Complexity of Inconsistency-Tolerant Query Answering in Datalog+/– under Cardinality-Based Repairs

Thomas Lukasiewicz  
Department of Computer Science  
University of Oxford, UK

Enrico Malizia  
Department of Computer Science  
University of Exeter, UK

Andrius Vaicenavičius  
Department of Computer Science  
University of Oxford, UK

Abstract

Querying inconsistent ontological knowledge bases is an important problem in practice, for which several inconsistency-tolerant query answering semantics have been proposed, including query answering relative to all repairs, relative to the intersection of repairs, and relative to the intersection of closed repairs. In these semantics, one assumes that the input database is erroneous, and the notion of repair describes a maximally consistent subset of the input database, where different notions of maximality (such as subset and cardinality maximality) are considered. In this paper, we give a precise picture of the computational complexity of inconsistency-tolerant (Boolean conjunctive) query answering in a wide range of Datalog languages under the cardinality-based versions of the above three repair semantics.

Introduction

In many ontology-based applications in practice, such as ontology-based data extraction from the Web, or ontology-based integration of different data sources, it is very likely that the data are inconsistent with the ontology, and thus inconsistency-tolerant query answering are urgently needed. Among the most prominent ontology languages are description logics (DLs) and existential rules from the context of Datalog languages under the cardinality-based version of inconsistency-tolerant (Boolean conjunctive) query answering is very well-understood. Rosati (2011) studied the data and combined complexity for a wide spectrum of DLs, while Bienvenu (2012) identified cases for simple ontologies (within the DL-Lite family) for which tractable data complexity results can be obtained. In (Lukasiewicz, Martinez, and Simari 2012; 2013; Lukasiewicz et al. 2015), the data and different types of combined complexity of consistent query answering have been studied for ontologies described via existential rules and negative constraints, and such investigation has recently been extended to the IAR and ICR semantics in (Lukasiewicz, Malizia, and Molinaro 2018a).

There are several reasons for the practical relevance of the IAR and the ICR semantics, and thus for motivating an in-depth analysis of their computational properties. First, they are two natural semantics that identify “surer” answers than consistent query answering, and so they can also be seen as under-approximations of the latter. Investigating their complexity helps to understand whether such approximations have actually lower complexities, which is the case for different languages and one complexity measure considered in this paper. Second, recent work on explanations in the context of inconsistency-tolerant query answering shows that explanations are much easier to define and compute for the IAR semantics (Bienvenu, Bourgaux, and Goasdoué 2016). Third, a crucial advantage of the IAR and the ICR semantics is that their intersection of (closed) repairs can be materialized, while the consistent query answering semantics exists only virtually (the intersection of (closed) repairs can be computed offline, and then standard querying algorithms can be employed online)—indeed, this has been used to implement the IAR semantics (Lembo et al. 2015), and for ICR, it has been remarked in (Bienvenu and Bourgaux 2016).

The complexity of consistent query answering when the ontology is described via one of the main DLs is well-understood. Rosati (2011) studied the data and combined complexity for a wide spectrum of DLs, while Bienvenu (2012) identified cases for simple ontologies (within the DL-Lite family) for which tractable data complexity results can be obtained. In (Lukasiewicz, Martinez, and Simari 2012; 2013; Lukasiewicz et al. 2015), the data and different types of combined complexity of consistent query answering have been studied for ontologies described via existential rules and negative constraints, and such investigation has recently been extended to the IAR and ICR semantics in (Lukasiewicz, Malizia, and Molinaro 2018a).

Alternative notions of maximality for repairs, however, such as cardinality-maximal repairs (Lopatenko and Bertossi 2007), rather than subset-maximal ones, have been
We analyze the data, fixed-program combined, bounded-

We consider the most popular Datalog

\[ D_p \]

instance

a first-order formula

\[ h \]

over

answer

is the identity on

General.

we will encounter in our analysis in this paper.

We now briefly recall some basics on Datalog

languages and for several different complexity measures:

▷ We consider different popular inconsistency-tolerant se-

manics, namely, the AR, the IAR, and the ICR semantics,

with maximum cardinality database repairs.

▷ We consider the most popular Datalog\( ^\pm \) languages:

linear, guarded, sticky, and acyclic existential rules, along

with “weak” generalizations, as well as full (i.e., non-

existential) restrictions, and full rules in general.

▷ We analyze the data, fixed-program combined, bounded-

arity combined, and combined complexity.

Detailed proofs are given in a forthcoming extended paper.

Preliminaries

We now briefly recall some basics on Datalog\( ^\pm \) (Cali, Gott-

lob, and Lukasiewicz 2012) and the complexity classes that

we will encounter in our analysis in this paper.

General. We assume sets \( C, N, \) and \( V \) of constants, labeled

nulls, and variables, respectively. A term \( t \) is a constant, null,

or variable. We also assume a set of predicates, each with

an arity, i.e., a non-negative integer. 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 \( p(t) \), where \( t \) is a tuple of constants and nulls. A
database \( D \) is a finite instance without nulls. A homomor-

phism is a mapping \( h: C \cup N \cup V \to C \cup N \cup V \) that

is the identity on \( C \) and maps \( N \) to \( C \cup N \). A conjunc-
tive query (CQ) \( Q \) has the form \( \exists Y \phi(X, Y) \), where \( \phi(X, Y) \)
is a conjunction of atoms without nulls. The answer to \( Q \)
over an instance \( I \), denoted \( Q(I) \), is the set of all tuples

\( t \) over \( C \) for which there is a homomorphism \( h \) such that

\[ h(\phi(X, Y)) \subseteq I \text{ and } h(X) = t. \]

A Boolean CQ (BCQ) \( Q \) is a CQ \( \exists Y \phi(Y) \), i.e., all variables are existentially quantified;

\( Q \) is true over \( I \), denoted \( I \models Q \), if \( Q(I) \neq \emptyset \), i.e., there is

a homomorphism \( h \) with \( h(\phi(Y)) \subseteq I \).

