Computing and Explaining Query Answers over Inconsistent DL-Lite Knowledge Bases

Meghyn Bienvenu, Camille Bourgaux, François Goasdoué

Journal of Artificial Intelligence Research 64 (2019) 563-644 Submitted 09/17; published 03/19

Computing and Explaining Query Answers over Inconsistent DL-Lite Knowledge Bases

Meghyn Bienvenu MEGHYN@LIRMM.FR
CNRS, Université Montpellier 2
Campus St Priest, 161 rue Ada
34090 Montpellier, France

Camille Bourgaux CAMILLE.BOURGAUX@TELECOM-PARISTECH.FR
Télécom ParisTech
46 rue Barrault
F-75634 Paris Cedex 13, France

François Goasdoué FG@IRISA.FR
Univ Rennes, CNRS, IRISA
6 rue de Kerampont, CS 80518,
22305 Lannion Cedex, France

Abstract

Several inconsistency-tolerant semantics have been introduced for querying inconsistent description logic knowledge bases. The first contribution of this paper is a practical approach for computing the query answers under three well-known such semantics, namely the AR, IAR and brave semantics, in the lightweight description logic DL-Lite$_\mathcal{R}$. We show that query answering under the intractable AR semantics can be performed efficiently by using IAR and brave semantics as tractable approximations and encoding the AR entailment problem as a propositional satisfiability (SAT) problem. The second issue tackled in this work is explaining why a tuple is a (non-)answer to a query under these semantics. We define explanations for positive and negative answers under the brave, AR and IAR semantics. We then study the computational properties of explanations in DL-Lite$_\mathcal{R}$. For each type of explanation, we analyze the data complexity of recognizing (preferred) explanations and deciding if a given assertion is relevant or necessary. We establish tight connections between intractable explanation problems and variants of SAT, enabling us to generate explanations by exploiting solvers for Boolean satisfaction and optimization problems. Finally, we empirically study the efficiency of our query answering and explanation framework using a benchmark we built upon the well-established LUBM benchmark.

1. Introduction

Description logic (DL) knowledge bases (KBs) consist of a TBox (ontology) that provides conceptual knowledge about the application domain and an ABox (dataset) that contains facts about particular entities (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003). The problem of querying such KBs using database-style queries (in particular, conjunctive queries, or CQs) has been a major focus of recent DL research. Since scalability is a key concern, much of the work has focused on lightweight DLs for which query answering can be performed in polynomial time w.r.t. the size of the ABox. The DL-Lite family of lightweight DLs (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2007) is especially

popular due to the fact that query answering can be reduced, via query rewriting, to the problem of standard database query evaluation. The importance of the DL-Lite family is witnessed by the inclusion of the OWL 2 QL profile (Motik, Cuenca Grau, Horrocks, Wu, Fokoue, & Lutz, 2012), based upon DL-Lite$_{\mathcal{R}}$, in the latest version of OWL (OWL Working Group, 2009), the W3C-standardized ontology language for the Semantic Web.

Since the TBox is usually developed by experts and subject to extensive debugging, it is often reasonable to assume that its contents are correct. By contrast, the ABox is typically substantially larger and subject to frequent modifications, making errors almost inevitable. As such errors may render the KB inconsistent, several inconsistency-tolerant semantics have been introduced to provide meaningful answers to queries posed over inconsistent KBs (see Bienvenu & Bourgaux, 2016, for a survey and Section 8.1 for further discussion and references). Arguably the most natural and well-known inconsistency-tolerant semantics is the *AR semantics* (AR for 'ABox Repair') (Lembo, Lenzerini, Rosati, Ruzzi, & Savo, 2010, 2015), inspired by work on consistent query answering in databases (initiated by Arenas, Bertossi, and Chomicki, 1999; see also the survey by Bertossi, 2011). Query answering under AR semantics amounts to considering those answers that can be obtained from every *repair*, the latter being defined as an inclusion-maximal subset of the ABox that is consistent with the TBox. The more cautious *IAR semantics* (IAR for 'Intersection of ABox Repairs') (Lembo et al., 2010, 2015) queries the intersection of the repairs and provides a lower bound on AR semantics. The *brave* semantics (Bienvenu & Rosati, 2013), which considers those answers holding in at least one repair, provides a natural upper bound.

The complexity of ontology-mediated query answering (OMQA) under inconsistency-tolerant semantics has been the subject of several papers (see e.g. Rosati, 2011; Bienvenu, 2012; Lukasiewicz, Martinez, & Simari, 2013). Not too surprisingly, given prior negative results on consistent query answering in databases, the AR semantics was shown to be coNP-hard in data complexity for DL-Lite ontologies (Lembo et al., 2010), and coNP-hardness has been shown to hold in even more restricted settings (Bienvenu, 2012), dashing hopes of taming intractability by limiting the expressivity of the ontology language. The IAR and brave semantics, by contrast, enjoy polynomial data complexity for DL-Lite ontologies (and more generally, all ontology languages that admit first-order query rewritings) (Lembo, Lenzerini, Rosati, Ruzzi, & Savo, 2011; Lembo et al., 2015; Bienvenu & Rosati, 2013).

As the complexity of inconsistency-tolerant query answering in the presence of ontologies is now well understood, attention has turned to the problem of implementing these alternative semantics. Most work has focused on the IAR semantics and DL-Lite ontology languages, due to the aforementioned tractability result. The QuID system (Rosati, Ruzzi, Graziosi, & Masotti, 2012) implements the IAR semantics, using either query rewriting or ABox cleaning, and the SaQAI system (Tsalapati, Stoilos, Stamou, & Koletsos, 2016) implements IAR using ABox cleaning and saturation, as well as the ICAR semantics (that corresponds to the IAR semantics over the saturated ABox, Lembo et al., 2010). For the natural but intractable AR semantics, some implementations of consistent query answering have been proposed in the database setting, but we are not aware of any system implementing the AR semantics for DL KBs. It was thus left open whether the IAR semantics constitutes a good approximation of the AR semantics, and whether one can devise practical querying algorithms for AR semantics despite the negative complexity results.

The first contribution of this paper is to develop and evaluate a practical method for CQ answering over inconsistent DL-Lite$_\mathcal{R}$ KBs under the AR semantics. Our approach first computes the answers under the simpler IAR and brave semantics, in order to obtain upper and lower bounds on the set of answers under AR semantics: answers holding under the stricter IAR semantics help us identify a portion of the AR-answers, and tuples that are not found to be brave-answers allow us to mark some tuples as non-answers under AR semantics[1]. To determine the status of the remaining tuples (i.e. tuples holding under brave semantics but not under IAR semantics), we provide an encoding in terms of propositional unsatisfiability, which can then be passed to an off-the-shelf SAT solver. In addition to using the IAR and brave semantics to reduce the number of calls to the SAT solver, we propose to use these three semantics to partition query answers into three classes of varying reliability: (Almost) Sure (those answers holding under IAR semantics), Likely (answers holding under AR semantics, but not IAR semantics), and Possible (answers holding only under brave semantics). Our approach has been implemented in the CQAPri system and evaluated over the benchmark we built starting from the well-known LUBM benchmark.[2] The principal conclusion of our experiments is that it is feasible to compute answers under the AR semantics, and this is due in large part to the fact that the IAR semantics is able to identify a large share of the AR-answers.

