Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

Generalized Consistent Query Answering under Existential Rules

Thomas Eiter
Institut für Informationssysteme
TU Wien, Austria
eiter@kr.tuwien.ac.at

Thomas Lukasiewicz
Department of Computer Science
University of Oxford, UK
firstname.lastname@cs.ox.ac.uk

Livia Predoiu*
Department of Computer Science
University of Oxford, UK
firstname.lastname@cs.ox.ac.uk

Abstract

Previous work has proposed consistent query answering as a way to resolve inconsistencies in ontologies. In these approaches to consistent query answering, however, only inconsistencies due to errors in the underlying database are considered. In this paper, we additionally assume that ontological axioms may be erroneous, and that some database atoms and ontological axioms may not be removed to resolve inconsistencies. This problem is especially well-suited in debugging mappings between distributed ontologies. We define two different semantics, one where ontological axioms as a whole are ignored to resolve an inconsistency, and one where only some of their instances are ignored. We then give a precise picture of the complexity of consistent query answering under these two semantics when ontological axioms are encoded as different classes of existential rules. In the course of this, we also close two open complexity problems in standard consistent query answering under existential rules.

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. This may be due to automated procedures, such as the automatic generation of mappings in data integration. As standard ontology languages adhere to the classical first-order semantics, inconsistencies are logical contradictions, which imply everything ("ex falso quodlibet") and make the whole ontology useless for reasoning. This shows the urgent 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 2012a; Bienvenu and Rosati 2013).

The complexity of query answering under the AR semantics when the ontology is described using one of the central DLs is rather well understood. The data and combined complexity were studied by Rosati (2011) 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 2012a; 2013) and (Lukasiewicz et al. 2015), the data and different types of combined complexity, respectively, of the AR semantics have been studied for ontologies described via existential rules, i.e., formulas $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \exists \mathbf{Y}\, p(\mathbf{X}, \mathbf{Y})$, and negative constraints $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \bot$, where $\bot$ denotes the truth constant *false*.

This paper continues this line of research. We develop two more sophisticated inconsistency-tolerant semantics for ontological query answering and analyze their complexity. The main contributions of this paper are briefly as follows.

- We introduce two new inconsistency-tolerant semantics for answering Boolean CQs (BCQs) under existential rules, called the *GR* and the *LGR semantics*, which generalize standard consistent BCQ answering. In these semantics, in addition to database atoms, also rules and rule instances, respectively, may be removed to resolve inconsistencies, and some atoms and rules are assumed to be non-removable. These semantics are especially well-suited in debugging mappings between distributed ontologies or ontology-based databases in OBDA (to resolve errors in—often automatically generated—mapping rules).
- We give a precise picture of the complexity of consistent

*Alternate affiliation: Department of Computer Science, Otto-von-Guericke University, Magdeburg, Germany.

BCQ answering under the GR and the LGR semantics for different classes of existential rules and different types of complexities. The complexity of consistent BCQ answering under the GR semantics (Table 2) coincides with the one of standard consistent BCQ answering, while the complexity of consistent BCQ answering under the LGR semantics (Table 3) moves slightly higher in several cases.

- In the course of the above analysis, we also close two open complexity problems in standard consistent query answering under existential rules (Lukasiewicz et al. 2015): We show that standard consistent BCQ answering under acyclic existential rules is $\mathrm{P}^{\mathrm{NEXP}}$-complete both in the combined and ba-combined complexity (see Table 1).

Preliminaries

We now recall some basics on existential rules from the context of Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012).

General. We assume 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 $h\colon \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \to \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ 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 $\mathbf{X} \cup \mathbf{Y} \subseteq \mathbf{V}$, $\varphi(\mathbf{X})$ 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 $\mathbf{X} \subseteq \mathbf{V}$, $\varphi(\mathbf{X})$ 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. Finite sets of TGDs and NCs are also called *programs*, and TGDs are also called *existential rules*.

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. Following Vardi's taxonomy (1982), the *combined complexity* of BCQ 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. In the context of description logics (DLs), the combined complexity in fact refers 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, while the *data complexity* additionally assumes that the query is also fixed.

Complexity Classes. We now briefly recall the complexity classes that we encounter in our complexity results below. The complexity classes PSPACE (resp., EXP, 2EXP) contain all decision problems that can be solved in polynomial space (resp., exponential, double exponential time) on a deterministic Turing machine, while the complexity classes NP, NEXP, and N2EXP contain all decision problems that can be solved in polynomial, exponential, and double exponential time on a nondeterministic Turing machine, respectively, and coNP, coNEXP, and coN2EXP are their complementary classes, where "Yes" and "No" instances are interchanged. The class Σ_2^p is the class of problems that can be solved in nondeterministic polynomial time using an NP-oracle, and Π_2^p is the complement of Σ_2^p. The above complexity classes and their inclusion relationships (which are all currently believed to be strict) are shown below:

$$\begin{aligned}\mathrm{NP}, \mathrm{coNP} \subseteq \Sigma_2^p, \Pi_2^p \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP} \subseteq \mathrm{NEXP}, \mathrm{coNEXP} \\ \subseteq \mathrm{P}^{\mathrm{NEXP}} \subseteq 2\mathrm{EXP} \subseteq \mathrm{N}2\mathrm{EXP}, \mathrm{coN}2\mathrm{EXP}\,.\end{aligned}$$

Generalized Inconsistency-Tolerance

In classical BCQ 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

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

asserting that p is a professor, postdoc, and a researcher, and that moreover he is the leader of the research group g, and that p' is the leader of g'. Assume a 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 in such cases are not very meaningful. For this reason, several inconsistency-tolerant semantics have been proposed in the literature. One of the central and well-accepted one is the *ABox repair (AR) semantics* (Lembo et al. 2010), which is based on the key notion of a *repair*, which is a $\subseteq$-maximal consistent subset of the given database D. Hence, it is assumed that errors leading to inconsistencies are only contained in the data of the database, but not the set Σ of TGDs and NCs. Answers to CQs are then defined relative to all repairs of D and Σ.

GR Semantics. In the following, instead, we define generalized inconsistency semantics, where we also allow for errors in Σ, and for some elements of D and Σ to be without errors. In detail, we define the notion of *generalized repair (GR)* semantics, which is based on allowing also (i) to minimally remove TGDs from Σ, and (ii) to partition both D and Σ into a hard and a soft part of non-removable and removable elements, respectively. The so partitioned database (resp., program) is called *flexible database* (resp., *program*).

Definition 1. A *flexible database* is a pair (D_h, D_s) of two databases D_h and D_s, denoted *hard and soft database*, respectively, while a *flexible program* is a pair (Σ_h, Σ_s) consisting of a finite set Σ_h of TGDs and NCs and a finite set Σ_s of TGDs, denoted *hard and soft program*, respectively.

Example 2. Consider again D and Σ of Example 1. A flexible database (D_h, D_s) and program (Σ_h, Σ_s) is then defined by $D_s = \{Prof(p),\ leaderOf(p, g)\}$, $\Sigma_s = \{leaderOf(X, Y) \to Prof(X)\}$, $D_h = D \setminus D_s$, and $\Sigma_h = \Sigma \setminus \Sigma_s$. ∎

We define the notion of *generalized repair (GR)* for flexible databases under flexible programs as follows.