 Dependencies. A tuple-generating dependency (TGD) \( \sigma \)
is a first-order formula \( \forall X \exists Y \phi(X, Y) \rightarrow \exists Z p(X, Z) \), where

\( X \cup Y \cup Z \subseteq V, \phi(X, Y) \) is a conjunction of atoms, and

\( p(X, Z) \) is an atom, all without nulls; \( \phi(X, Y) \) is the body

of \( \sigma \), denoted \( body(\sigma) \), while \( p(X, Z) \) is the head of \( \sigma \), de-
noted \( head(\sigma) \). 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 \( \sigma \),

written \( I \models \sigma \), if the following holds: whenever there exists

a homomorphism \( h \) such that \( h(\phi(X, Y)) \subseteq I \), then there

exists \( h' \supseteq h|_X \), where \( h|_X \) is the restriction of \( h \) on \( X \),

such that \( h'(p(X, Z)) \subseteq I \). A negative constraint (NC) \( \nu \)
is a first-order formula \( \forall X \phi(X) \rightarrow \bot \), where \( X \subseteq V \), \( \phi(X) \)
is a conjunction of atoms without nulls, called the body of \( \nu \),
denoted \( body(\nu) \), and \( \bot \) denotes the truth constant false. An
instance \( I \) satisfies \( \nu \), written \( I \models \nu \), if there is no homomor-

phism \( h \) such that \( h(\phi(X)) \subseteq I \). Given a set \( \Sigma \) of TGDs and

NCs, \( I \) satisfies \( \Sigma \), written \( I \models \Sigma \), if \( I \) satisfies each TGD

and NC of \( \Sigma \). For brevity, we omit the universal quantifiers

in front of TGDs and NCs, and use the comma (instead of \( \land \)) for conjoining body atoms. Given a class of TGDs \( \mathcal{C} \), we
denote by \( \mathcal{C}_+ \) the formalism obtained by combining \( \mathcal{C} \) with

arbitrary NCs. Finite sets of TGDs and NCs are also called

programs, and TGDs are also called existential rules.

Knowledge Bases. A knowledge base is a pair \( (D, \Sigma) \),

where \( D \) is a database, and \( \Sigma \) is a program. For programs \( \Sigma, \Sigma_T \) and \( \Sigma_{NC} \) are the subsets of \( \Sigma \) containing the TGDs and

NCs of \( \Sigma \), respectively. The set of models of \( KB = (D, \Sigma) \),
denoted \( mods(KB) \), is the set of instances \( \{ I \mid I \supseteq D, I \models \Sigma \} \).
We say that \( KB \) is consistent, if \( mods(KB) \neq \emptyset \), oth-

erwise \( KB \) is inconsistent. The answer to a CQ \( Q \) relative to

\( KB \) is the set of tuples \( ans(Q, KB) = \bigcap \{ Q(I) \mid I \in mods(KB) \} \).
The answer to a BCQ \( Q \) is true, denoted \( KB \models Q \), if \( ans(Q, KB) \neq \emptyset \). The decision version of the CQ answer-

ing problem is as follows: given a knowledge

base \( KB \), a CQ \( Q \), and a tuple of constants \( t \), decide

whether \( t \in ans(Q, KB) \). Since CQ answering can be re-

duced in LogSpace to BCQ answering, we focus on BCQs. Following Vardi (1982), the combined complexity of BCQ

answering considers the database, the set of dependencies,

and the query as part of the input. The bounded-arity com-

bined (or ba-combined) complexity assumes that the arity of

the underlying schema is bounded by an integer constant.

The fixed-program combined (or fp-combined) complexity

considers the sets of TGDs and NCs as fixed; the data com-

plexity also assumes the query fixed.

Decidability Paragrigms. The main (syntactic) conditions

on TGDs that guarantee the decidability of BCQ answer-

ing are guardedness (Cali, Gottlob, and Kifer 2013), stick-

iness (Cali, Gottlob, and Pieris 2012), and acyclicity, each

having a “weak” counterpart: weak guardedness (Cali, Got-

tlob, and Kifer 2013), weak stickiness (Cali, Gottlob, and

Pieris 2012), and weak acyclicity (Fagin et al. 2005).

A TGD \( \sigma \) is guarded, if there exists an atom in the body

that contains (or “guards”) all the body variables of \( \sigma \). The
class of guarded TGDs, denoted \( G \), is the family of all possi-

ble sets of guarded TGDs. A key subclass of guarded TGDs

is linear TGDs with just one body atom (which is 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 \( L \subseteq G \subseteq WG \).

Stickiness is inherently different from guardedness, and

its central property is as follows: variables that appear more

than once in a body (i.e., join variables) are always propa-

gated (or “stick”) to the inferred atoms. A set of TGDs with

the above property is 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 that enjoys weak stickiness is weakly sticky, and

the associated class is denoted \( WS \). Observe that \( S \subseteq WS \).
A set $\Sigma$ of TGDs is acyclic, if its predicate graph is acyclic, and the underlying class is denoted $A$. $\Sigma$ is 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, $A \subseteq WA$. Observe also that $WA \subseteq WS$.

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 full TGDs enjoy linearity, guardedness, stickiness, or acyclicity, then we obtain the classes $LF$, $GF$, $SF$, and $AF$, respectively. Observe that $F \subseteq WA$ and $F \subseteq WG$.

For a summary of the complexity of (classical) BCQ answering over these Datalog variant languages, see Table 1.