While efficiency is crucial when developing ontology-mediated query answering systems, it is also important to consider other aspects related to the usability of such systems. In particular, the need to equip reasoning systems with explanation services is widely acknowledged by the DL community (see Section 8 for discussion and references). The study of explanation services for DLs has thus far focused primarily on explaining entailed TBox axioms or ABox assertions, and the problem of explaining answers to CQs under the classical semantics for consistent KBs has been studied in only a few works, mostly for consistent KBs. A proof-theoretic approach to explaining positive answers to CQs over consistent DL-Lite$_\mathcal{A}$ KBs was introduced by Borgida, Calvanese, and Rodriguez-Muro (2008). It outputs a single proof, involving both TBox axioms and ABox assertions, that is generated by 'tracing back' the relevant part of the rewritten query, using minimality criteria to select a 'simplest' proof. For negative answers, explanations for why a tuple is not an answer to a CQ are defined by Calvanese, Ortiz, Simkus, and Stefanoni (2013) as sets of ABox assertions that can be added to the ABox to make the tuple become an answer. Practical algorithms and an implementation for computing such explanations were described by Du, Wang, and Shen (2014). The latter work was recently extended to the case of inconsistent KBs (Du, Wang, & Shen, 2015): essentially the idea is to add a set of ABox assertions that will lead to the answer holding under IAR semantics (in particular, the new assertions must not introduce any inconsistencies). Explanation facilities are all the more essential when using inconsistency-tolerant semantics, as has been argued by Arioua, Tamani, and Croitoru (2014). Indeed, as we propose to use the brave, AR, and IAR semantics conjointly

1. The use of tractable upper and lower bounds is a common technique for coping with the intractability of reasoning, cf. the seminal work of Selman and Kautz (1991) on Horn approximations, or more recent work on using Datalog approximations for CQ answering over expressive DL ontologies (Zhou, Grau, Nenov, Kaminski, & Horrocks, 2015). In these works, the original KB is approximated using a tractable language, whereas we keep the original KB but adopt tractable approximate semantics.
2. The CQAPri system and benchmark can be downloaded from https://www.lri.fr/~bourgaux/CQAPri.

to classify query answers into three categories of increasing reliability, a user may naturally wonder why a given tuple was assigned to, or excluded from, one of these categories.

Our second contribution is the development and experimentation of a framework for explaining query (non)answers under inconsistency-tolerant semantics. Specifically, we define explanations of positive and negative query answers under brave, AR and IAR semantics. Intuitively, such explanations pinpoint the portions of the ABox that, in combination with the TBox, suffice to obtain a given query (non-)answer under the considered semantics. We focus on ABox assertions since inconsistencies are assumed to stem from errors in the ABox, and because this already yields a non-trivial framework to study. We investigate the main search and decision problems related to explanations: generating an (arbitrary) explanation, generating a most preferred explanation according to some natural ranking criteria, recognizing (most preferred) explanations, and checking whether an assertion is relevant/necessary (i.e. appears in some/all explanations). We study the data complexity of these problems for DL-Lite$_\mathcal{R}$, showing (in)tractability of each of the tasks and pinpointing the exact complexity of the intractable decision problems. Interestingly, we establish tight connections between the intractable decision problems, as well as the problem of generating (preferred) explanations, and SAT-based reasoning tasks like computing minimal models or minimal unsatisfiable sets of clauses (so-called MUSes). This enables effective solutions to these problems using solvers for Boolean satisfaction and optimization problems. We have implemented our explanation services in CQAPri and performed experiments to understand the practical difficulty of computing (preferred) explanations and the sets of relevant and necessary assertions. Our experiments show that explanations can generally be computed in a reasonable amount of time.

This article extends two conference papers (Bienvenu, Bourgaux, & Goasdoué, 2014, 2016). Compared to the conference versions, the present article includes some improved complexity upper bounds (the AC^0 membership results in Table 1) and full proofs of the complexity results; it also describes the results of a new set of experiments with an improved benchmark and provides a more thorough analysis of the cost of explanations.

Organization of this paper Section 2 presents the syntax and semantics of DL-Lite knowledge bases, introduces query answering and query rewriting, and recalls relevant complexity classes and complexity results. In Section 3, we define the three inconsistency-tolerant semantics considered in this paper (brave, IAR, and AR) and discuss their properties. The following section describes the algorithms we use to perform query answering under these semantics, and in particular, we define the SAT encoding used for the AR semantics. Our explanation framework is presented in Section 5, with a first subsection devoted to introducing and illustrating the different notions of explanation, and a second subsection providing a comprehensive complexity analysis of the main decision and generating tasks related to explanations. In Section 6, we describe the CQAPri system and the benchmark we developed, and Section 7 presents the results of our experimental evaluation. We conclude the paper with a discussion of related work (Section 8) and directions for future work (Section 9). To improve readability, we have moved some proofs to Appendix A and large figures and tables to Appendix B.

$$\begin{aligned}\mathcal{T}^* &= \{\mathsf{AProf} \sqsubseteq \mathsf{Prof}, \mathsf{FProf} \sqsubseteq \mathsf{Prof}, \exists\mathsf{Advise} \sqsubseteq \mathsf{Prof}, \mathsf{Prof} \sqsubseteq \mathsf{PhD}, \mathsf{Postdoc} \sqsubseteq \mathsf{PhD},\\ &\quad\ \mathsf{AProf} \sqsubseteq \neg\mathsf{FProf}, \mathsf{Prof} \sqsubseteq \neg\mathsf{Postdoc}\}\\ \mathcal{A}^* &= \{\mathsf{Postdoc}(ann), \mathsf{AProf}(ann), \mathsf{FProf}(ann), \mathsf{Advise}(ann, bob),\\ &\quad\ \mathsf{Teach}(ann, c_1), \mathsf{Teach}(ann, c_2), \mathsf{Teach}(ann, c_3)\}\end{aligned}$$

$$q_1(x) = \mathsf{Prof}(x) \qquad q_2(x) = \exists y\, \mathsf{PhD}(x) \wedge \mathsf{Teach}(x, y) \qquad q_3(x) = \exists y\, \mathsf{Teach}(x, y)$$

Figure 1: TBox, ABox, and queries used in the running example

2. Preliminaries

In this section, we recall the framework of ontology-mediated query answering in the setting of description logics, focusing on the lightweight logic DL-Lite$_\mathcal{R}$, which provides the logical underpinnings of the OWL 2 QL profile.

Syntax A DL *knowledge base (KB)* consists of an ABox and a TBox, both constructed from a set $\mathsf{N_C}$ of *concept names* (unary predicates), a set of $\mathsf{N_R}$ of *role names* (binary predicates), and a set $\mathsf{N_I}$ of *individuals* (constants). The *ABox* (dataset) consists of a finite number of *concept assertions* of the form $A(a)$ and *role assertions* of the form $R(a, b)$, where $A \in \mathsf{N_C}$, $R \in \mathsf{N_R}$, $a, b \in \mathsf{N_I}$. We use $\mathsf{Ind}(\mathcal{A})$ to refer to the set of individuals that appear in $\mathcal{A}$. The *TBox* (ontology) consists of a set of axioms whose form depends on the DL in question. In DL-Lite$_\mathcal{R}$, TBox axioms are *concept inclusions* $B \sqsubseteq C$ and *role inclusions* of the form $S \sqsubseteq Q$ built according to the following syntax, where $A \in \mathsf{N_C}$ and $R \in \mathsf{N_R}$:

$$B := A \mid \exists S, \quad C := B \mid \neg B, \quad S := R \mid R^-, \quad Q := S \mid \neg S$$

A TBox axiom of the form $B_1 \sqsubseteq B_2$ or $S_1 \sqsubseteq S_2$ is a *positive inclusion*, and a TBox axiom of the form $B_1 \sqsubseteq \neg B_2$ or $S_1 \sqsubseteq \neg S_2$ is a *negative inclusion*.

Example 1. For our running example, we will use the DL-Lite$_\mathcal{R}$ KB $\mathcal{K}^* = \langle \mathcal{T}^*, \mathcal{A}^* \rangle$ in Figure 1 about the university domain. The TBox $\mathcal{T}^*$ expresses relationships between concepts for professors (Prof) of two levels of seniority (AProf, FProf), PhD holders (PhD), postdoctoral researchers ($\mathsf{Postdoc}$), and roles linking instructors to their courses (Teach) and advisors to their students (Advise). For example, the third inclusion states that everyone who advises someone is a professor, and the last asserts the disjointness of concepts Prof and $\mathsf{Postdoc}$. The ABox $\mathcal{A}^*$ provides information about an individual *ann*. For instance, the first and last assertions state respectively that *ann* is a postdoc and *ann* teaches c_3. ◁

Semantics An *interpretation* has the form $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set and $\cdot^\mathcal{I}$ maps each $A \in \mathsf{N_C}$ to $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, each $R \in \mathsf{N_R}$ to $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and each $a \in \mathsf{N_I}$ to $a^\mathcal{I} \in \Delta^\mathcal{I}$, with $a^\mathcal{I} \neq b^\mathcal{I}$ for $a \neq b$ (the last condition is the well known Unique Names Assumption, or UNA[3]). The function $\cdot^\mathcal{I}$ is straightforwardly extended to general concepts and roles, e.g. $(R^-)^\mathcal{I} = \{(c, d) \mid (d, c) \in R^\mathcal{I}\}$ and $(\exists Q)^\mathcal{I} = \{c \mid \exists d : (c, d) \in Q^\mathcal{I}\}$.