Definition 2. A *generalized repair* of a flexible database (D_h, D_s) and a flexible program (Σ_h, Σ_s) is a pair $((D_h, D'_s), (\Sigma_h, \Sigma'_s))$, where $D'_s \subseteq D_s$ and $\Sigma'_s \subseteq \Sigma_s$ such that (i) $mods(D_h \cup D'_s \cup \Sigma_h \cup \Sigma'_s) \neq \varnothing$, and (ii) there is no $e \in (D_s \cup \Sigma_s) \setminus (D'_s \cup \Sigma'_s)$ for which $mods(D_h \cup D'_s \cup \Sigma_h \cup \Sigma'_s \cup \{e\}) \neq \varnothing$. The set of all such repairs $((D_h, D'_s), (\Sigma_h, \Sigma'_s))$ is denoted $drep((D_h, D_s), (\Sigma_h, \Sigma_s))$.

Example 3. Consider again the flexible database (D_h, D_s) and program (Σ_h, Σ_s) of Example 2. There are two repairs $((D_h, D'_s), (\Sigma_h, \Sigma'_s))$ and $((D_h, D''_s), (\Sigma_h, \Sigma''_s))$:

$$\begin{array}{lcl@{\qquad}lcl} D'_s & = & \{leaderOf(p, g)\} & D''_s & = & \varnothing \\ \Sigma'_s & = & \varnothing & \Sigma''_s & = & \Sigma_s\,. \end{array}$$

In both, the atom $Prof(p)$ is removed; in the first one, also the rule $leaderOf(X, Y) \to Prof(X)$ is removed, while in the second one, the atom $leaderOf(p, g)$ is removed. ∎

Generalizing standard consistent query answering, also called the AR semantics (Lembo et al. 2010), the GR semantics is based on the idea that a BCQ should hold, if it can be inferred from every generalized repair.

Definition 3. A BCQ q is entailed by a flexible database (D_h, D_s) and program (Σ_h, Σ_s) under the *generalized repair (GR) semantics*, denoted $(D_h, D_s) \cup (\Sigma_h, \Sigma_s) \models_{GR} q$, if $D_h \cup D'_s \cup \Sigma_h \cup \Sigma'_s \models q$, for every $((D_h, D'_s), (\Sigma_h, \Sigma'_s)) \in drep((D_h, D_s), (\Sigma_h, \Sigma_s))$. We refer to (B)CQ answering under the GR semantics as *GR-(B)CQ answering*.

Example 4. Consider again the flexible database (D_h, D_s) and program (Σ_h, Σ_s) of the running example, and the CQs

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

The former asks whether a researcher exists, while the latter asks whether a researcher exists who is also a professor and a group leader. It is then easy to verify that q_1 holds in both repairs, while q_2 holds only in the first repair. Thus, q_1 is entailed under the GR semantics, while q_2 is not. ∎

LGR Semantics. We next introduce the notion of *local generalized repairs (LGRs)* as a second, more elaborate consistency-tolerant semantics, which is obtained by minimally removing database facts and only rule instances (but not whole rules). Here, for a set of TGDs Σ, we denote by $ground(\Sigma)$ the set of all ground instances of elements of Σ relative to $\mathbf{C} \cup \mathbf{N}$, i.e., all rules $h(\varphi(\mathbf{X}) \to p(\mathbf{X}, \mathbf{Y}))$ for a TGD $\forall \mathbf{X}\, \varphi(\mathbf{X}) \to \exists \mathbf{Y}\, p(\mathbf{X}, \mathbf{Y})$ in Σ and a homomorphism h to $\mathbf{C} \cup \mathbf{N}$ that maps each $Y \in \mathbf{Y}$ to a distinct null $h(Y)$ such that $h^{-1}(Y) = \{Y\}$. We say that two instances of a TGD σ are *isomorphic*, if they have the same body.

Definition 4. A *local generalized repair* of a flexible database (D_h, D_s) and program (Σ_h, Σ_s) is a pair $((D_h, D'_s), (\Sigma_h, \Sigma'_s))$, where $D'_s \subseteq D_s$ and $\Sigma'_s \subseteq ground(\Sigma_s)$ such that (i) $mods(D_h \cup D'_s \cup \Sigma_h \cup \Sigma'_s) \neq \varnothing$, (ii) Σ'_s is *admissible*, i.e., no two rule instances $r_1 \in \Sigma'_s$ and $r_2 \in ground(\Sigma_s) \setminus \Sigma'_s$ of the same rule are isomorphic, and (iii) there is no strict superset $D''_s \cup \Sigma''_s \subseteq D_s \cup ground(\Sigma_s)$ of $D'_s \cup \Sigma'_s$ with (i) and (ii). We denote by $ldrep((D_h, D_s), (\Sigma_h, \Sigma_s))$ the set of all such repairs $((D_h, D'_s), (\Sigma_s, \Sigma'_s))$.

Example 5. Consider again the flexible database (D_h, D_s) and program (Σ_h, Σ_s) of Example 2. Under the LGR semantics, the first repair of Example 3 is refined to the repair $((D_h, D'_s), (\Sigma_h, \Sigma'_s))$ defined by $D'_s = \{leaderOf(p, g)\}$ and $\Sigma'_s = ground(\Sigma_s) \setminus \{leaderOf(p, g) \to Prof(p)\}$. ∎

We finally define BCQ answering under the LGR semantics, i.e., relative to all local generalized repairs.

Definition 5. A BCQ q is entailed by a flexible database (D_h, D_s) and program (Σ_h, Σ_s) under the *local generalized repair (LGR) semantics*, denoted $(D_h, D_s) \cup (\Sigma_h, \Sigma_s) \models_{LGR} q$, if $D_h \cup D'_s \cup \Sigma_h \cup \Sigma'_s \models q$, for every $((D_h, D'_s), (\Sigma_s, \Sigma'_s)) \in ldrep((D_h, D_s), (\Sigma_h, \Sigma_s))$. We refer to (B)CQ answering under the LGR semantics as *LGR-(B)CQ answering*.

Example 6. It is easy to verify that the query q_2 of Example 4 holds in both repairs of Example 5, as desired, as only the rule instance related to the inconsistency is removed. ∎

Applications. The new GR and LGR semantics for BCQ answering under existential rules are especially well-suited for debugging mappings between distributed ontologies (Meilicke, Stuckenschmidt, and Tamilin 2007).

Distributed ontologies (Borgida and Serafini 2003) are a framework for formalizing multiple ontologies that are pairwise linked by directed semantic mappings. In this context, a distributed ontology is a pair of ontologies $\mathbb{T} = (\mathcal{T}_i)_{i \in I}$ and associated mappings $\mathbb{M} = (\mathcal{M}_{i,j})_{i,j \in I, i \neq j}$, where I is an index set. Every $\mathcal{T}_i$ is an ontology, and thus contains definitions of concepts and properties, and axioms relating them. A concept C from $\mathcal{T}_i$ is also written as $i\colon C$. Every mapping $\mathcal{M}_{i,j}$ is a set of *bridge rules* that establishes semantic relations from $\mathcal{T}_i$ to $\mathcal{T}_j$, which allow a partial translation of $\mathcal{T}_i$'s language into the language of $\mathcal{T}_j$. For example, the (*into*) bridge rule $i\colon C \sqsubseteq j\colon D$ states that concept $i\colon C$ is, from $\mathcal{T}_j$'s point of view, less general than or as general as concept $j\colon D$. The analogous (*onto*) bridge rule $i\colon C \sqsupseteq j\colon D$ states that $i\colon C$ is more general than or as general as $j\colon D$.