**Complexity Classes.** The complexity class $AC^0$ is the class of all decision problems that can be solved by uniform families of Boolean circuits of polynomial size and constant depth. $PSPACE$ (resp., $P$, $EXP$, $2EXP$) is the class of all problems that can be decided in polynomial space (resp., polynomial time, exponential time, double exponential time) on a deterministic Turing machine. $NP$ and $NEXP$ are the classes of all problems that are decidable in polynomial and exponential time on a nondeterministic Turing machine, respectively, and co-$NP$ and co-$NEXP$ are their complementary classes, where ‘yes’ and ‘no’ instances are interchanged. The class $\Sigma^p_2$ is the class of all problems that can be decided in nondeterministic polynomial time using an $NP$ oracle, and $\Pi^p_2$ is the complement of $\Sigma^p_2$. The class $\Theta^p_2$ (resp., $\Theta^p_3$) is the class of all problems that can be decided in polynomial time by a deterministic Turing machine with either a logarithmic number of calls to an $NP$ oracle (resp., $\Sigma^p_2$ oracle), or (equivalently) a constant number of rounds of polynomially many parallel calls to an $NP$ oracle (resp., $\Sigma^p_2$ oracle). $P^{\text{nEXP}}$ is the class of all problems that are decidable in deterministic polynomial time using a $\text{nEXP}$ oracle. The above complexity classes and their inclusion relationships (which are all currently believed to be strict) are: $AC^0 \subseteq P \subseteq \text{co-NP} \subseteq \Theta^p_2 \subseteq \Sigma^p_2$, $\Pi^p_2 \subseteq \Theta^p_3 \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP}$, co-$\text{NEXP} \subseteq P^{\text{nEXP}} \subseteq 2\text{EXP}$.

**Inconsistency-Tolerant Semantics.** In classical BCQ answering, for an inconsistent knowledge base $KB$ (i.e., $\text{mods}(KB) = \emptyset$), every query is entailed, as everything follows from a contradiction. Clearly, the answers obtained in such cases are not meaningful.

Several inconsistency-tolerant semantics have been proposed. We now recall three prominent ones for ontology-based query answering, namely, the ABox repair ($AR$), the intersection of repairs ($IAR$), and the intersection of closed repairs ($ICR$) semantics (Lembo et al. 2010; Bienvenu 2012); all three are based on the notion of repair, which is a maximal consistent subset of the given database. Below we define these inconsistency-tolerant semantics for a generic concept of repair maximality.

Given a knowledge base $KB = (D, \Sigma)$, a selection $D'$ of $KB$ is a database such that $D' \subseteq D$. A selection $D'$ of $KB$ is consistent, if $\text{mods}((D', \Sigma)) \neq \emptyset$. Consistent selections of knowledge bases can be ordered according to some criteria to select the more desired ones. Given a preorder $\sqsubseteq$ over a set $\mathcal{S}$ of databases, for two elements $D'$ and $D'' \in \mathcal{S}$, $D' \prec D''$ denotes that $D' \sqsubseteq D''$ and $D'' \not\sqsubseteq D'$. A database $D \in \mathcal{S}$ is $\prec$-maximal in $\mathcal{S}$ if there is no $D' \in \mathcal{S}$ such that $D \prec D'$.

**Definition 1.** A $\prec$-repair of a knowledge base $KB$ is a consistent selection of $KB$ that is $\prec$-maximal in the set of all the consistent selections of $KB$.

We now define the three different inconsistency-tolerant semantics for BCQ answering. In what follows, $\text{Rep}_{\prec}(KB)$ denotes the set of all $\prec$-repairs of a knowledge base $KB$. For a knowledge base $KB = (D, \Sigma)$, the closure $\text{Cl}(KB)$ of $KB$ is the set of all ground atoms, built from constants in $D$ and $\Sigma$, entailed by $D$ and the TGDs of $\Sigma$.

**Definition 2.** Let $KB$ be a knowledge base, let $Q$ be a BCQ, and let $\prec$ be an order over the consistent selections of $KB$.

- $KB$ entails $Q$ under the ABox repair semantics and order $\prec$ ($\prec$-AR), denoted by $KB \models_{\prec-AR} Q$, if, for all $D' \in \text{Rep}_{\prec}(KB)$, $(D', \Sigma) \models Q$.
- $KB$ entails $Q$ under the intersection of repairs semantics and order $\prec$ ($\prec$-IAR), denoted by $KB \models_{\prec-IAR} Q$, if $(D^*, \Sigma) \models Q$, where $D^* = \bigcap\{D' \mid D' \in \text{Rep}_{\prec}(KB)\}$.
- $KB$ entails $Q$ under the intersection of closed repairs semantics and order $\prec$ ($\prec$-ICR), denoted by $KB \models_{\prec-ICR} Q$, if $(D_1, \Sigma) \models Q$, where $D_1 = \bigcap\{\text{Cl}(D') \mid D' \in \text{Rep}_{\prec}(KB)\}$.

Different orders over the set of consistent selections give rise to different inconsistency-tolerant semantics for BCQ answering, because they select different repairs.

Inclusion-maximal repairs have often been considered in the literature. An interesting class of repairs are those selected by the cardinality order ‘≤’ (Bienvenu, Bourgaux, and Goasdoué 2014). In this order, consistent selections of larger cardinality are preferred over ones of smaller cardinality (without looking at inclusion-wise relationships between the selections: only the cardinality counts). Hence, a $\preceq$-repair of a knowledge base $KB$ is a consistent selection of $KB$ of maximum cardinality. In this paper, we consider only the ‘≤’ order, therefore, we often call $\preceq$-repairs simply repairs, and by $\text{Rep}(KB)$, we mean $\text{Rep}_{\preceq}(KB)$.

Cardinality-maximal repairs are very appropriate when it is known (or believed) that all the facts in the database have the same (possibly small) probability of being erroneous. In these cases, larger repairs are preferred, because fewer facts are dropped (Bienvenu, Bourgaux, and Goasdoué 2014). When facts in the database have different likelihoods of being erroneous, then other concepts of repairs can also be taken into consideration (Bienvenu 2012).

It can be shown that the following semantic relationships hold between the above three different semantics.