3. Our results for DL-Lite$_\mathcal{R}$ do not require the UNA, but it is standard to adopt the UNA for databases and DL-Lite and necessary for the proper treatment of other DL-Lite dialects admitting functionality.

We say that $\mathcal{I}$ satisfies an inclusion $G \sqsubseteq H$ if $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$; it satisfies $A(a)$ (resp. $R(a,b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$). We call $\mathcal{I}$ a *model* of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{I}$ satisfies all axioms in $\mathcal{T}$ and assertions in $\mathcal{A}$. A KB $\mathcal{K}$ is *consistent* if it has a model; otherwise it is inconsistent, denoted $\mathcal{K} \models \bot$. An ABox $\mathcal{A}$ is *$\mathcal{T}$-consistent* if the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent.

Queries A *first-order (FO) query* is a first-order logic formula whose atoms are built using the predicate symbols in $\mathsf{N_C} \cup \mathsf{N_R}$ and constants in $\mathsf{N_I}$. For the user queries, we will focus on the subclass of *conjunctive queries* (CQs) which take the form $q(\vec{x}) = \exists \vec{y}\, \psi(\vec{x}, \vec{y})$, where ψ is a conjunction of atoms of the forms $A(t)$ or $R(t, t')$ whose terms are either variables from $\vec{x} \cup \vec{y}$ or individuals. A CQ consisting of a single atom is called an *instance query* (IQ). For convenience, we introduce functions atoms and terms that return respectively the set of atoms and terms in a query. A query without free variables is called *Boolean*. Given a query $q(\vec{x})$ with free variables $\vec{x} = (x_1, \ldots, x_k)$ and a tuple of individuals $\vec{a} = (a_1, \ldots, a_k)$, we say that q has *arity* k and use $q(\vec{a})$ to denote the Boolean query resulting from replacing each x_i by a_i.

A tuple of individuals $\vec{a}$ is an *answer* to an FO query $q(\vec{x})$ of an interpretation $\mathcal{I}$, written $\mathcal{I} \models q(\vec{a})$, iff it has the same arity as q and the Boolean query $q(\vec{a})$ is satisfied in $\mathcal{I}$ according to standard first-order logic semantics. When q is a CQ, one can define answers in terms of the existence of matches, where a *match for $q(\vec{a})$ in $\mathcal{I}$* is a function $\pi : \mathsf{terms}(q) \to \Delta^{\mathcal{I}}$ such that (i) $\pi(\vec{x}) = (a_1^{\mathcal{I}}, \ldots, a_k^{\mathcal{I}})$, (ii) $\pi(t) = t^{\mathcal{I}}$ for every $t \in \mathsf{N_I}$, (iii) $\pi(t) \in A^{\mathcal{I}}$ for every $A(t) \in \mathsf{atoms}(q)$, and (iv) $(\pi(t), \pi(t')) \in R^{\mathcal{I}}$ for every $R(t, t') \in \mathsf{atoms}(q)$. Thus, $\mathcal{I} \models q(\vec{a})$ iff there is a match for $q(\vec{a})$ in $\mathcal{I}$. The set of answers for a query $q(\vec{x})$ in $\mathcal{I}$ is denoted $\mathsf{ans}(q, \mathcal{I})$.

When we speak of query answering, we mean the problem of computing the set of certain answers to the query, defined as follows. A tuple $\vec{a}$ is a *certain answer* to q over $\mathcal{K}$, written $\mathcal{K} \models q(\vec{a})$, iff $\vec{a}$ is an *answer* to $q(\vec{x})$ in every model of $\mathcal{K}$. The set of certain answers for $q(\vec{x})$ over $\mathcal{K}$ is written $\mathsf{cert}(q, \mathcal{K})$. We remark that when $\mathcal{K}$ is inconsistent, $\mathsf{cert}(q, \mathcal{K})$ contains every tuple of ABox individuals having the same arity as q.

Complexity measures When analyzing the complexity of query answering, we consider the associated decision problem: given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, query $q(\vec{x})$, and tuple of individuals $\vec{a}$ of the same arity as $\vec{x}$, decide whether $\mathcal{K} \models q(\vec{a})$. There are different ways of measuring the complexity of query answering (or query evaluation), depending on which parameters are regarded as the input. *Data complexity* considers only the size of the ABox (data), denoted $|\mathcal{A}|$, whereas *combined complexity* considers the size of the whole problem, i.e. the size of the KB ($|\mathcal{K}| = |\mathcal{A}| + |\mathcal{T}|$) plus the size of the query $|q|$ (defined as the number of atoms in the query). Data complexity is considered the more relevant measure when the size of the TBox and the size of the query are negligible compared to the size of the ABox, which is typically the case in the context of OMQA.

The following complexity classes are used in this work (we refer the reader to Papadimitriou, 1995; Arora & Barak, 2009, for introductions to computational complexity):

- AC^0: problems that can be solved by a uniform family of cicuits of constant depth and polynomial-size, with unbounded-fanin AND and OR gates.
- NL: problems that can be solved in non-deterministic logarithmic space.
- PTime: problems which are solvable in polynomial time in the size of the input.
- NP: problems which are solvable in non-deterministic polynomial time.

- coNP: problems whose complement is in NP.
- BH_2: problems that are the intersection of a problem in NP and a problem in coNP.
- Σ_2^p: problems which are solvable in non-deterministic polynomial time with access to an NP oracle.
- Π_2^p: problems whose complement is in Σ_2^p.

These classes are related as follow: $\mathsf{AC}^0 \subseteq \mathrm{NL} \subseteq \mathrm{PTime}$, $\mathrm{PTime} \subseteq \mathrm{NP} \subseteq \Sigma_2^p$ and $\mathrm{PTime} \subseteq \mathrm{coNP} \subseteq \Pi_2^p$. It is known that $\mathsf{AC}^0 \subsetneq \mathrm{PTime}$, and it is widely believed that all of the inclusions are proper.

We recall that in the database setting, the problem of deciding if a tuple is an answer to an FO query is in AC^0 in data complexity (Immerman, 1987), and (the decision version of) CQ evaluation is NP-complete in combined complexity (Chandra & Merlin, 1977).

Query answering through rewriting We now introduce query rewriting, which is a prominent algorithmic approach to OMQA. The basic idea is to rewrite the query so as to incorporate all relevant information from the TBox, and then evaluate the rewritten query over the ABox, which is treated as a database.

Formally, with every ABox $\mathcal{A}$, we associate a corresponding interpretation $\mathcal{I}_\mathcal{A} = (\Delta^{\mathcal{I}_\mathcal{A}}, \cdot^{\mathcal{I}_\mathcal{A}})$ defined as follows:

$$\Delta^{\mathcal{I}_\mathcal{A}} = \mathsf{Ind}(\mathcal{A}) \qquad A^{\mathcal{I}_\mathcal{A}} = \{a \mid A(a) \in \mathcal{A}\} \text{ for every } A \in \mathsf{N_C}$$
$$a^{\mathcal{I}_\mathcal{A}} = a \text{ for every } a \in \mathsf{Ind}(\mathcal{A}) \qquad R^{\mathcal{I}_\mathcal{A}} = \{(a,b) \mid R(a,b) \in \mathcal{A}\} \text{ for every } R \in \mathsf{N_R}$$

The interpretation treats the ABox like a relational database by making true precisely those assertions appearing in the ABox. A *(first-order) rewriting* of a query q w.r.t. a TBox $\mathcal{T}$ and $\mathcal{T}$-consistent ABoxes is an FO-query q' such that $\mathsf{cert}(q, \langle \mathcal{T}, \mathcal{A}\rangle) = \mathsf{ans}(q', \mathcal{I}_\mathcal{A})$ for all $\mathcal{T}$-consistent ABoxes $\mathcal{A}$. Observe that the original problem of computing $\mathsf{cert}(q, \langle \mathcal{T}, \mathcal{A}\rangle)$ is reduced to the task of evaluating a first-order query q' over the finite interpretation $\mathcal{I}_\mathcal{A}$, and the latter task can be handled by a relational database system. (We shall see later in the section how to check whether the input ABox is $\mathcal{T}$-consistent, which will allow us to devise rewritings that work for arbitrary ABoxes.)