Encoded as existential rules, each ontology is a program over pairwise disjoint schemas $\mathbf{S}_i$, while every mapping $\mathcal{M}_{i,j}$ is a finite set of existential rules connecting atoms over $\mathbf{S}_i$ to atoms over $\mathbf{S}_j$. Importantly, every ontology for itself is error-free, whereas the mappings between the ontologies may be erroneous (e.g., as they are automatically generated). Similarly, some (e.g., manually checked) parts of the underlying databases may be without errors, while other (e.g., automatically generated) parts may also contain errors. BCQ answering in this context can now exactly be formulated via the GR (resp., LGR) semantics. So, inconsistent distributed ontologies are repaired by removing a minimal set of database atoms and existential rules (resp., instances of existential rules) from the mappings.

Another important application is debugging ontologies that have been created in part manually (or checked manually, ensuring error-freeness) and in part enriched by automatically learned additional parts. The manually created part is modeled as the hard database and program, while the additionally learned part is the soft database and program.

Overview of Complexity Results

In the next section, we analyze the computational complexity of GR-BCQ and LGR-BCQ answering under the main decidable classes of TGDs, enriched with arbitrary NCs. We also close an open problem in standard consistent BCQ answering. We first briefly recall those classes and then give a brief overview of our complexity results. Here, we assume some elementary background in complexity theory; see (Johnson 1990; Papadimitriou 1994).

Decidability Paradigms. The main (syntactic) conditions on TGDs that guarantee the decidability of BCQ answering are guardedness (Calì, Gottlob, and Kifer 2013), stickiness (Calì, Gottlob, and Pieris 2012), and acyclicity. Interestingly, each such conditions has its "weak" 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 *guarded*, if there exists an atom $\mathbf{a} \in \mathit{body}(\sigma)$ that 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 that 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 *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 *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, $\mathsf{A} \subset \mathsf{WA}$. Observe also that $\mathsf{WA} \subset \mathsf{WS}$.

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 $\mathsf{F} \subset \mathsf{WA}$ and $\mathsf{F} \subset \mathsf{WG}$.

Refined Isomorphism of Rules in LGRs. For several classes of programs ($\mathsf{L}_\perp, \mathsf{G}_\perp, \mathsf{WG}_\perp, \mathsf{S}_\perp$, and $\mathsf{WS}_\perp$), we need to refine the notion of isomorphism between rules in LGRs.

Given a database D and a finite set of linear, guarded, or weakly guarded TGDs Σ, the *cloud* of an atom a over $\mathbf{C} \cup \mathbf{N}$, denoted $cld(a)$, is the set of all entailed atoms over constants in $D \cup \Sigma$ and arguments from a. Two pairs $(a, cld(a))$ and $(a', cld(a'))$ consisting of two atoms and their clouds relative to D and Σ are *isomorphic*, if there is an isomorphism between them, which maps (1) constants identically to themselves, and (2) nulls to nulls. Two instances σ and σ' of the same TGD are *isomorphic*, if their weak guards along with the clouds are isomorphic. In LGRs for $\mathsf{L}_\perp$, $\mathsf{G}_\perp$, and $\mathsf{WG}_\perp$, we assume that this generalized isomorphism (relative to $D = D_h \cup D'_s$ and $\Sigma = \Sigma_h \cup \Sigma'_s$) between rules is used in condition (ii) of Definition 4. Note that in all classes below $\mathsf{L}_\perp$, $\mathsf{G}_\perp$, and $\mathsf{WG}_\perp$, this new isomorphism naturally simplifies to the isomorphism of Definition 4.

As for $\mathsf{S}_\perp$ and $\mathsf{WS}_\perp$, we use the following isomorphism instead of the one in Definition 4. Two instances σ and σ' of the same TGD are *isomorphic*, if there is an isomorphism between their bodies, which maps (1) constants identically to themselves, (2) nulls in positions with finite rank in the underlying dependency graph identically to themselves, and (3) all the other nulls to nulls not in positions with finite rank in the underlying dependency graph. Note that, also here, in all classes below $\mathsf{S}_\perp$ and $\mathsf{WS}_\perp$, the new isomorphism naturally simplifies to the isomorphism of Definition 4.

Complexity Results. As a first important complexity result of this paper, we have determined the precise complexity of standard consistent BCQ answering under acyclic existential rules in the combined and the ba-combined case, which is complete for P^{NEXP}, closing two open problems from (Lukasiewicz et al. 2015); see the relative entries in Table 1. Furthermore, we provide a precise picture of the complexity of consistent BCQ answering from existential rules under the GR and the LGR semantics, which is compactly summarized in Tables 2 and 3, respectively; it ranges from coNP- to coN2EXP-completeness. More precisely, consistent BCQ answering under the GR semantics (see Table 2) is complete for coNP (resp., Π_2^p) in the data complexity and the *fp*-combined complexity for all fragments of existential rules, except for $\mathsf{WG}_\perp$, where it is complete for EXP. The combined complexity of consistent BCQ answering under the GR semantics is among PSPACE (for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, and $\mathsf{AF}_\perp$), EXP (for $\mathsf{F}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, and $\mathsf{GF}_\perp$), P^{NEXP} (for $\mathsf{A}_\perp$), and 2EXP (for $\mathsf{G}_\perp$, $\mathsf{WG}_\perp$, $\mathsf{WA}_\perp$, and $\mathsf{WS}_\perp$), while the ba-combined complexity is among Π_2^p (for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{F}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, and $\mathsf{GF}_\perp$), EXP (for $\mathsf{G}_\perp$ and $\mathsf{WG}_\perp$), P^{NEXP} (for $\mathsf{A}_\perp$), and 2EXP (for $\mathsf{WA}_\perp$ and $\mathsf{WS}_\perp$), Thus, the complexity of consistent BCQ answering under the GR semantics coincides with the complexity of standard consistent BCQ answering (see Table 1). The complexity of consistent BCQ answering under the LGR semantics (see Table 3), in contrast, moves slightly higher in several cases, namely, from PSPACE and EXP to coNEXP, and from P^{NEXP} and 2EXP to coN2EXP.

Derivation of Complexity Results

We now sketch some proofs of our complexity results; detailed proofs of all results will be given in an extended paper.

GR Semantics. We first describe the proofs of our complexity results for consistent BCQ answering under the GR semantics, which are derived from the complexity of standard consistent BCQ answering. Closing an open complexity problem in standard consistent BCQ answering, the following result first shows that standard consistent BCQ answering in the acyclic case is complete for P^{NEXP} in the combined and the ba-combined complexity.

Theorem 6. *Standard consistent query answering from databases D under acyclic programs Σ is complete for P^{NEXP} in the combined and the ba-combined complexity.*

Proof (sketch). Membership of this problem has been shown in (Lukasiewicz et al. 2015); it remains to prove the P^{NEXP}-hardness. To this end, we exhibit a P^{NEXP}-complete problem that can be conveniently reduced to consistent query answering. In particular, the following extended tiling problem serves this purpose (for tiling problems, cf. Theorem 15):

(ETP): Given a triple (m, TP_1, TP_2) of an integer m in tally and tiling problems TP_1 and TP_2 for the exponential square $2^n \times 2^n$, does, for every initial condition $w = w_0 \dots w_{m-1}$, either TP_1 have no solution with w, or does TP_2 have some solution with w?