**Proposition 3.** Let $KB$ be a knowledge base, let $Q$ a BCQ. $(KB \models_{\preceq-IAR} Q) \Rightarrow (KB \models_{\preceq-ICR} Q) \Rightarrow (KB \models_{\preceq-AR} Q)$.

**Overview of Complexity Results.**

We give a precise picture of the complexity of BCQ answering from existential rules under the $\preceq-AR$, $\preceq-IAR$, and $\preceq-ICR$ inconsistency-tolerant semantics. Our results are summarized in Tables 2 and 3, and they range from $\Theta^p_2$-completeness to $2\text{EXP}$-completeness.
Intuitively, in order to combine classical reasoning in the data / subset-maximal repairs (see Lukasiewicz et al. 2015). The complexity compared to the case of AR semantics, we observe that there is an algorithmically many calls to an oracle for answering or the complexity associated with the evaluation of the inconsistency-tolerant semantics. This extra effort does not always influence the computational cost for using IAR semantics with subset-maximal repairs (see, Lukasiewicz et al. 2015). Different proof of membership in (Bienvenu, Bourgaux, and Goasdoué 2014). Different proof of hardness for IAR in (Bienvenu, Bourgaux, and Goasdoué 2014).

In this section, we first derive the membership results and then the hardness results of Tables 2 and 3.

### Derivation of Complexity Results

The following theorem shows that if BCQ answering from knowledge bases over some Datalog language \( L \) is in \( \mathbb{C} \) in the data (resp., ba-combined) and combined complexity, then \( \leq \text{IAR}-\text{BCQ} \) and \( \leq \text{ICR}-\text{BCQ} \) answering from knowledge bases over \( L \) can be done in polynomial time with logarithmically many calls to an oracle for \( NP \) in the data (resp., ba-combined and combined) complexity. This result holds for \( \leq \text{ICR}-\text{BCQ} \) answering as well, but only in the data and ba-combined complexity. For the combined complexity case of \( \leq \text{ICR}-\text{BCQ} \) answering, we will need a different proof.

**Theorem 4.** Let \( L \) be a Datalog language. If BCQ answering from knowledge bases over \( L \) is in \( \mathbb{C} \) in the data / ba-combined / combined complexity (resp., data / ba-combined complexity)

---

**Table 1:** Complexity of BCQ answering (Lukasiewicz et al. 2015). All non-“in” entries are completeness results.

<table>
<thead>
<tr>
<th>Data</th>
<th>( f_p )-comb.</th>
<th>ba-comb.</th>
<th>Comb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_{\perp}, LF_{\perp}, AF_{\perp} ) in AC( 0 )</td>
<td>NP</td>
<td>NP</td>
<td>PS( \text{SPACE} )</td>
</tr>
<tr>
<td>( G_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
</tr>
<tr>
<td>( WG_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
</tr>
<tr>
<td>( S_{\perp}, SF_{\perp} ) in AC( 0 )</td>
<td>NP</td>
<td>NP</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( F_{\perp}, GF_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
</tr>
<tr>
<td>( A_{\perp} ) in AC( 0 )</td>
<td>NP</td>
<td>NEXP</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( WS_{\perp}, WA_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
</tr>
</tbody>
</table>

---

**Table 2:** Complexity of \( \leq \text{AR} \) BCQ answering. All entries are completeness results. *Different proof of membership in (Bienvenu, Bourgaux, and Goasdoué 2014). *Different proof of hardness for \( L_{\perp}, G_{\perp}, S_{\perp}, \) and \( WS_{\perp} \) in (Bienvenu, Bourgaux, and Goasdoué 2014).

<table>
<thead>
<tr>
<th>Data</th>
<th>( f_p )-comb.</th>
<th>ba-comb.</th>
<th>Comb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_{\perp}, LF_{\perp}, AF_{\perp} ) ( \Theta_2^{\perp} ) *</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>PS( \text{SPACE} )</td>
</tr>
<tr>
<td>( G_{\perp} ) ( \Theta_2^{\perp} ) *</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( WG_{\perp} ) ( \Theta_2^{\perp} ) *</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( S_{\perp}, SF_{\perp} ) ( \Theta_2^{\perp} ) *</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( F_{\perp}, GF_{\perp} ) ( \Theta_2^{\perp} ) *</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( A_{\perp} ) ( \Theta_2^{\perp} ) *</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( WS_{\perp}, WA_{\perp} ) ( \Theta_2^{\perp} ) *</td>
<td>( \Pi_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
</tbody>
</table>

---

**Table 3:** Complexity of \( \leq \text{IAR} \) and \( \leq \text{ICR} \) BCQ answering. All entries are completeness results. *Different proof of membership for IAR in (Bienvenu, Bourgaux, and Goasdoué 2014). *Different proof of hardness for IAR for \( L_{\perp}, G_{\perp}, S_{\perp}, \) and \( WS_{\perp} \) in (Bienvenu, Bourgaux, and Goasdoué 2014).

<table>
<thead>
<tr>
<th>Data</th>
<th>( f_p )-comb.</th>
<th>ba-comb.</th>
<th>Comb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_{\perp}, LF_{\perp}, AF_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>PS( \text{SPACE} )</td>
</tr>
<tr>
<td>( G_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( WG_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( S_{\perp}, SF_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( F_{\perp}, GF_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( A_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
<tr>
<td>( WS_{\perp}, WA_{\perp} )</td>
<td>( \Theta_2^{\perp} )</td>
<td>( \Theta_3^p )</td>
<td>EXP ( 2 \text{EXP} )</td>
</tr>
</tbody>
</table>
Proof. Let $KB = (D, \Sigma)$ be a knowledge base over $L$, and let $q$ be a query. First, we compute the size $\text{max}$ of the largest consistent selections of $KB$. This can be done by a machine in polynomial time calling, logarithmically many times, an oracle in $NP^C$. Indeed, we can perform a binary search in the range $[0, |D|]$ by asking the oracle whether there is a consistent selection of size at least $k$. The oracle has to guess a subset of size $k$ of the database, and then check whether the guess is consistent. Since we are assuming that classical reasoning in fragment $L$ is in $C$, the oracle can guess a selection in $NP$ and then check its consistency in $C$.