The DL-Lite family possesses the first-order rewritability property, meaning that for standard DL-Lite dialects like DL-Lite$_\mathcal{R}$, there exists an FO-rewriting of every CQ and every TBox (which is not the case for most DLs). In fact, it is sufficient to consider *UCQ-rewritings*, which take the more restricted form of unions of conjunctive queries. We note that when working with UCQs, we will sometimes treat them as sets of CQs, using e.g. $q \in Q$ to denote that q is a disjunct of the UCQ Q.

Example 2. The UCQ $q'(x) = \mathsf{Prof}(x) \vee \mathsf{AProf}(x) \vee \mathsf{FProf}(x) \vee \exists y \mathsf{Advise}(x,y)$ is a rewriting of $q_1(x) = \mathsf{Prof}(x)$ w.r.t. $\mathcal{T}^*$ and consistent ABoxes. Observe that the disjuncts capture the four different ways to infer that an individual is a professor according to the TBox $\mathcal{T}^*$. ◁

Several concrete UCQ-rewriting algorithms have been developed and implemented for DL-Lite$_\mathcal{R}$. The original *PerfectRef* algorithm of Calvanese et al. (2007) and most subsequent proposals employ a backwards-chaining approach, in which each rewriting step replaces a query atom by another atom which implies it. Importantly, while the resulting

UCQ may be of exponential size, every CQ in the disjunction has no more atoms than the original query (a property that we will exploit later).

Answering a CQ q over a consistent DL-Lite KB $\langle \mathcal{T}, \mathcal{A} \rangle$ amounts to searching for matches for the CQs in a UCQ-rewriting of q in $\mathcal{I}_\mathcal{A}$. We introduce the notion of image to capture the parts of the ABox that participate in a match. Formally, an *image* of a CQ $q(\vec{x}) = \exists \vec{y}\, \psi(\vec{x}, \vec{y})$ in an ABox $\mathcal{A}$ is a set of assertions $\mathcal{B} \subseteq \mathcal{A}$ such that there is a match $\pi : \vec{x} \cup \vec{y} \mapsto \mathsf{Ind}(\mathcal{A})$ for q in $\mathcal{I}_\mathcal{A}$ such that $\mathcal{B} = \mathsf{atoms}(\psi(\pi(\vec{x}), \pi(\vec{y})))$. We can extend the notion of image to UCQs as follows: a set of assertions is an image of a UCQ Q iff it is the image of some CQ $q \in Q$.

The following theorem recalls the complexity of query answering in DL-Lite$_\mathcal{R}$. Membership in AC^0 for data complexity is an immediate consequence of the reduction to FO-query evaluation, which is known to be in AC^0 for data complexity, and the NP lower bound is likewise inherited from the NP-hardness of CQ evaluation over relational databases. For the NP upper bound, we cannot directly use the UCQ-rewriting, as it can be of exponential size. Instead, one can guess a single CQ in the UCQ-rewriting, together with a polynomial proof that this CQ appears in the rewriting and has an image in the ABox. For instance queries, one can exploit the fact that each disjunct in the UCQ-rewriting has a single atom.

Theorem 2.1 (Calvanese et al., 2007; Artale, Calvanese, Kontchakov, & Zakharyaschev, 2009)**.** *In DL-Lite$_\mathcal{R}$, conjunctive query answering is* AC^0 *w.r.t. data complexity and* NP*-complete w.r.t. combined complexity. Instance checking is in* PTime *(more precisely:* NL*-complete) w.r.t. combined complexity.*

ABox consistency In DL-Lite$_\mathcal{R}$, ABox consistency checking reduces to answering a union of conjunctive queries corresponding to each possible violation of the TBox constraints: we can build a Boolean query $q^\mathcal{T}_{unsat}$ that looks for a counterexample to one of the negative inclusions entailed by $\mathcal{T}$, and which evaluates to true over $\mathcal{I}_\mathcal{A}$ just in the case that the ABox $\mathcal{A}$ is $\mathcal{T}$-inconsistent.

We briefly explain how to construct $q^\mathcal{T}_{unsat}$. First, for each negative inclusion $\zeta \in \mathcal{T}$ in the TBox, build a Boolean CQ γ_ζ that tests for a direct violation of ζ. Then each CQ γ_ζ is rewritten w.r.t. $\mathcal{T}$, yielding a UCQ Γ_ζ that tests for any (possibly indirect) violation of ζ. Finally, we take the disjunction of the rewritten queries: $q^\mathcal{T}_{unsat} = \bigvee_\zeta \Gamma_\zeta$, where ζ ranges over the negative inclusions in $\mathcal{T}$. We illustrate the procedure in the following example.

Example 3. The TBox $\mathcal{T}^*$ contains two negative inclusions $\zeta_1 = \mathsf{AProf} \sqsubseteq \neg\mathsf{FProf}$ and $\zeta_2 = \mathsf{Prof} \sqsubseteq \neg\mathsf{Postdoc}$. To check for a direct violation of $\mathsf{AProf} \sqsubseteq \neg\mathsf{FProf}$, we can use the CQ $\gamma_{\zeta_1} = \exists x \mathsf{AProf}(x) \wedge \mathsf{FProf}(x)$. The rewriting step leaves this CQ unchanged (as there is no way to derive AProf or FProf), so $\Gamma_{\zeta_1} = \gamma_{\zeta_1}$. For the second negative inclusion, we build the CQ $\gamma_{\zeta_2} = \exists x \mathsf{Prof}(x) \wedge \mathsf{Postdoc}(x)$, which is rewritten into the following UCQ

$$(\exists x\, \mathsf{Prof}(x) \wedge \mathsf{Postdoc}(x)) \vee (\exists x\, \mathsf{AProf}(x) \wedge \mathsf{Postdoc}(x)) \vee$$
$$(\exists x\, \mathsf{FProf}(x) \wedge \mathsf{Postdoc}(x)) \vee (\exists xy\, \mathsf{Advise}(x, y) \wedge \mathsf{Postdoc}(x))$$

Finally, we take the disjunction of the computed UCQs, to get $q^{\mathcal{T}^*}_{unsat} = \Gamma_{\zeta_1} \vee \Gamma_{\zeta_2}$. ◁

The next theorem recalls the complexity of consistency checking in DL-Lite$_\mathcal{R}$. To show the polynomial combined complexity result, one can exploit the fact that each CQ appearing in $q^\mathcal{T}_{unsat}$ has at most two atoms, and thus it suffices to examine all (polynomially many) subsets of the ABox with at most two assertions to see if they contain a match for $q^\mathcal{T}_{unsat}$.

Theorem 2.2 (Calvanese et al., 2007; Artale et al., 2009)**.** *In DL-Lite$_\mathcal{R}$, consistency checking is in* AC^0 *w.r.t. data complexity, and in* PTime *(more precisely:* NL*-complete) w.r.t. combined complexity.*

We conclude this section by observing that one can combine in a straightforward manner a rewriting of q w.r.t. $\mathcal{T}$ and $\mathcal{T}$-consistent ABoxes with the unsatisfiability query $q^{\mathcal{T}}_{unsat}$ in order to obtain a rewriting that outputs the certain answers irregardless of whether the considered ABox is $\mathcal{T}$-consistent. Indeed, it suffices to consider the query $q_{rw} \vee (q^{\mathcal{T}}_{unsat} \wedge q_{all})$, where q_{rw} is a rewriting for consistent ABoxes and q_{all} is a query that returns all tuples of ABox individuals of the same arity as q.

