: Inconsistency-tolerant conjunctive query answering for simple ontologies. In: Proc. DL-2012. CEUR Workshop Proceedings, vol. 846. CEUR-WS.org (2012)
8. Bienvenu, M.: On the complexity of consistent query answering in the presence of simple ontologies. In: Proc. AAAI-2012. pp. 705–711. AAAI Press (2012)
9. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. KR-2008. pp. 70–80. AAAI Press (2008)
10. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
11. Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. Proceedings of the VLDB Endowment 3(1/2), 554–565 (2010)
12. Calì, A., Gottlob, G., Pieris, A.: Query answering under non-guarded rules in Datalog+/-. In: Proc. RR-2010. LNCS, vol. 6333, pp. 1–17. Springer (2010)
13. Chandra, A.K., Lewis, H.R., Makowsky, J.A.: Embedded implicational dependencies and their inference problem. In: Proc. STOC-1981. pp. 342–354. ACM Press (1981)
14. Chomicki, J.: Consistent query answering: Five easy pieces. In: Proc. ICDT-2007. LNCS, vol. 4353, pp. 1–17. Springer (2007)
15. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Comput. Surv. 33(3), 374–425 (2001)
16. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Proc. PODS-2008. pp. 149–158. ACM Press (2008)
17. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comp. Sci. 336(1), 89–124 (2005)
18. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: Proc. IJCAI-2005. pp. 354–359. Professional Book Center (2005)
19. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Proc. IJCAI-2011. pp. 963–968. IJCAI/AAAI Press (2011)
20. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Proc. RR-2010. LNCS, vol. 6333, pp. 103–117. Springer (2010)
21. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Query rewriting for inconsistent *DL-Lite* ontologies. In: Proc. RR-2011. LNCS, vol. 6902, pp. 155–169. Springer (2011)
22. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Inconsistency handling in Datalog+/– ontologies. In: Proc. ECAI-2012. pp. 558–563. IOS Press (2012)

23. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Inconsistency-tolerant query rewriting for linear Datalog+/–. In: Proc. Datalog 2.0. LNCS, vol. 7494, pp. 123–134. Springer (2012)
24. Ma, Y., Hitzler, P.: Paraconsistent reasoning for OWL 2. In: Proc. RR-2009. LNCS, vol. 5837, pp. 197–211. Springer (2009)
25. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Proc. WWW-2005. pp. 633–640. ACM Press (2005)
26. Qi, G., Du, J.: Model-based revision operators for terminologies in description logics. In: Proc. IJCAI-2009. pp. 891–897 (2009)
27. Rosati, R.: On the complexity of dealing with inconsistency in description logic ontologies. In: Proc. IJCAI-2011. pp. 1057–1062. IJCAI/AAAI Press (2011)