Then, we exploit the $NP^C$ oracle to decide whether the BCQ query $q$ is entailed under the considered semantics. In particular, we ask the oracle whether the query is not entailed. The way in which the oracle finds the answer depends on the specific semantics, $AR$, $IAR$, or $ICR$, considered.

$AR$: The oracle guesses a consistent selection $D'$ of size $\text{max}$, and checks that $(D', \Sigma) \not\models q$.

$IAR$: The oracle guesses a database $D^* \subseteq D$ and checks that: (1) $(D^*, \Sigma) \not\models q$, and (2) there exist repairs $D'_\alpha$ with $\alpha \notin D'_\alpha$, one for each $\alpha \in D \setminus D^*$ (witnessing that the intersection of the repairs is a subset of $D^*$). Point (2) is checked via additional guesses of the various $D'_\alpha$.

$ICR$: The oracle verifies the existence of a subset $D'$ of $Cn(KB)$ (the size of $Cn(KB)$ is polynomial in the input, because the program has, in the worst case, bounded arity) such that: (i) for each atom $\alpha \in (Cn(KB) \setminus D^*)$, there is a repair $D'_\alpha$ such that $\alpha \notin Cn(D'_\alpha)$; (ii) $(D', \Sigma) \not\models q$. The oracle can do this by first guessing such a set $D'$ along with the witnesses $D'_\alpha$, and second checking that the various $D'_\alpha$ are consistent, their size is $\text{max}$, and $\alpha \notin Cn(D'_\alpha)$, and that $(D', \Sigma) \not\models q$. The existence of the various repairs $D'_\alpha$ with the properties above reported guarantees that $D_I \subseteq D'$, and $(D', \Sigma) \not\models q$ implies $(D_I, \Sigma) \not\models q$ by monotonicity.

As mentioned above, to characterize the combined complexity of $\leq-ICR$-BCQ answering, we need a different proof. Indeed, in a combined complexity scenario, the size of $Cn(KB)$ is exponential in the size of the input, and the $D' \subseteq Cn(KB)$ needed to be guessed in the ICR case of the proof above could be too big for an NP machine.

The next result proves all the upper bounds for $\leq-ICR$ BCQ answering in Table 3 for the combined complexity.

Theorem 5. $\leq-ICR$ BCQ answering from knowledge bases over Datalog languages $L$ in the combined complexity is in the complexity classes shown in Table 3.

We next analyze the $fp$-combined complexity and show that if BCQ answering from knowledge bases over some Datalog language $L$ in $D$ (resp., $C$) in the data (resp., $fp$-combined) complexity, then $\leq-AR$, $\leq-IAR$, and $\leq-ICR$, BCQ answering from knowledge bases over $L$ can be done in polynomial time with logarithmically many calls to an oracle for $\text{NP}^D$ followed by a computation in $\text{co-NP}^C$ (for the $\leq-AR$) or a computation in $C$ (for the $\leq-IAR$ and $\leq-ICR$). This is shown in a similar way as Theorem 4.

Theorem 6. If BCQ answering from knowledge bases over a Datalog language $L$ is in $D$ in the data complexity and in $C$ in the $fp$-combined complexity, then $\leq-AR$ (resp., $\leq-IAR$ and $\leq-ICR$) BCQ answering from knowledge bases over $L$ is possible by a computation in $p$ with an oracle for $\text{NP}^D(O(|\log n|))$, followed by a computation in $\text{co-NP}^C$ (resp., $C$), in the $fp$-combined complexity.

As a corollary of the theorems above, we obtain all the upper bounds for $\leq-AR$ BCQ answering in Table 2, and all the upper bounds for $\leq-IAR$ and $\leq-ICR$ BCQ answering in Table 3. In particular, the $\Theta_2^p$ membership of $\leq-AR$ and $\leq-IAR$ BCQ answering in the data complexity for all the languages considered.

Corollary 7. $\leq-AR$ (resp., $\leq-IAR$ and $\leq-ICR$) BCQ answering from knowledge bases over Datalog languages $L$ in Table 2 (resp., Table 3) in the $fp$-combined, combined, and combined complexity belongs to the complexity classes shown in Table 2 (resp., Table 3).

Hardness Results

The hardness results not explicitly proven follows from the hardness of classical reasoning over consistent ontologies.

We now show that $\leq-AR$, $\leq-IAR$, and $\leq-ICR$ BCQ answering is $\Theta_2^p$-hard in the data complexity for all the Datalog languages considered.

Theorem 8. For every $C \in \{AR, IAR, ICR\}$, $\leq-C$ BCQ answering from knowledge bases over LFI, AFI, and SF (and hence also for L, A, S, G, GF, F, WS, and WA) is $\Theta_2^p$-hard in the data complexity.

Proof. We use a reduction from the $\Theta_2^p$-complete problem INALLMAXIS (Lopatenko and Bertossi 2007; 2016): given a graph $G$ and a vertex $v$, decide whether $v$ belongs to all the independent sets of $G$ of maximum size.

Let $(G, w)$ be an instance of INALLMAXIS, where $G = (V, E)$ and $w \in V$, and $n = |V|$. From $(G, w)$ we build the knowledge base $KB_{\text{MaxIS}} = (D_{\text{MaxIS}}, \Sigma_{\text{MaxIS}})$ and the query $q_{\text{MaxIS}}$ as follows.

For each vertex $v \in V$, there is a fact $\text{In}(v)$ in $D_{\text{MaxIS}}$. For each edge $(v_1, v_2) \in E$, there are $n$ facts $\text{Edge}(v_1, v_2, i)$ in $D_{\text{MaxIS}}$, with $1 \leq i \leq n$. $D_{\text{MaxIS}}$ contains also the fact $\text{dist}(w)$, and the query $q_{\text{MaxIS}}$ as follows.