3. Inconsistency-Tolerant Semantics

When the ABox is inconsistent with the TBox, it may be impossible to clean the data to make it consistent, either for lack of time because the data is too large, or for lack of information on how to resolve the contradictions. It is therefore crucial to be able to retrieve meaningful answers from inconsistent data. However, when the KB is inconsistent, every Boolean query is entailed under the classical semantics, and for non-Boolean queries, every tuple of individuals of the appropriate arity is considered a certain answer. Classical semantics is essentially useless in the inconsistent case, as it fails to provide any relevant information, motivating the need for alternative semantics. In this section, we present three *inconsistency-tolerant semantics* that have been proposed in the literature to query inconsistent KBs, namely, the AR, IAR, and brave semantics, and we recall what is known about the complexity of query answering under these semantics in DL-Lite$_\mathcal{R}$.

The *AR semantics* (ABox Repair semantics) (Lembo et al., 2010) adapts the *consistent query answering* framework developed in the database arena (Arenas et al., 1999; Bertossi, 2006; Chomicki, 2007; Bertossi, 2011) to DL KBs. Consistent query answering amounts to considering those answers that hold in every *repair*, defined as consistent subsets of the database that are "as close as possible" to the actual database instance. In the database setting, which adopts the closed world assumption, repairs may be obtained from the actual database by deletion or insertion of tuples (or changes of attributes values), since constraints may be violated by either the presence or absence of certain tuples. By contrast, in the DL setting where open world semantics is used, inconsistency can only stem from the presence of incompatible ABox assertions, leading to a simpler notion of repair based upon removal of assertions:

Definition 3.1 (Repair)**.** An *ABox repair* of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$, or repair for short, is an inclusion-maximal subset of the ABox $\mathcal{A}$ which is $\mathcal{T}$-consistent. The set of all repairs of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is denoted by $Rep(\mathcal{K})$ or $Rep(\mathcal{T}, \mathcal{A})$.

Example 4. The inconsistent KB $\mathcal{K}^*$ has the following three repairs:

$\mathcal{R}_1 = \{\mathsf{Postdoc}(ann), \mathsf{Teach}(ann, c_1), \mathsf{Teach}(ann, c_2), \mathsf{Teach}(ann, c_3)\}$

$\mathcal{R}_2 = \{\mathsf{AProf}(ann), \mathsf{Advise}(ann, bob), \mathsf{Teach}(ann, c_1), \mathsf{Teach}(ann, c_2), \mathsf{Teach}(ann, c_3)\}$

$\mathcal{R}_3 = \{\mathsf{FProf}(ann), \mathsf{Advise}(ann, bob), \mathsf{Teach}(ann, c_1), \mathsf{Teach}(ann, c_2), \mathsf{Teach}(ann, c_3)\}$

Each repair contains one of the three assertions about the position *ann* occupies ($\mathsf{Postdoc}$ or AProf or FProf), and omits all conflicting assertions. For instance, the first repair, $\mathcal{R}_1$, retains $\mathsf{Postdoc}(ann)$ and drops the three assertions that contradict it according to the knowledge in $\mathcal{T}^*$, namely, $\mathsf{AProf}(ann)$, $\mathsf{FProf}(ann)$, and $\mathsf{Advise}(ann, bob)$. ◁

The repairs correspond to all possible ways of repairing the ABox while preserving as much information as possible (in the sense of set inclusion) and can be viewed as 'possible worlds'. If the quality of the data is relatively good, we can assume that one of the repairs corresponds to the real world, but in the absence of further information, we cannot know which one. For this reason, the AR semantics stipulates that for a tuple to be counted as an answer to a query over an inconsistent KB, it must hold w.r.t. *every repair*. Formally:

Definition 3.2 (AR semantics)**.** A tuple $\vec{a}$ is an answer to a query q over a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ under *AR semantics*, written $\langle \mathcal{T}, \mathcal{A} \rangle \models_{\text{AR}} q(\vec{a})$, if and only if $\langle \mathcal{T}, \mathcal{R} \rangle \models q(\vec{a})$ for every repair $\mathcal{R} \in \mathit{Rep}(\mathcal{T}, \mathcal{A})$. We call $\vec{a}$ a (positive) *AR-answer*.

Example 5. In our running example, we consider the three queries given in Figure 1. Evaluating these queries over the KB $\mathcal{K}^*$ under AR semantics yields the following results:

$$\mathcal{K}^* \not\models_{\text{AR}} \mathsf{Prof}(ann) \qquad \mathcal{K}^* \models_{\text{AR}} \exists y\, \mathsf{PhD}(ann) \wedge \mathsf{Teach}(ann, y) \qquad \mathcal{K}^* \models_{\text{AR}} \exists y\, \mathsf{Teach}(ann, y)$$

Indeed, it can be verified that $\langle \mathcal{T}^*, \mathcal{R}_1 \rangle \not\models \mathsf{Prof}(ann)$, and for every $1 \leq i \leq 3$, we have $\langle \mathcal{T}^*, \mathcal{R}_i \rangle \models \exists y\, \mathsf{PhD}(ann) \wedge \mathsf{Teach}(ann, y)$ and $\langle \mathcal{T}^*, \mathcal{R}_i \rangle \models \exists y\, \mathsf{Teach}(ann, y)$. ◁

The AR semantics is arguably the most natural inconsistency-tolerant semantics. Unfortunately, as the next result shows, query answering under AR semantics is intractable in data complexity, even for instance queries. The coNP lower bound exploits the fact that the number of repairs may be exponential in the size of the ABox, while the coNP upper bound can be shown by a simple guess-and-check procedure: to show $\langle \mathcal{T}, \mathcal{A} \rangle \not\models_{\text{AR}} q(\vec{a})$, guess a subset $\mathcal{R} \subseteq \mathcal{A}$ and verify that it is a repair such that $\langle \mathcal{T}, \mathcal{R} \rangle \not\models q(\vec{a})$.

Theorem 3.3 (Lembo et al., 2010)**.** *Conjunctive query answering and instance checking under AR semantics over DL-Lite$_\mathcal{R}$ knowledge bases are both co*NP*-complete w.r.t. data complexity.*

We remark that tractability cannot be regained by reducing the expressivity of the ontology: Bienvenu (2012) showed that CQ answering is coNP-complete w.r.t. data complexity even for simple ontologies consisting of axioms of the form $A_1 \sqsubseteq (\neg) A_2$, where $A_1, A_2 \in \mathsf{N_C}$.

As for classical semantics, where the combined complexity of CQ answering is higher than its data complexity (NP vs. AC^0), the combined complexity of query answering under AR semantics is a level higher in the polynomial hierarchy than its data complexity.

Theorem 3.4 (Bienvenu & Rosati, 2013)**.** *Conjunctive query answering under AR semantics over DL-Lite$_\mathcal{R}$ knowledge bases is Π_2^p-complete w.r.t. combined complexity, and instance checking is co*NP*-complete w.r.t. combined complexity.*

The negative complexity results for AR semantics led Lembo et al. (2010) to propose an approximation of AR semantics that corresponds to querying the intersection of the repairs. The resulting IAR (Intersection of ABox Repairs) semantics is formally defined as follows:

Definition 3.5 (IAR semantics)**.** A tuple $\vec{a}$ is an answer to a query q over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ under *IAR semantics*, written $\langle \mathcal{T}, \mathcal{A} \rangle \models_{\text{IAR}} q(\vec{a})$, if and only if $\langle \mathcal{T}, \mathcal{R}_\cap \rangle \models q(\vec{a})$, where $\mathcal{R}_\cap$ is the intersection of the repairs of $\mathcal{K}$. We call $\vec{a}$ a (positive) *IAR-answer*.