(The initial condition w puts tiles w_j on positions $(j, 0)$, $0 \leq j < m$.) To encode this problem in LGR semantics, we show that any instance TP_i as above is reducible to BCQ answering from acyclic TGDs in polynomial time such that an atom $tiling_i$ is entailed by a theory $\Sigma^{TP_i,|w|} \cup D^{TP_i} \cup D^w$, where $\Sigma^{TP_i,|w|}$ is constructed from TP_i and $|w|$, D^{TP_i} from TP_i, and $D^w = \{init_j(w_j) \mid 0 \leq j \leq m\}$, iff TP_i has a solution with w. We then combine copies of the theories for TP_1 and TP_2 using auxiliary atoms into a theory $\Sigma \cup D$ such that, under LGR semantics a query atom q is entailed iff TP_1 has no solution with w, but TP_2 has one. Here, a constraint $tiling_1, p, p' \rightarrow \perp$ will effect that p and p' are both true in all repairs iff TP_1 has no solution for w. TGDs $p, p' \rightarrow q;\ tiling_2 \rightarrow q$ will then define the query atom q.

This construction works for a *fixed* w; in a final step, all initial conditions w of length m are created in different repairs, by (roughly) adding all possible initialization facts for $init_j(t)$, $0 \leq j < m$, and all tiles t to the database D, and setting up constraints $init_i(t), init_i(t') \rightarrow \perp$ for every such i and distinct t, t'; these constraints will enforce that at most one fact $init_i(t)$ for every i will be in a repair. □

The next result shows that all hardness results for standard consistent BCQ answering under the different classes of existential rules carry over to GR-BCQ answering.

Theorem 7. *If consistent query answering from databases under programs over some Datalog$^\pm$ language L is $\mathcal{C}$-*

	Data	Comb.	ba-comb.	fp-comb.
$\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$	coNP	PSPACE	Π_2^p	Π_2^p
$\mathsf{G}_\perp$	coNP	2EXP	EXP	Π_2^p
$\mathsf{WG}_\perp$	EXP	2EXP	EXP	EXP
$\mathsf{F}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, $\mathsf{GF}_\perp$	coNP	EXP	Π_2^p	Π_2^p
$\mathsf{A}_\perp$	coNP	$\text{P}^{\text{NEXP}\star}$	$\text{P}^{\text{NEXP}\star}$	Π_2^p
$\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$	coNP	2EXP	2EXP	Π_2^p

Table 1: Complexity of standard consistent BCQ answering under existential rules; all entries are completeness results; those marked with "$\star$" are novel results (proved in this paper), while the others are known (Lukasiewicz et al. 2015).

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

Table 2: Complexity of GR-BCQ answering under existential rules; all entries are completeness results.

	Data	Comb.	ba-comb.	fp-comb.
$\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$	coNP	coNEXP	Π_2^p	Π_2^p
$\mathsf{G}_\perp$	coNP	coN2EXP	coNEXP	Π_2^p
$\mathsf{WG}_\perp$	coNEXP	coN2EXP	coNEXP	coNEXP
$\mathsf{F}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, $\mathsf{GF}_\perp$	coNP	coNEXP	Π_2^p	Π_2^p
$\mathsf{A}_\perp$, $\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$	coNP	coN2EXP	coN2EXP	Π_2^p

Table 3: Complexity of LGR-BCQ answering under existential rules; all entries are completeness results.

hard in the data, combined, ba-combined, and fp-combined complexity, then consistent query answering from flexible databases under flexible programs over L under the GR semantics is also $\mathcal{C}$-hard in the data, combined, ba-combined, and fp-combined complexity, respectively.

Proof. Consistent query answering from databases D under programs Σ over L coincides with consistent query answering from the flexible databases $(\varnothing, D)$ under flexible programs $(\Sigma, \varnothing)$ over L. As the former is assumed to be $\mathcal{C}$-hard (in the data, combined, and *ba*- and *fp*-combined complexity), also the latter is $\mathcal{C}$-hard (in the data, combined, and *ba*- and *fp*-combined complexity, respectively). $\Box$

As for the upper complexity bounds, it is easy to verify that all upper complexity bounds for standard consistent BCQ answering from databases D under Datalog$^{\pm}$ programs Σ carry over to consistent GR-BCQ answering from databases (D_h, D_s) under Datalog$^{\pm}$ programs $(\Sigma_h, \varnothing)$. Based on this, the next result shows that also all membership results for standard consistent BCQ answering under existential rules carry over to GR-BCQ answering, as long as the existential rules are closed under adding 0-ary body atoms.

Theorem 8. *Let L be a Datalog$^{\pm}$ language that is closed under adding 0-ary atoms to rule bodies. If consistent query answering from databases under programs over L is in $\mathcal{C}$ in the data, combined, ba-combined, and fp-combined complexity, then consistent query answering from flexible databases under flexible programs over L under the GR semantics is also in $\mathcal{C}$ in the data, combined, ba-combined, and fp-combined complexity, respectively.*

Proof. Let (D_h, D_s) be a flexible database under a flexible program (Σ_h, Σ_s). We then construct a program Σ'_h as the set of (1.i) all TGDs in Σ_h, and (1.ii) all TGDs $\phi \wedge p_r() \rightarrow \psi$ and all NCs $\phi \wedge p_c() \rightarrow \bot$, for every TGD $r\colon \phi \rightarrow \psi$ and every NC $c\colon \phi \rightarrow \bot$ in Σ_s, respectively, where $p_r()$ and $p_c()$ are fresh 0-ary predicate symbols, one for every TGD and NC in Σ_s. Furthermore, we construct a database D'_s as the set of (2.i) all atoms in D_s and (2.ii) all fresh 0-ary predicate symbols from (1.ii). Then, consistent query answering from (D_h, D_s) under (Σ_h, Σ_s) coincides with consistent query answering from (D_h, D'_s) under $(\Sigma'_h, \varnothing)$. If the latter is in $\mathcal{C}$ in the data, combined, *ba*-combined, and *fp*-combined complexity, then also the former is in $\mathcal{C}$ in the data, combined, *ba*-combined, and *fp*-combined complexity, respectively. $\Box$

By the complexity of consistent query answering shown in Table 1 (which includes the new results of Theorem 6), we immediately obtain the following result for all cases except for $\mathsf{L}_\bot$ and $\mathsf{LF}_\bot$. As for $\mathsf{L}_\bot$ (and thus also $\mathsf{LF}_\bot$), it is not hard to derive the upper bounds by genuine proofs (rules in the repair can be polynomially guessed, like data).

Corollary 9. *Consistent query answering under the GR semantics from flexible databases under flexible programs over the Datalog$^{\pm}$ languages L in Table 2 is complete for the complexity classes shown there in the data, combined, ba-combined, and fp-combined complexity, respectively.*

LGR Semantics. We next focus on the complexity of consistent BCQ answering under the LGR semantics.

The following shows that the entries in Table 3 for $\mathsf{G}_\bot$ and $\mathsf{WS}_\bot$ in the data and the *fp*-combined complexity are upper complexity bounds for LGR-BCQ answering in these cases.

Theorem 10. *LGR-BCQ answering for $\mathsf{G}_\bot$ and $\mathsf{WS}_\bot$ is in* conp *(resp., Π^p_2) in the data (resp., fp-combined) complexity.*

Proof. Let (D_h, D_s) be a flexible database, let (Σ_h, Σ_s) be a flexible program in $\mathsf{G}_\bot$ or $\mathsf{WS}_\bot$, and let q be a BCQ.