For each vertex $v \in V$, there is a fact $\text{In}(v)$ in $D_{\text{MaxIS}}$. For each edge $(v_1, v_2) \in E$, there are $n$ facts $\text{Edge}(v_1, v_2, i)$ in $D_{\text{MaxIS}}$, with $1 \leq i \leq n$. $D_{\text{MaxIS}}$ contains also the fact $\text{dist}(w)$. The only dependency in $\Sigma_{\text{MaxIS}}$ is the NC $\text{In}(X) \land \text{In}(Y) \land \text{Edge}(X, Y, Z) \rightarrow \bot$. Finally, $q_{\text{MaxIS}} = \exists X(\text{In}(X) \land \text{dist}(w))$. Observe that $\Sigma_{\text{MaxIS}}$ is vacuously linear, sticky, acyclic, and full.

We can show that $(G, w)$ is a ‘yes’-instance of INALLMAXIS iff $KB_{\text{MaxIS}}$ entails $q_{\text{MaxIS}}$ under any $\leq-C$, for $C \in \{AR, IAR, ICR\}$. Intuitively, repairs contain all facts $\text{Edge}(v_1, v_2, i)$, $\text{dist}(w)$ and a set of facts $\text{In}(v)$ corresponding to a maximum independent set of $G$. 

□
For $\leq AR$ BCQ answering, we show its $\Pi_3^p$-hardness in the $fp$-combined complexity, for all the Datalog languages considered. This follows from a reduction in (Lukasiewicz et al. 2015), which also applies in the case of maximum cardinality. Note that also a simplification of the reduction used here to prove Theorem 10 can be used to show this result.

**Theorem 9.** $\leq AR$ BCQ answering from knowledge bases over $LF_\perp$, $AF_\perp$, and $SF_\perp$ (and hence also for $L_\perp$, $A_\perp$, $S_\perp$, $G_\perp$, $GF_\perp$, $F_\perp$, $WS_\perp$, and $WA_\perp$) is $\Pi_3^p$-hard in the $fp$-combined complexity.

We now show that, for any $C \in \{AR, IAR, ICR\}$, $\leq C$ BCQ answering is $\Theta_3^p$-hard in the $ba$-combined complexity, already for the simplest languages here considered.

**Theorem 10.** For every $C \in \{AR, IAR, ICR\}$, $\leq C$ BCQ answering from knowledge bases over $L,F,S,G,GF,F,WS,WA$ (and hence also for $L_S$, $G_S$, $GF_S$, and $F_S$) is $\Theta_3^p$-hard in the $ba$-combined complexity.

**Proof sketch.** We prove the statement via a reduction from the $\Theta_3^p$-complete problem $\text{COMP-VALID}_2$, which is a generalization of the problem $\text{COMP-SAT}$ (Lukasiewicz and Malizia 2016): given two sets $A$ and $B$ of quantified Boolean formulas characterized by 2 alternating quantifiers, decide whether the number of valid formulas in $A$ is greater than the number of valid formulas in $B$. The $\Theta_3^p$-hardness of $\text{COMP-VALID}_2$ holds even if the following restrictions are imposed over the instances (Lukasiewicz and Malizia 2017):

- $|A| = |B|$;
- all formulas of $A$ and $B$ are of the kind $(\forall X)(\exists Y)\phi(X, Y)$, where $\phi(X, Y)$ is a non-quantified 3CNF formula;
- all formulas of $A$ and $B$ have the same number of clauses in the non-quantified part;
- all formulas of $A$ and $B$ have the same sets of universally quantified variables and existentially quantified variables;
- sets $A = \{\Phi_1, \ldots, \Phi_n\}$ and $B = \{\Psi_1, \ldots, \Psi_m\}$ are such that $\Phi_{n+1}$ (resp., $\Psi_{m+1}$) being valid implies $\Phi_u$ (resp., $\Psi_u$) being valid as well, for any $u$ (intuitively, all valid formulas have the lowest indices in sets $A$ and $B$).

From the last assumption it follows that $(A, B)$ is a ‘yes’-instance of $\text{COMP-VALID}_2$ iff there is an index $u$ such that $\Phi_u \in A$ is valid and $\Psi_u \in B$ is not valid.

Given the restrictions listed above, we can assume, without simplifying the computational complexity of the problem, that all formulas of $A$ and $B$ are $\text{NQB}_F_2^V$ ones (Greco et al. 2009; 2011; Schaefer 2001). $\text{NQB}_F_2^V$ formulas are quantified Boolean formulas of the kind $\Phi = (\forall X)(\exists Y)\phi(X, Y)$, in which $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ are two disjoint sets of Boolean variables, and $\phi(X, Y) = c_{i(1)} \wedge c_{i(2)} \wedge \cdots \wedge c_{i(n)} \wedge c_{i(n)} \wedge c_{i(1)} \wedge \cdots \wedge c_{i(m)}$ is a 3CNF formula, where each universally quantified variable $x_k \in X$ occurs only in the two clauses $c_{i(k)} = (x_k \lor \neg y_k)$ and $c_{i(k)} = (\neg x_k \lor y_k)$—intuitively, each variable $x_k$ enforces the truth value of a corresponding variable $y_k$, which thus plays its role in the formula $\phi(X, Y)$. Indeed, a formula $(\forall X)(\exists Y)c_1 \wedge \cdots \wedge c_m$ can be replaced by $(\forall X)(\exists Y)c_1 \wedge c_2 \wedge \cdots \wedge c_m$.

where $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1^2, \ldots, y_m^2\}$ and each $c'_i$ is obtained by replacing every $x_j \in X$ by $y_j^2 \in Y$ in $c_i$. We will refer to the clauses $c_1, \ldots, c_m$ of an $\text{NQB}_F_2^V$ formula as the main clauses of the formula.