This semantics follows the 'when in doubt throw it out' principle, proposed in the area of belief revision and update, and provides a more conservative semantics than AR. The answers holding under IAR semantics can reasonably be considered the most reliable answers, as they do not rely upon any assertions involved in contradictions.

Example 6. Intersecting the repairs $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ yields the following ABox:

$$\mathcal{A}^*_\cap = \{\mathsf{Teach}(ann, c_1), \mathsf{Teach}(ann, c_2), \mathsf{Teach}(ann, c_3)\}$$

If we evaluate our example queries over the KB $\langle \mathcal{T}^*, \mathcal{A}^*_\cap \rangle$, we obtain:

$$\mathcal{T}^*\mathcal{A}^*_\cap \not\models \mathsf{Prof}(ann) \qquad \langle \mathcal{T}^*, \mathcal{A}^*_\cap \rangle \not\models \exists y\, \mathsf{PhD}(ann) \wedge \mathsf{Teach}(ann, y)$$
$$\langle \mathcal{T}^*, \mathcal{A}^*_\cap \rangle \models \exists y\, \mathsf{Teach}(ann, y)$$

We therefore obtain the following results under IAR semantics:

$$\mathcal{K}^* \not\models_{\text{IAR}} \mathsf{Prof}(ann) \quad \mathcal{K}^* \not\models_{\text{IAR}} \exists y\, \mathsf{PhD}(ann) \wedge \mathsf{Teach}(ann, y) \quad \mathcal{K}^* \models_{\text{IAR}} \exists y\, \mathsf{Teach}(ann, y)$$

Observe that if we move from AR to IAR semantics, *ann* is no longer considered an answer to q_2. This is because there is no single justification for $\mathsf{PhD}(ann)$ common to all repairs. ◁

The next theorem summarizes what is known about the complexity of query answering under IAR semantics. The AC^0 upper bound in data complexity (which matches that of classical semantics) is shown by defining a first-order query rewriting procedure. The general idea is to add to the classical UCQ-rewriting expressions that ensure that the assertions used to derive the query are not contradicted by other assertions by enumerating the different possible contradictions. See Section 8.2.1 for other approaches to IAR query answering.

Theorem 3.6 (Lembo et al., 2011, 2015; Bienvenu & Rosati, 2013)**.** *CQ answering under IAR semantics over DL-Lite$_\mathcal{R}$ knowledge bases is in* AC^0 *w.r.t. data complexity and* NP*-complete w.r.t. combined complexity. Instance checking is* NL*-complete w.r.t. combined complexity.*

While the IAR semantics is the most cautious inconsistency-tolerant semantics, the brave semantics introduced by Bienvenu and Rosati (2013), which considers those answers that can be obtained from at least one repair, can be viewed as the most permissive:

Definition 3.7 (Brave semantics)**.** A tuple $\vec{a}$ is an answer for a query q over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ under *brave semantics*, written $\langle \mathcal{T}, \mathcal{A} \rangle \models_{\text{brave}} q(\vec{a})$, if and only if $\langle \mathcal{T}, \mathcal{R} \rangle \models q(\vec{a})$ for some repair $\mathcal{R} \in \mathit{Rep}(\mathcal{T}, \mathcal{A})$. We call $\vec{a}$ a (positive) *brave-answer*.

It is possible to perform query answering under brave semantics by means of first-order query rewriting, yielding a low AC^0 data complexity:

Theorem 3.8 (Bienvenu & Rosati, 2013)**.** *CQ answering under brave semantics over DL-Lite knowledge bases is in* AC^0 *w.r.t. data complexity and* NP*-complete w.r.t. combined complexity. Instance checking is* NL*-complete w.r.t. combined complexity.*

Example 7. Evaluating our example queries under the brave semantics yields:

$$\mathcal{K}^* \models_{\text{brave}} \mathsf{Prof}(ann) \quad \mathcal{K}^* \models_{\text{brave}} \exists y\, \mathsf{PhD}(ann) \wedge \mathsf{Teach}(ann, y) \quad \mathcal{K}^* \models_{\text{brave}} \exists y\, \mathsf{Teach}(ann, y)$$

Indeed, for all three queries, there exists at least one repair in which ann is obtained as an answer. Specifically, we have $\langle \mathcal{T}^*, \mathcal{R}_2 \rangle \models \mathsf{Prof}(ann)$, $\langle \mathcal{T}^*, \mathcal{R}_1 \rangle \models \exists y\, \mathsf{PhD}(ann) \wedge \mathsf{Teach}(ann, y)$, and $\langle \mathcal{T}^*, \mathcal{R}_1 \rangle \models \exists y\, \mathsf{Teach}(ann, y)$. Observe that under brave semantics, ann is an answer to q_1, whereas it is not an answer under AR semantics. ◁

We point out that the set of answers that are obtained using brave semantics may be inconsistent when considered together. In our example, the brave semantics allows us to infer both $\mathsf{Postdoc}(ann)$ and $\mathsf{AProf}(ann)$, which contradict each other in the presence of the TBox. By contrast, it is impossible to derive a contradiction from the set of query answers obtained using AR (or IAR) semantics, since there exists a single repair from which all such answers can be derived, and by definition, a repair is consistent with the TBox.

The relations between the three semantics are stated in the following proposition. The IAR semantics is an under-approximation of the AR semantics, i.e. every query answer under IAR semantics also counts as an answer under AR semantics, whereas brave semantics provides an over-approximation, i.e. every query answer obtained using AR semantics is also an answer under brave semantics.

Proposition 3.9. *The IAR, AR, and brave semantics are related as follows:*

$$\mathcal{K} \models_{IAR} q(\vec{a}) \quad \Longrightarrow \quad \mathcal{K} \models_{AR} q(\vec{a}) \quad \Longrightarrow \quad \mathcal{K} \models_{brave} q(\vec{a})$$

None of the reverse implications holds.

By using the three semantics together, we can classify query answers into three categories based upon their reliability, with IAR semantics identifying the surest answers, AR semantics identifying answers that are reasonably likely to hold, and brave semantics identifying possible answers, which have at least one consistent reason to hold.

4. Efficient Inconsistency-Tolerant Query Answering in DL-Lite

This section presents the algorithms we use to compute the query answers under the brave, IAR, and AR semantics. Their implementation is described later in Section 6.

We should emphasize that the novel contribution of this section is the algorithm for the AR semantics and the use of the three semantics together to classify query answers, as algorithms for querying DL-Lite KBs under the IAR and brave semantics have already been proposed by Lembo et al. (2011, 2015) and Bienvenu and Rosati (2013).

Henceforth, to simplify formulations, we will sometimes refer to $\mathcal{T}$ without explicitly stating that it is a TBox, and likewise, for $\mathcal{A}$ (ABox), $\mathcal{K}$ (KB), q (conjunctive query), $\vec{x}$ (tuple of variables), and $\vec{a}$ (tuple of individuals).

4.1 Conflicts and Causes

As a first step, we introduce two central notions of conflicts and causes, as well as basic procedures for computing them. Conflicts are simply defined as the minimal sets of assertions responsible for a KB being inconsistent:

Definition 4.1 (Conflict)**.** A *conflict*[4] of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is an inclusion-minimal $\mathcal{T}$-inconsistent subset of $\mathcal{A}$. The set of conflicts of $\mathcal{K}$ is denoted $\mathsf{conflicts}(\mathcal{K})$.

Observe that a consistent KB has an empty set of conflicts, while an inconsistent KB must possess at least one conflict.

Example 8. The KB $\mathcal{K}^*$ has the following conflicts:

$$\begin{aligned}\mathsf{conflicts}(\mathcal{K}^*) = &\{\{\mathsf{AProf}(ann), \mathsf{FProf}(ann)\}, \{\mathsf{AProf}(ann), \mathsf{Postdoc}(ann)\},\\ &\{\mathsf{FProf}(ann), \mathsf{Postdoc}(ann)\}, \{\mathsf{Advise}(ann, bob), \mathsf{Postdoc}(ann)\}\}\end{aligned}$$ ◁

The following proposition relates the conflicts of a KB with the images of the unsatisfiability query $q^{\mathcal{T}}_{unsat}$ from Section 2.