As for $\mathsf{G}_\bot$, to decide the complementary problem, we guess a subset D'_s of D_s and a subset Σ'_s of the set of all ground instances of elements of Σ_s (relative to the constants and nulls in the finite part of the guarded chase forest of (D_h, D_s) under (Σ_h, Σ_s) necessary for evaluating q and all NCs in Σ_s). As (Σ_h, Σ_s) is fixed, the guess has a polynomial size and is thus in NP. We then check that the guess of Σ'_s is admissible, that (D_h, D'_s) is consistent under (Σ_h, Σ'_s), that they are maximal, and that q evaluates to false, which can all be done in deterministic (resp., nondeterministic) polynomial time in the data (resp., *fp*-combined) complexity. Overall, LGR-BCQ answering for $\mathsf{G}_\bot$ is in coNP (resp., Π^p_2) in the data (resp., *fp*-combined) complexity.

As for $\mathsf{WS}_\bot$, we reduce (Σ_h, Σ_s) to a flexible program (Σ'_h, Σ'_s) in $\mathsf{S}_\bot$ by replacing all multiple occurrences of variables with bounded domains in rule bodies by constants and nulls (encoded via Skolem terms). As (Σ_h, Σ_s) is fixed, the number of all such constants and nulls is polynomial, and also (Σ'_h, Σ'_s) is polynomial. As (Σ'_h, Σ'_s) is sticky, the polynomial witness property holds. To decide the complementary problem, we guess a subset D'_s of D_s and a subset Σ''_s of the set of all ground instances of elements of Σ'_s (relative to the constants and nulls in the finite part of the chase of (D_h, D_s) under (Σ'_h, Σ'_s) necessary for evaluating q and all NCs in Σ'_s). This guess has a polynomial size and is thus in NP. We then check that the guess of Σ''_s is admissible, that (D_h, D'_s) is consistent under (Σ_h, Σ''_s), that they are maximal, and that q evaluates to false, which can all be done in deterministic (resp., nondeterministic) polynomial time in the data (resp., *fp*-combined) complexity. Overall, LGR-BCQ answering for $\mathsf{WS}_\bot$ is in coNP (resp., Π^p_2) in the data (resp., *fp*-combined) complexity. $\Box$

As standard consistent BCQ answering is a special case of LGR-BCQ answering, the lower complexity bounds of the former (see Table 1) are also lower complexity bounds of the latter. As (1) $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, $\mathsf{F}_\bot$, $\mathsf{S}_\bot$, $\mathsf{SF}_\bot$, $\mathsf{GF}_\bot$, $\mathsf{A}_\bot$, and $\mathsf{WA}_\bot$ are special cases of $\mathsf{WS}_\bot$, and (2) $\mathsf{L}_\bot$ is a special case of $\mathsf{G}_\bot$, the upper complexity bounds of the latter are also upper complexity bounds of the former. We thus immediately obtain the following corollary, proving the entries in Table 3 for $\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, $\mathsf{G}_\bot$, $\mathsf{F}_\bot$, $\mathsf{S}_\bot$, $\mathsf{SF}_\bot$, $\mathsf{GF}_\bot$, $\mathsf{A}_\bot$, $\mathsf{WS}_\bot$, and $\mathsf{WA}_\bot$ in the data and the *fp*-combined complexity.

Corollary 11. *LGR-BCQ answering for $\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, $\mathsf{G}_\bot$, $\mathsf{F}_\bot$, $\mathsf{S}_\bot$, $\mathsf{SF}_\bot$, $\mathsf{GF}_\bot$, $\mathsf{A}_\bot$, $\mathsf{WS}_\bot$, and $\mathsf{WA}_\bot$ is complete for* coNP *(resp., Π^p_2) in the data (resp., fp-combined) complexity.*

The next result shows that the entries in Table 3 for $\mathsf{L}_\bot$, $\mathsf{F}_\bot$, and $\mathsf{S}_\bot$ in the *ba*-combined complexity are upper complexity bounds for LGR-BCQ answering in these cases.

Theorem 12. *LGR-BCQ answering for $\mathsf{L}_\bot$, $\mathsf{F}_\bot$, and $\mathsf{S}_\bot$ is in Π^p_2 in the ba-combined complexity.*

Proof (sketch). As for $\mathsf{L}_\perp$, to decide the complementary problem, we guess a subset D'_s of D_s and a subset Σ'_s of the set of all ground instances of elements of Σ_s (relative to the constants in (D_h, D_s) and w different nulls, where w is the maximal arity of a predicate symbol). As we are in the bounded arity case, the guess has a polynomial size and is thus in NP. We then check that the guess of Σ'_s is admissible, that (D_h, D'_s) is consistent under (Σ_h, Σ'_s), that they are maximal, and that q evaluates to false, which all is in NP in the ba-combined complexity. Overall, LGR-BCQ answering for $\mathsf{L}_\perp$ is in Π_2^p in the ba-combined complexity.

The proof for $\mathsf{F}_\perp$ is technically involved; here, we only sketch the main ideas behind it. We first guess and verify with an NP oracle in polynomial time a set of ground atoms S that represents a maximal repair candidate, which is a repair that includes all ground instances of TGDs satisfied in the entailed set of ground atoms. We then check that S does not satisfy the (w.l.o.g. atomic) query, which can be done in polynomial time, and we check that the guessed maximal repair candidate is actually maximal, which can be done with an NP oracle in polynomial time. Overall, the complementary problem is in Σ_2^p, and thus the problem is in Π_2^p. □

Since standard consistent BCQ answering is a special case of LGR-BCQ answering, the lower complexity bounds of the former (see Table 1) also apply to the latter. Since $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{SF}_\perp$, and $\mathsf{GF}_\perp$ are special cases of $\mathsf{F}_\perp$, the upper complexity bounds of the latter also apply to the former. We thus immediately obtain the following result, proving the entries in Table 3 for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{F}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, and $\mathsf{GF}_\perp$ in the ba-combined complexity.

Corollary 13. *LGR-BCQ answering for* $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{F}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, *and* $\mathsf{GF}_\perp$ *is complete for* Π_2^p *in the* ba*-combined complexity.*

The following theorem shows that the entries in Table 3 for $\mathsf{L}_\perp$, $\mathsf{F}_\perp$, and $\mathsf{S}_\perp$ in the combined complexity are upper complexity bounds for LGR-BCQ answering in these cases.

Theorem 14. *LGR-BCQ answering for* $\mathsf{L}_\perp$, $\mathsf{F}_\perp$, *and* $\mathsf{S}_\perp$ *is in* coNEXP *in the combined complexity.*

Proof (sketch). As for $\mathsf{L}_\perp$, to decide the complementary problem, we guess a subset D'_s of D_s and a subset Σ'_s of the set of all ground instances of elements of Σ_s (relative to the constants in (D_h, D_s) and w different nulls, where w is the maximal arity of a predicate symbol). The guess has an exponential size and is thus in NEXP. We then check that the guess of Σ'_s is admissible, that (D_h, D'_s) is consistent under (Σ_h, Σ'_s), that maximality holds, and that q evaluates to false, which all is in EXP in the combined complexity. Overall, LGR-BCQ answering for $\mathsf{L}_\perp$ is in coNEXP in the combined complexity.