Let $I = (A, B)$ be an instance of $\text{COMP-VALID}_2$ satisfying all the restrictions above. We denote by $X$ and $Y$ the sets of the universally and existentially quantified variables, respectively. From $I$, we build the knowledge base $KB_CV(I) = (D_CV, \Sigma_CV)$ and query $QCV(I)$ as follows. In what follows, the first two arguments of each fact of $D_CV$ specify the identifier of the set (i.e., whether the formula belongs to set $A$ or $B$) and the numeric identifier of the formula in the set, respectively. In this way, we can discriminate facts referring to the different formulas in the two sets.

For each variable $x_i \in X$, in $D_CV$ there are facts:

$$\text{Val}(s, u, x_i, t, k) \quad \text{Val}(s, u, x_i, f, k)$$

where $s \in \{a, b\}$ is a constant representing sets $A$ and $B$, respectively, $u \in \{1, \ldots, v\}$ is a numeric constant representing the index of the formula in the sets, $x_i$ is a constant representing the respective variable, $t$ and $f$ are constants representing Boolean values $true$ and $false$, respectively, and $k \in \{1, 2, 3\}$ is a numeric constant (allowing us to have three copies of the facts).

There are facts $D_CV$ that will be used to impose the consistency of the assignments to the literals:

$$\text{SimLit}(s, u, t, k) \quad \text{OppLit}(s, u, t, k)$$

where $s \in \{a, b\}$, $u \in \{1, \ldots, v\}$, $t$, $f$, and $k \in \{1, 2, 3\}$, are constants with the same meaning as above.

There are facts in $D_CV$ that will be used to select possible ways of satisfying clauses in the formulas:

$$\text{ClSat}(s, u, t, f, k) \quad \text{ClSat}(s, u, t, k)$$

where $s \in \{a, b\}$, $u \in \{1, \ldots, v\}$, $t$, $f$, and $k \in \{1, 2, 3\}$, are constants with the same meaning as above.

Finally, $D_CV$ also contains facts that will be used to signal the validity or the non-validity of the formulas:

$$\text{NonValid}(s, u, k) \quad \text{Valid}(s, u)$$

where $s \in \{a, b\}$, $u \in \{1, \ldots, v\}$, and $k \in \{1, 2\}$, are constants with the same meaning as above. Note that we have two copies for facts NonValid, and one copy for facts Valid.

In the program $S_CV$, there is no TGD, and there are the following NCs. For notational convenience, in the NCs below, we will use the underscore ‘_’ as a placeholder for a fresh variable not appearing anywhere else in the NC.

The first negative constraint is

$$\text{Val}(S, U, X, t, _) \quad \text{Val}(S, U, X, f, _) \rightarrow \bot.$$
For simplicity in the exposition, let us assume that we are considering the formula \( \Phi_1 = (\forall X)(\exists Y)\phi_1(X,Y) \) belonging to set \( A \). The following construction for the NC can be generalized to any formula of \( A \) or \( B \). Since this NC is intricate, we look at its various constituent pieces.

A first piece of the NC checks that at least one of the copies of the structural facts is in the selection:

\[
\text{Config}(a,1) \equiv \bigwedge_{p(a,1,c,\_)} p(a,1,c,\_).
\]

By using a fresh variable as the last argument of all the predicates in Config, the presence of just one copy of each structural fact is enough to support the activation of Config.

A second piece of the NC “reads” the assignment on the variables in \( X \) encoded in the selection of the database:

\[
\text{Assign}(a,1,1) \equiv \bigwedge_{i=1}^n \text{Val}(a,1,x_i,T_i,\_).
\]

Below we use the following notation: \( l_{j,k} \) is the \( k \)th literal in the \( j \)th main clause, \( c_j \), and \( v_{j,k} \) is the variable of \( l_{j,k} \).

A third piece of the NC aims at “copying” the assignment on the variables in \( X \) onto the associated variables in \( Y \):

\[
\text{Copy}(a,1) \equiv \bigwedge_{i=1}^n \text{SimLit}(a,1,T_i,T_{j,k},\_),
\]

where \( T_{j,k} \) is a variable for the Boolean value of the literal \( l_{j,k} \) for which \( v_{j,k} = y_i \) in \( \phi_1(X,Y) \). Observe that, in order for Copy to work properly, each variable \( y_i \) must appear as a positive literal in one of the main clauses of \( \phi_1(X,Y) \) at least once. This can be assumed without loss of generality, because if \( y_i \) always appears as a negative literal in all the main clauses of \( \phi_1(X,Y) \), then we can replace all the occurrences of the negative literal \( \neg y_i \) with the positive literal \( y_i \) without altering the satisfiability properties of \( \phi_1(X,Y) \).

A fourth piece of the NC forces that the facts selected to simulate the assignment on the variables in \( Y \) are consistent. In the notation below, \( \ell_{j,k} \sim \ell_{j',k'} \) means that literals \( \ell_{j,k} \) and \( \ell_{j',k'} \) are both positive or negative, while \( \ell_{j,k} \not\sim \ell_{j',k'} \) means that one literal is positive and the other is negative.

\[
\text{Consist}(a,1) \equiv \bigwedge_{\forall (\ell_{j,k},\ell_{j',k'},\_)} \bigwedge_{\ell_{j,k} \sim \ell_{j',k'}} \text{SimLit}(a,1,T_{j,k},T_{j',k'},\_)
\]

\[
\bigwedge_{\forall (\ell_{j,k},\ell_{j',k'},\_)} \bigwedge_{\ell_{j,k} \not\sim \ell_{j',k'}} \text{OppLit}(a,1,T_{j,k},T_{j',k'},\_),
\]

where \( T_{j,k} \) is a variable with the same meaning as above.