Proposition 4.2. *A subset $\mathcal{U} \subseteq \mathcal{A}$ belongs to $\mathsf{conflicts}(\mathcal{T}, \mathcal{A})$ iff $\mathcal{U}$ is an inclusion-minimal image of $q^{\mathcal{T}}_{unsat}$ in $\mathcal{A}$.*

We give an example that shows that some images of $q^{\mathcal{T}}_{unsat}$ may not be conflicts:

Example 9. Consider $\mathcal{T} = \{\exists R^- \sqsubseteq \neg \exists R\}$ and ABox $\mathcal{A} = \{R(a, a), R(a, b)\}$. The associated query $q^{\mathcal{T}}_{unsat} = \exists xyz R(x, y) \wedge R(y, z)$ has two images in $\mathcal{A}$: $\{R(a, a), R(a, b)\}$ and $\{R(a, a)\}$. However, due to the minimality requirement, only $\{R(a, a)\}$ is a conflict. ◁

In Algorithm 1, we propose a simple procedure ComputeConflicts for computing conflicts that is based upon Proposition 4.2 and can use any existing UCQ-rewriting procedure. The first step (line 2) is to construct the query $q^{\mathcal{T}}_{unsat}$ using the method described in Section 2. Next, in lines 3-4, the algorithm considers in turn each CQ $\exists \vec{y}\, \psi(\vec{y})$ that is a disjunct of $q^{\mathcal{T}}_{unsat}$ and computes the set of answers to ψ over $\mathcal{I}_{\mathcal{A}}$. Observe that we use the non-Boolean query $\psi(\vec{y})$ (which treats $\vec{y}$ as free variables) in order to obtain all images of $\exists \vec{y}\, \psi(\vec{y})$, rather than merely checking for the existence of an image. For each answer $\vec{a}$ to $\psi(\vec{y})$, we substitute $\vec{a}$ for the variables $\vec{y}$ and add to *Images* the set of atoms in $\psi(\vec{a})$. At this point, *Images* contains all of the images of $q^{\mathcal{T}}_{unsat}$ in $\mathcal{A}$. Finally, the elements of *Images* are compared, and the non-minimal images are removed in order to keep only the conflicts (lines 6 to 9).

We point out that ComputeConflicts can be applied to any DL that admits UCQ-rewritings of unsatisfiability. The complexity depends of course on the chosen DL. For DL-Lite$_{\mathcal{R}}$, we have seen in Section 2 that the disjuncts of $q^{\mathcal{T}}_{unsat}$ all contain at most two atoms, so it is possible to construct and evaluate $q^{\mathcal{T}}_{unsat}$ in polynomial time, which gives us:

Proposition 4.3. ComputeConflicts *runs in polynomial time in $|\mathcal{A}|$ on input $\langle \mathcal{T}, \mathcal{A} \rangle$.*

The fact that every conflict of a DL-Lite$_{\mathcal{R}}$ KB has at most two assertions enables a natural representation of conflicts in terms of a *conflict graph*[5]:

4. The notion of a minimal inconsistent set has been considered in numerous works in knowledge representation and reasoning, and in particular, in the works by Lembo et al. (2015) and by Bienvenu and Rosati (2013). We introduce the term 'conflict' to be able to more easily refer to such sets.
5. The notion of conflict graph is inspired by the conflict hypergraphs from the work by Chomicki, Marcinkowski, and Staworko (2004a, 2004b)

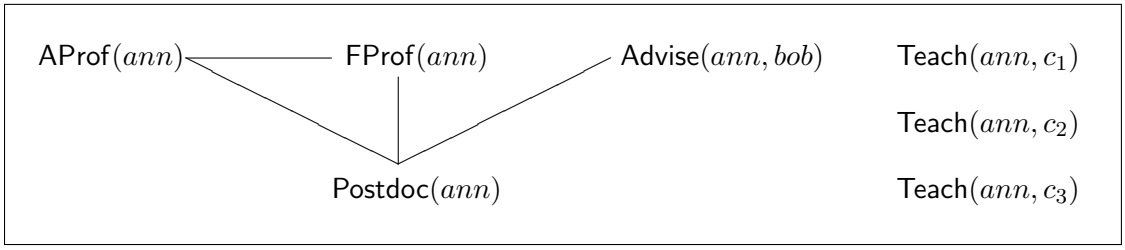
ALGORITHM 1: ComputeConflicts

Input: a TBox $\mathcal{T}$, an ABox $\mathcal{A}$
Output: the set of conflicts of $\langle \mathcal{T}, \mathcal{A} \rangle$

```
Images ← ∅;
q^T_unsat ← BuildUnsatQuery(T) ;
foreach ∃y⃗ ψ(y⃗) ∈ q^T_unsat do                    /* compute images of q^T_unsat */
    foreach a⃗ ∈ ans(ψ, I_A) do                  /* all variables in y⃗ are free here */
        Images ← Images ∪ {atoms(ψ(a⃗))};
foreach U ∈ Images do                              /* remove non-minimal images */
    foreach U' ∈ Images do
        if U' ⊆ U then
            Images ← Images \ {U};
Output Images;
```

Definition 4.4 (Conflict graph)**.** The *conflict graph* for a DL-Lite$_\mathcal{R}$ KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is a graph whose vertices are the assertions in $\mathcal{A}$ and which contains an edge from α to β iff $\{\alpha, \beta\}$ is a conflict (note that there is an edge from α to itself iff $\{\alpha\}$ is a conflict).

Example 10. The conflict graph for $\mathcal{K}^*$ is displayed below.



◁

We will utilize this notion of conflict graph in Section 6 when discussing our implementation of algorithms for inconsistency-tolerant querying.

For logics like DL-Lite$_\mathcal{R}$ that admit only unary or binary conflicts, it makes sense to speak of the assertions that enter into a conflict with a given assertion (or set of assertions):

Definition 4.5 (Conflicts of a set of assertions)**.** Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{R}$ KB and $\mathcal{B} \subseteq \mathcal{A}$. The set of *conflicts of* $\mathcal{B}$, denoted $\mathsf{confl}(\mathcal{B}, \mathcal{K})$, is:

$$\mathsf{confl}(\mathcal{B}, \mathcal{K}) = \{\beta \mid \exists \alpha \in \mathcal{B}, \{\alpha, \beta\} \in \mathsf{conflicts}(\mathcal{K})\} \cup \{\alpha \mid \alpha \in \mathcal{B}, \{\alpha\} \in \mathsf{conflicts}(\mathcal{K})\}.$$

Example 11. We have $\mathsf{confl}(\{\mathsf{AProf}(ann)\}, \mathcal{K}^*) = \{\mathsf{FProf}(ann), \mathsf{Postdoc}(ann)\}$. ◁

Observe that in the graphical view of conflicts, the set $\mathsf{confl}(\mathcal{B}, \mathcal{K})$ contains all assertions that are adjacent to an assertion from $\mathcal{B}$. To compute the set $\mathsf{confl}(\mathcal{B}, \mathcal{K})$ for a set of assertions $\mathcal{B} \subseteq \mathcal{A}$, it suffices to examine the conflicts produced by ComputeConflicts, and output all assertions α such that there exists a conflict $\{\alpha, \beta\}$ with $\beta \in \mathcal{B}$.

Since an inconsistent KB has no model, every tuple of individuals of the same arity as $\vec{x}$ is an answer to the query $q(\vec{x})$. However, only some of these answers actually have some consistent reasons to hold, which are captured by the notion of *causes*.