As for $\mathsf{F}_\perp$, to decide the complementary problem, we guess a subset D'_s of D_s and a subset Σ'_s of the set of all ground instances of elements of Σ_s (relative to the constants in (D_h, D_s)). Since the set of all such ground instances is exponential, the guess is in NEXP. We then check that (D_h, D'_s) is consistent under (Σ_h, Σ'_s), that we have maximality, and that q evaluates to false, which all is in EXP in the combined complexity. Overall, LGR-BCQ answering for $\mathsf{F}_\perp$ is in coNEXP in the combined complexity.

As for $\mathsf{S}_\perp$, we only sketch the main ideas of showing NEXP membership of the complementary problem. We use the EXP alternating Turing machine encoding of BCQ query answering in the sticky case of (Calì, Gottlob, and Pieris 2012). In the sticky case, we have no nulls in positions with finite rank in the underlying dependency graph. Thus, we guess a subset of exponentially many pairwise non-isomorphic rule instances, and check via the EXP alternating algorithm that we actually have consistency and maximality, and that the repair does not satisfy the given BCQ. □

The next theorem shows that the entries in Table 3 for $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, and $\mathsf{SF}_\perp$ in the combined complexity are lower complexity bounds for LGR-BCQ answering in these cases.

Theorem 15. *LGR-BCQ answering for* $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, *and* $\mathsf{SF}_\perp$ *is* coNEXP*-hard in the combined complexity.*

Proof (sketch). We provide a polynomial reduction from the NEXP-hard *tiling problem* (Fürer 1983): Let $T = \{t_0, \ldots, t_k\}$ be a set of square tile types, $H, V \subseteq T \times T$ be the horizontal and vertical compatibility relations, respectively, and n be an integer in unary. A $2^n \times 2^n$ tiling is a function $f\colon \{1, \ldots, 2^n\} \times \{1, \ldots, 2^n\} \to T$ such that (i) $f(1,1) = t_0$, and (ii) $(f(i,j), f(i,j+1)) \in H$ and $(f(i,j), f(i+1,j)) \in V$, for each i and j. An instance of the tiling problem is a tuple (T, H, V, n), and the question is whether a $2^n \times 2^n$ tiling exists.

Given an instance of the tiling problem (T, H, V, n), we first construct a flexible database (D_h, D_s), a flexible program (Σ_h, Σ_s) in $\mathsf{LF}_\perp$ (and $\mathsf{SF}_\perp$), and a BCQ q such that (D_h, D_s) and (Σ_h, Σ_s) do not entail q under the LGR semantics iff (T, H, V, n) is solvable. We define $D_s = \varnothing$ and $D_h = \{p(0, \ldots, 0)\}$, where p is a $2n$-ary predicate symbol. Furthermore, Σ_h is the set of all the following rules:

- For every $i \in \{1, \ldots, 2n\}$, we have the two rules:

  $p_{x_i=0}(\mathbf{x}) \to p_{x_i=1}(\mathbf{x})$ and $p_{x_i=1}(\mathbf{x}) \to p_{x_i=0}(\mathbf{x})$,

  where $p_{x_i=b}(\mathbf{x}) = p(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$.

- For every $i \in \{1, \ldots, n\}$, we have the rule:

  $p(x_1, \ldots, x_{i-1}, 0, 1, \ldots, 1, \mathbf{y}) \to$
  $succ_x(x_1, \ldots, x_{i-1}, 0, 1, \ldots, 1, \mathbf{y}, x_1, \ldots, x_{i-1}, 1, 0, \ldots, 0, \mathbf{y})$.

- For every $i \in \{n+1, \ldots, 2n\}$, we have the rule:

  $p(\mathbf{x}, y_1, \ldots, y_{i-1}, 0, 1, \ldots, 1) \to$
  $succ_y(\mathbf{x}, y_1, \ldots, y_{i-1}, 0, 1, \ldots, 1, \mathbf{x}, y_1, \ldots, y_{i-1}, 1, 0, \ldots, 0)$.

- For every $i, j \in \{1, \ldots, k\}$, we have the two rules:

  $succ_x(u, v) \to tsucc_x(t_i, u, t_j, v)$
  $succ_y(u, v) \to tsucc_y(t_i, u, t_j, v)$.

- For every $j, j' \in \{1, \ldots, k\}, j \neq j'$, we have the NC:

  $tp(t_j, u),\ tp(t_{j'}, u) \to \bot$.

- For every $i, j \in \{1, \ldots, k\}$ such that $(i, j) \notin H$, we have:

  $tsucc_x(t_i, u, t_j, v) \to bad$.

- For every $i, j \in \{1, \ldots, k\}$ such that $(i, j) \notin V$, we have:

  $tsucc_y(t_i, u, t_j, v) \to bad$.

For every $i, j \in \{1, \ldots, k\}$, Σ_s contains the four rules:

$tsucc_x(t_i, u, t_j, v) \rightarrow tp(t_i, u)$
$tsucc_x(t_i, u, t_j, v) \rightarrow tp(t_j, v)$
$tsucc_y(t_i, u, t_j, v) \rightarrow tp(t_i, u)$
$tsucc_y(t_i, u, t_j, v) \rightarrow tp(t_j, v)$.

It is then not difficult to verify that there exists a maximal consistent set that does not satisfy bad (i.e., bad is not LGR-entailed) iff the tiling problem is solvable. Furthermore, the flexible program (Σ_h, Σ_s) belongs to both $\mathsf{LF}_\perp$ and $\mathsf{SF}_\perp$.

As for $\mathsf{AF}_\perp$, we only sketch the main ideas behind the hardness proof: There exists a similar reduction from the tiling problem (T, H, V, n) to a flexible database (D_h, D_s) and program (Σ_h, Σ_s) in $\mathsf{AF}_\perp$, using multiple atoms in rule bodies, but no cyclic dependencies between predicates. ☐

As $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{SF}_\perp$, and $\mathsf{GF}_\perp$ are special cases of $\mathsf{F}_\perp$, the upper complexity bounds of the latter also apply to the former. As (1) $\mathsf{LF}_\perp$ is a special case of $\mathsf{L}_\perp$, $\mathsf{GF}_\perp$, and $\mathsf{F}_\perp$, and (2) $\mathsf{SF}_\perp$ is a special case of $\mathsf{S}_\perp$, the lower complexity bounds of the former also apply to the latter. We thus immediately obtain the following result as a corollary of Theorems 14 and 15, proving the entries in Table 3 for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{F}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, and $\mathsf{GF}_\perp$ in the combined complexity.

Corollary 16. *LGR-BCQ answering for* $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{F}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, *and* $\mathsf{GF}_\perp$ *is* coNEXP*-complete in the combined complexity.*

The following result shows that the entry in Table 3 for $\mathsf{WS}_\perp$ in the combined complexity is an upper complexity bound for LGR-BCQ answering in this case.

Theorem 17. *LGR-BCQ answering for* $\mathsf{WS}_\perp$ *is in* coN2EXP *in the combined complexity.*

Proof (sketch). We sketch the main ideas of proving N2EXP membership of the complementary problem. Similar to the proof of Theorem 14, we use the alternating Turing machine encoding of BCQ query answering of (Calì, Gottlob, and Pieris 2012). In the weakly sticky case, however, it is in 2EXP, and we have nulls in positions with finite rank in the underlying dependency graph. So, we now guess a subset of double exponentially many pairwise non-isomorphic rule instances, and check via the 2EXP alternating algorithm that we actually have consistency and maximality, and that the repair does not satisfy the given BCQ. ☐

The next result shows that the entry in Table 3 for $\mathsf{A}_\perp$ in the ba-combined complexity is a lower complexity bound for LGR-BCQ answering in this case.