The last piece of the NC checks \( \phi_1(X,Y) \)’s satisfiability:

\[
\text{Satisfied}(a,1) \equiv \bigwedge_{j=1}^n \text{Clsat}(a,1,T_{j,1},T_{j,2},T_{j,3},\_),
\]

these predicates are only for the main clauses of \( \phi_1(X,Y) \).

To conclude, the NC associated with \( \Phi_1 \in A \) is:

\[
\text{Config}(a,1), \text{Assign}(a,1), \text{Copy}(a,1), \\
\text{Consist}(a,1), \text{Satisfied}(a,1), \text{NonValid}(a,1,\_) \rightarrow \bot.
\]

The last NC is: \( \text{NonValid}(S,U,\_), \text{Valid}(S,U) \rightarrow \bot \).

The query is:

\[
q_{CV}(\mathcal{I}) = (\exists U)(\text{Valid}(a,1), \text{NonValid}(b,U,\_)).
\]

\( KB_{CV}(\mathcal{I}) \) has no TGDs, and bounded arity predicates.

It can be shown that \( \mathcal{C} \) is a ‘yes’-instance of COMP-VALIDiff \( KB_{CV}(\mathcal{I}) \) entails \( q_{CV}(\mathcal{I}) \) under \( \leq \mathcal{C} \) semantics, for any \( C \in \{AR, IAR, ICR\} \).

The following result shows that \( \leq \mathcal{A}-BCQ \leq \mathcal{IAR}-BCQ \), and \( \leq \mathcal{ICR}-BCQ \) answering for \( \mathcal{A} \) are \( p^{\text{NEXP}} \)-hard in the \( \text{ba-combined complexity} \), proving all \( p^{\text{NEXP}} \)-hardness results in Tables 2 and 3, including those for the more general combined complexity.

**Theorem 11.** For any \( C \in \{AR, IAR, ICR\} \), \( \leq \mathcal{C} \) BCQ answering for \( \mathcal{A} \) are \( p^{\text{NEXP}} \)-hard in the \( \text{ba-combined and combined complexity} \).

**Proof sketch.** Intuitively, the reduction for the \( p^{\text{NEXP}} \)-hardness proof in (Eiter, Lukasiewicz, and Predoiu 2016) for \( \leq \mathcal{A}-BCQ \) answering for \( \mathcal{A} \) in the \( \text{ba-complexity} \) is already a reduction for \( \leq \mathcal{A}-BCQ \) answering, as maximal-cardinality consistent database subsets there coincide with maximal-subset consistent database subsets. Furthermore, since the reduction uses a ground atomic query, this also shows the \( p^{\text{NEXP}} \)-hardness of \( \leq \mathcal{IAR}-BCQ \) answering for \( \mathcal{A} \), as \( \leq \mathcal{IAR}-BCQ \) answering coincides with \( \leq \mathcal{A}-BCQ \) answering for ground queries. Finally, the reduction for the \( p^{\text{NEXP}} \)-hardness proof in (Eiter, Lukasiewicz, and Predoiu 2016) for \( \leq \mathcal{A}-BCQ \) answering for \( \mathcal{A} \) in the \( \text{ba-complexity} \) is turned into a \( p^{\text{NEXP}} \)-hardness proof for \( \leq \mathcal{IAR}-BCQ \) answering in this case. There, one encodes initial tiling assignments \( v_1(X_1), \ldots, v_n(X_n) \) and has a ground atomic query \( q \), which we now also include in the database along with a fresh ground atom \( q_n \) and the NC \( v_1(X_1) \land \ldots \land v_n(X_n) \land q \land q \rightarrow \bot \). This intuitively “forces” the atom \( q \) into the database.

The \( \Theta_2^p \)-hardness in the \( \text{fp-combined complexity of} \leq \mathcal{A}-BCQ \) and \( \leq \mathcal{IAR}-BCQ \) answering over all the languages considered follows from the \( \Theta_2^p \)-hardness in the data complexity of \( \leq \mathcal{IAR}-BCQ \) answering.

**Theorem 12.** \( \leq \mathcal{IAR} \) and \( \leq \mathcal{ICR} \) BCQ answering is \( \Theta_2^p \)-hard in the \( \text{fp-combined complexity} \) for \( \mathcal{L}_\bot, \mathcal{L}_\bot, \mathcal{A}_\bot, \mathcal{AF}_\bot, \mathcal{S}_\bot, \mathcal{SF}_\bot, \mathcal{G}_\bot, \mathcal{GF}_\bot, \mathcal{F}_\bot, \mathcal{WS}_\bot, \mathcal{A}_\bot \), and \( \mathcal{WA}_\bot \) knowledge bases.

**Summary and Outlook**

We have given a precise picture of the complexity of BCQ answering under different cardinality-maximal inconsistency-tolerant semantics, namely, the ABox repair, the intersection of repairs (IAR), and the intersection of closed repairs (ICR) semantics, for the most popular Datalog\(^*\) languages and complexity measures. Note that these complexity results can now also be used to derive further complexity
results for other Datalog$^\pm$ languages. For example, for shy Datalog$^\pm$ (Leone et al. 2012), by the results of this paper, it is immediate that BCQ answering is complete for $\Theta_k^P$ and EXP in the data and combined complexity, respectively.

Future research lines include considering other classes of existential rules and defining other semantics for inconsistency-tolerant ontological query answering. In particular, it would be interesting to explore whether there are data-tractable and/or even first-order rewritable other such semantics. Furthermore, a more fine-grained way to analyze the complexity of query answering would be a non-uniform approach, looking at the complexity of a single ontology or a single ontology-mediated query (see, e.g., (Bienvenu et al. 2014; Koutris and Suciu 2014; Hernich et al. 2017)).

Acknowledgments. This work was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, and by the EPSRC grants EP/R013667/1, EP/L012138/1, and EP/M025268/1.

References


Leone, N.; Manna, M.; Terracina, G.; and Veltri, P. 2012. Efficiently computable Datalog$^\pm$ programs. In Proc. KR.