Theorem 18. *LGR-BCQ answering for* $\mathsf{A}_\perp$ *is* coN2EXP*-hard in the* ba*-combined complexity.*

Proof (sketch). Hardness for coN2EXP is proved by a polynomial reduction from the following N2EXP-complete tiling problem (to the complementary problem): Given $T = \{t_0, \ldots, t_k\}$, $H, V \subseteq T \times T$, and $n \geq 0$, decide if a tiling of the $2^{(2^n)} \times 2^{(2^n)}$ square exists that satisfies H and V. ☐

As (1) $\mathsf{A}_\perp$ is a special case of $\mathsf{WA}_\perp$, and (2) $\mathsf{WA}_\perp$ is a special case of $\mathsf{WS}_\perp$, the upper complexity bounds of the latter also apply to the former, and the lower complexity bounds of the former also apply to the latter. We thus immediately obtain the following result as a corollary of Theorems 17 and 18, proving the entries in Table 3 for $\mathsf{A}_\perp$, $\mathsf{WA}_\perp$, and $\mathsf{WS}_\perp$ in the combined and the ba-combined complexity.

Corollary 19. *LGR-BCQ answering for* $\mathsf{A}_\perp$, $\mathsf{WA}_\perp$, *and* $\mathsf{WS}_\perp$ *is* coN2EXP*-complete in the combined and the ba-combined complexity.*

We next show that the entries in Table 3 for $\mathsf{WG}_\perp$ in the combined and the ba-combined complexity are upper complexity bounds for LGR-BCQ answering in these cases.

Theorem 20. *LGR-BCQ answering for* $\mathsf{WG}_\perp$ *is in* coN2EXP *(resp.,* coNEXP*) in the combined (resp., ba-combined) complexity.*

Proof (sketch). We refine the 2EXP alternating algorithm for BCQ answering in the weakly guarded case by Calì et al. (2013): we guess a subset of a double exponential number of pairwise non-isomorphic rule instances along with their weak guards and clouds, and then check via the 2EXP alternating algorithm that we actually have consistency and maximality, and that the repair does not satisfy the given BCQ. In the ba-combined case, the complexity of Calì et al.'s alternating algorithm drops to EXP, and the above refined alternating algorithm drops to NEXP. ☐

The following shows that the entries in Table 3 for $\mathsf{G}_\perp$ in the combined and the ba-combined complexity and for $\mathsf{WG}_\perp$ in the data complexity are lower complexity bounds for LGR-BCQ answering in these cases.

Theorem 21. *(a) LGR-BCQ answering for* $\mathsf{G}_\perp$ *is hard for* coN2EXP *(resp.,* coNEXP*) in the combined (resp., ba-combined) complexity. (b) LGR-BCQ answering for* $\mathsf{WG}_\perp$ *is* coNEXP*-hard in the data complexity.*

Proof (sketch). We show this by reductions from a novel coN2EXP- resp. coNEXP-complete problem that exploits results for BCQ answering from guarded resp. weakly guarded TGDs by Calì et al. (2013). They showed that the problem is 2EXP-complete in both cases, and proved 2EXP-hardness by an encoding of alternating Turing machines (ATMs) with exponential space; for bounded arities, they proved EXP-hardness by similar encodings of polynomial space ATMs. As well-known, 2EXP = AEXPSPACE and EXP = APSPACE (Chandra, Kozen, and Stockmeyer 1981). Furthermore, the encoding established that for weakly guarded TGDs, BCQ answering is EXP-complete even for a fixed program.

We introduce a generalization of ATMs that can be encoded into LGR-CQ answering, by suitably extending the encodings of Calì et al. with minor changes. These are ATMs M with two (possibly identical) transitions for each step, where each configuration c of M has a *branching instruction* $bi(c) \in \{1, 2\}$ associated. Besides existential and universal states also *branching states* exist, which intuitively tell M which nondeterministic move to make (branch 1 or 2) depending on the current configuration; note that existential states *depend only on the current state and symbol under the r/w- head*. A branching configuration c (i.e., with a branching state) accepts, if the successor configuration c_i such that $i = bi(c)$ accepts; M accepts an input I, if it accepts I for some bi assignment. As we show, the acceptance problem for such *branching ATMs* with polynomial (resp. exponential) workspace is complete for NEXP (resp., N2EXP). □

As (1) $\mathsf{G}_\perp$ is a special case of $\mathsf{WG}_\perp$, and (2) the data complexity is a special case of the *fp*-combined complexity, the upper complexity bounds of the latter also apply to the former, and the lower complexity bounds of the former also apply to the latter. Furthermore, LGR-BCQ answering in the *fp*-combined complexity can be reduced to an exponential number of LGR-BCQ answering problems in the ba-combined complexity (via all exponentially many possible instantiations of predicates with unbounded arity). We thus immediately obtain the following result as a corollary of Theorems 20 and 21, proving the entries in Table 3 for $\mathsf{G}_\perp$ and $\mathsf{WG}_\perp$ in the combined and the ba-combined complexity, and for $\mathsf{WG}_\perp$ in the data and the *fp*-combined complexity.

Corollary 22. *(a) LGR-BCQ answering for* $\mathsf{G}_\perp$ *is complete for* coN2EXP *(resp.,* coNEXP*) in the combined (resp., ba-combined) complexity. (b) LGR-BCQ answering for* $\mathsf{WG}_\perp$ *is complete for* coN2EXP *in the combined complexity and for* coNEXP *in the data and the* ba*- and fp-combined complexity.*

Related Work

Inconsistency handling has been widely studied in databases and in knowledge representation in the last three decades. Consistent query answering, first developed for relational databases (Arenas, Bertossi, and Chomicki 1999), and then generalized as the AR semantics for several description logics (DLs) (Lembo et al. 2010; 2015a), is the most widely accepted semantics for querying inconsistent ontologies. However, to our knowledge, no previous works in the literature introduced a generalized approach to consistent query answering under existential rules, where up to rule instances (rather than only database atoms) are removed to gain consistency, and analyzed the complexity of such an approach.

Chomicki (2007) addresses the basic concepts and results of consistent query answering for relational databases. More recent work in databases considers the data complexity of consistent query answering under local-as-view (LAV), global-as-view (GAV), and weakly acyclic existential rules (ten Cate, Fontaine, and Kolaitis 2012). In (ten Cate, Halpert, and Kolaitis 2014), very restricted forms of such rules are considered for inconsistency-tolerant query answering over a target database enriched by data transferred from a source database via mappings. Furthermore, target queries are rewritten as source queries over the source schema. That is, the repair rather happens in the source database than in the target database. For general existential rules, the authors show that an inconsistency-tolerant rewriting of unions of CQs is possible relative to the stable models of a disjunctive logic program over a suitable expansion of the source schema. Both recent works consider only inconsistencies caused by equality-generating dependencies.

Different approaches for inconsistency handling in various classes of DL ontologies are described in (Qi and Du 2009; Huang, van Harmelen, and ten Teije 2005; Ma and Hitzler 2009), ranging from revision for DL terminologies to different kinds of reasoning with inconsistent ontologies. In (Lembo et al. 2010; 2011; 2015a), adapting consistent query answering for inconsistent ontologies in the *DL-Lite* family is studied. Besides the AR semantics, three other inconsistency-tolerant query answering semantics are proposed: closed ABox repair (CAR) semantics, the approximations by the intersection AR (IAR) and the intersection CAR (ICAR) semantics. Moreover, first-order (FO) rewritability is investigated in (Lembo et al. 2011; 2015a), and query answering under IAR shown to be FO-rewritable for unions of CQs in the most expressive *DL-Lite* logic considered. FO-rewritability is also investigated by Bienvenu (2011). In (Bienvenu 2012), she analyzed the complexity of the AR and IAR semantics for a fragment of *DL-Lite*$_{\text{core}}$ and proposed an alternative approximate inconsistency-tolerant query answering semantics, viz. intersection of closed repairs (ICR). Rosati (2011) presented a computational analysis of instance checking and CQ answering under inconsistency-tolerant semantics for a range of DLs. Bienvenu and Rosati (2013) proposed two parameterized inconsistency-tolerant semantics for DLs, which approximate the AR and the IAR semantics, respectively.

Lukasiewicz, Martinez, and Simari (2012b) have shown the FO-rewritability under the AR semantics and FO-rewritable existential rules. In (Lukasiewicz, Martinez, and Simari 2013), the data complexity of inconsistency-tolerant query answering under the AR, IAR, and ICR semantics is studied, and in (Lukasiewicz et al. 2015), the combined, *ba*-combined, and *fp*-combined complexity is investigated, both for several languages of existential rules. Lukasiewicz, Martinez, and Simari (2012a) explored a general framework for inconsistency management under existential rules based on incision functions, while Bienvenu et al. (2014) studied the data and combined complexity of inconsistency-tolerant query answering under the AR and IAR semantics for different types of preferred repairs.

Closest in spirit to the idea of repairing rules (as a whole, but not rule instances) are current mapping repair applications for mappings between ontologies (Meilicke, Stuckenschmidt, and Tamilin 2007; Jiménez-Ruiz et al. 2013). However, only very simple mappings are considered in this context, namely concept to concept mappings, which correspond to full linear rules with unary predicates. The ontologies are assumed to be correct, and conflicting mappings are deleted as a whole, which is a special case of the GR semantics. Note that in applications for ontology debugging (Parsia, Sirin, and Kalyanpur 2005), only inconsistent concepts are repaired. Mappings in inconsistency-tolerant OBDA have been considered by Lembo et al. (2014; 2015b), where inconsistency caused by mappings and redundancy recognition of mappings are studied, but query answering is not considered, and also no repair is provided.

Less closely related approaches to repairing ontological axioms are error-tolerant reasoning (Ludwig and Peñaloza 2014), which repairs inferences identified as modeling errors, and preferential $\mathcal{ALC}$ (Deane, Broda, and Russo 2015).

Summary and Outlook

In this paper, we have introduced the GR and the LGR semantics as two new inconsistency-tolerant semantics for

BCQ answering from databases under existential rules. In these semantics, in addition to database atoms, also rules and rule instances, respectively, may be erroneous and thus removed to resolve inconsistencies, and some atoms and rules are assumed to be without errors and thus non-removable. These semantics are especially well-suited in debugging mappings between distributed ontologies. We have given a precise picture of the complexity of consistent BCQ answering under the GR and the LGR semantics for different classes of existential rules and different types of complexities. We have also closed two open complexity problems in standard consistent query answering under existential rules.

Topics for future research are to consider other classes of existential rules and to generalize 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 cases.

Acknowledgments

This work has received funding from the UK EPSRC grants EP/J008346/1, EP/L012138/1, and EP/M025268/1, and the EU FP7 Marie-Curie IE Fellowship PRODIMA.

References

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

Bienvenu, M., and Rosati, R. 2013. Tractable approximations of consistent query answering for robust ontology-based data access. In *Proc. IJCAI*, 775–781.

Bienvenu, M.; Bourgaux, C.; and Goasdoué, F. 2014. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *Proc. AAAI*, 996–1002.

Bienvenu, M. 2011. First-order expressibility results for queries over inconsistent *DL-Lite* knowledge bases. In *Proc. DL*.

Bienvenu, M. 2012. On the complexity of consistent query answering in the presence of simple ontologies. In *Proc. AAAI*, 705–711.

Borgida, A., Serafini, L. 2003. Distributed description logics: Assimilating information from peer sources. *J. Data Sem.* 1:153–184.

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 Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14:57–83.

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.; Kozen, D.; and Stockmeyer, L. J. 1981. Alternation. *J. ACM* 28(1):114–133.

Chomicki, J. 2007. Consistent query answering: Five easy pieces. In *Proc. ICDT*, 1–17.

Deane, G.; Broda, K.; and Russo, A. 2015. Reasoning in the presence of inconsistency through preferential $\mathcal{ALC}$. In *Proc. LPAR-20, Short Presentations*, 67–80.

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 *Proc. Logic and Machines*, 312–319.

Huang, Z.; van Harmelen, F.; and ten Teije, A. 2005. Reasoning with inconsistent ontologies. In *Proc. IJCAI*, 354–359.

Jiménez-Ruiz, E.; Meilicke, C.; Cuenca Grau, B.; and Horrocks, I. 2013. Evaluating mapping repair systems with large biomedical ontologies. In *Proc. DL*.

Johnson, D. S. 1990. A catalog of complexity classes. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science*, volume A. MIT Press. chapter 2, 67–161.

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

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2011. Query rewriting for inconsistent *DL-Lite* ontologies. In *Proc. RR*, 155–169.

Lembo, D.; Mora, J.; Rosati, R.; Savo, D. F.; and Thorstensen, E. 2014. Towards mapping analysis in ontology-based data access. In *Proc. RR*, 108–123.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2015a. Inconsistency-tolerant query answering in ontology-based data access. *J. Web Sem.* 33:3–29.

Lembo, D.; Mora, J.; Rosati, R.; Savo, D. F.; and Thorstensen, E. 2015b. Mapping analysis in ontology-based data access: Algorithms and complexity. In *Proc. ISWC*, 217–234.

Ludwig, M., and Peñaloza, R. 2014. Error-tolerant reasoning in the description logic $\mathcal{EL}$. In *Proc. JELIA*, 107–121.

Lukasiewicz, T.; Martinez, M. V.; Pieris, A.; and Simari, G. I. 2015. From classical to consistent query answering under existential rules. In *Proc. AAAI*, 1546–1552.

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

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2012b. Inconsistency-tolerant query rewriting for linear Datalog+/–. In *Proc. Datalog 2.0*, 123–134.

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

Ma, Y., and Hitzler, P. 2009. Paraconsistent reasoning for OWL 2. In *Proc. RR*, 197–211.

Meilicke, C.; Stuckenschmidt, H.; and Tamilin, A. 2007. Repairing ontology mappings. In *Proc. AAAI*, 1408–1413.

Papadimitriou, C. H. 1994. *Computational Complexity*.

Parsia, B.; Sirin, E.; and Kalyanpur, A. 2005. Debugging OWL ontologies. In *Proc. WWW*, 633–640.

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

Qi, G., and Du, J. 2009. Model-based revision operators for terminologies in description logics. In *Proc. IJCAI*, 891–897.

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

ten Cate, B.; Fontaine, G.; and Kolaitis, P. G. 2012. On the data complexity of consistent query answering. In *Proc. ICDT*, 22–33.

ten Cate, B.; Halpert, R. L.; Kolaitis, P. G. 2014. Exchange-repairs: Managing inconsistency in data exchange. In *Proc. RR*, 140–156.

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