ALGORITHM 2: ComputeCauses

Input: a conjunctive query $q(\vec{x})$, a candidate answer $\vec{a}$, a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: the set of causes for $q(\vec{a})$ in $\mathcal{K}$

```
1  Conflicts ← ComputeConflicts(T, A);
2  Q ← UCQRef(q(a), T);
3  Images ← ∅;
4  foreach ∃y ψ(y) ∈ Q do                                  /* compute images of Q */
5  |  foreach b ∈ ans(ψ, I_A) do                  /* all variables in y are free here */
6  |  |  Images ← Images ∪ {atoms(ψ(b))};
7  foreach C ∈ Images do                                        /* filter images */
8  |  if C ∩ confl(C, K) ≠ ∅ then                          /* C is T-inconsistent */
9  |  |  Images ← Images \ {C};
10 |  foreach C' ∈ Images do
11 |  |  if C' ⊆ C then                                      /* C is non-minimal */
12 |  |  |  Images ← Images \ {C};
13 Output Images;
```

Definition 4.6 (Cause)**.** A *cause*[6] for a Boolean query q in a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is an inclusion minimal $\mathcal{T}$-consistent subset $\mathcal{C} \subseteq \mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{C} \rangle \models q$. We use $\mathsf{causes}(q, \mathcal{K})$ to refer to the set of causes for q in $\mathcal{K}$.

Note that the definition refers to Boolean queries, but we can derive a notion of cause for answers to non-Boolean queries in the obvious way: if $q(\vec{x})$ is a non-Boolean query with candidate answer $\vec{a}$, then we can commit a slight abuse of terminology and speak of the *causes for $\vec{a}$ being an answer to $q(\vec{x})$* when referring to the elements of $\mathsf{causes}(q(\vec{a}), \mathcal{K})$.

Example 12. There are twelve causes for $q_2(ann) = \exists y\, \mathsf{PhD}(ann) \wedge \mathsf{Teaches}(ann, y)$:

$$\{\mathsf{Postdoc}(ann), \mathsf{Teach}(ann, c_i)\} \qquad \{\mathsf{Advise}(ann, bob), \mathsf{Teach}(ann, c_i)\}$$

$$\{\mathsf{AProf}(ann), \mathsf{Teach}(ann, c_i)\} \qquad \{\mathsf{FProf}(ann), \mathsf{Teach}(ann, c_i)\}$$

for $i \in \{1, 2, 3\}$. ◁

The following proposition relates the causes of a query to the images of a UCQ-rewriting of the query.

Proposition 4.7. *Consider a TBox $\mathcal{T}$, a Boolean CQ q, and a UCQ-rewriting q' of q w.r.t. $\mathcal{T}$ and consistent ABoxes. Then $\mathcal{C} \subseteq \mathcal{A}$ is a cause for q in $\langle \mathcal{T}, \mathcal{A} \rangle$ iff $\mathcal{C}$ is an inclusion-minimal $\mathcal{T}$-consistent image of q' in $\mathcal{A}$.*

By the preceding proposition, the causes for an answer $\vec{a}$ to query $q(\vec{x})$ in a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ correspond to the images of the CQs in a UCQ-rewriting of $q(\vec{a})$ in $\mathcal{A}$ that are $\mathcal{T}$-consistent and do not contain any other such image. Based upon this idea, we present in Algorithm 2 a simple procedure ComputeCauses that produces the causes of a tuple $\vec{a}$ being an answer to a query q over a DL-Lite$_\mathcal{R}$ KB. It first computes the conflicts of the KB (line 1), to be

6. Causes were referred to as *minimal $\mathcal{T}$-supports* by Bienvenu and Rosati (2013).

used for checking consistency of the images. The query $q(\vec{x})$ is then rewritten and evaluated over $\mathcal{I}_\mathcal{A}$ (using some existing UCQ-rewriting algorithm UCQRef) with all of its variables treated as free in order to construct the set of its images (lines 4 to 6). The inconsistent or non-minimal images are then discarded (lines 7 to 12). To check the consistency of an image $\mathcal{C}$, the algorithm verifies that $\mathcal{C}$ has an empty intersection with the set $\mathsf{confl}(\mathcal{C}, \mathcal{K})$ of assertions that conflict with it. If the set *Conflicts* computed in line 1 is stored in an appropriate data structure (e.g. as a conflict graph), then it is easy to read off the assertions in $\mathsf{confl}(\mathcal{C}, \mathcal{K})$.

Observe that the size of each computed cause is bounded by the maximum number of atoms in a disjunct of the UCQ-rewriting. Since the rewriting is computed independently of the ABox, the number of potential causes is polynomial in the size of $\mathcal{A}$. It follows that:

Proposition 4.8. ComputeCauses *runs in polynomial time in* $|\mathcal{A}|$ *on input* $\langle \mathcal{T}, \mathcal{A} \rangle$, q, $\vec{a}$.

The algorithm ComputeCauses computes the causes for a Boolean query or a single answer tuple $\vec{a}$. However, by making some minor modifications to the procedure, we can compute causes for all answers to a non-Boolean query $q(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ at the same time. In line 5, we compute answers to the considered disjunct $\psi(\vec{x}, \vec{y})$ of the UCQ-rewriting of $q(\vec{x})$, with variables in both $\vec{x}$ and $\vec{y}$ treated as free. Just after line 6, we insert an additional step where we partition the images so that each group of images assigns the same tuple $\vec{a}$ to the variables $\vec{x}$ (these are the possible causes for $q(\vec{a})$). Finally, for every such tuple $\vec{a}$, we perform lines 7-12 to remove all inconsistent or non-minimal images, thereby obtaining the causes for $q(\vec{a})$.

The preceding algorithms for computing conflicts and causes have the advantage of allowing us to directly use existing UCQ-rewriting methods. Alternatively, we could adopt techniques by Lembo et al. (2011, 2015) and Bienvenu and Rosati (2013) that modify the UCQ-rewriting in order to incorporate the minimality checks directly into the rewriting. This approach yields a better theoretical complexity (AC^0), but the modified rewritings can be larger and more complex, leading to increased evaluation times (Lembo et al., 2015).

4.2 Algorithms for Query Answering under Inconsistency-Tolerant Semantics

We first explain how we can compute the answers that hold under brave and IAR semantics using the procedures ComputeConflicts (Algorithm 1) and ComputeCauses (Algorithm 2) from the preceding subsection, and afterwards we propose a SAT-based approach to handle the AR semantics.

Brave Semantics A Boolean query is entailed under brave semantics just in the case that it is supported by some internally consistent set of facts, i.e. has at least one cause in the KB. Therefore, to decide if a query is entailed under brave semantics, we simply need to compute its causes with ComputeCauses and verify that the output is not empty. For non-Boolean queries, we can use the modified version of ComputeCauses to compute all brave-answers of a non-Boolean query.

IAR Semantics The next proposition[7] provides a characterization of the IAR semantics in terms of causes and conflicts:

Proposition 4.9. *The following are equivalent: (a)* $\mathcal{K} \models_{IAR} q$*, (b) there exists* $\mathcal{C} \in \mathsf{causes}(q, \mathcal{K})$ *such that* $\mathcal{C} \subseteq \bigcap_{\mathcal{R} \in Rep(\mathcal{K})} \mathcal{R}$*, and (c) there exists* $\mathcal{C} \in \mathsf{causes}(q, \mathcal{K})$ *such that* $\mathsf{confl}(\mathcal{C}, \mathcal{K}) = \emptyset$*.*

Proof. To prove (a) implies (b), suppose that $\mathcal{K} \models_{\text{IAR}} q$. Then $\langle \mathcal{T}, \bigcap_{\mathcal{R} \in Rep(\mathcal{K})} \mathcal{R} \rangle \models q$ and $\bigcap_{\mathcal{R} \in Rep(\mathcal{K})} \mathcal{R}$ is $\mathcal{T}$-consistent, so there is some $\mathcal{C} \subseteq \bigcap_{\mathcal{R} \in Rep(\mathcal{K})} \mathcal{R}$ that belongs to $\mathsf{causes}(q, \mathcal{K})$.

To show (b) implies (c), take some $\mathcal{C} \in \mathsf{causes}(q, \mathcal{K})$ such that $\mathcal{C} \subseteq \bigcap_{\mathcal{R} \in Rep(\mathcal{K})} \mathcal{R}$, and suppose for a contradiction that $\alpha \in \mathcal{C} \cap \mathcal{D}$ for some $\mathcal{D} \in \mathsf{conflicts}(\mathcal{K})$. By the minimality of conflicts, $\mathcal{D} \setminus \{\alpha\}$ is $\mathcal{T}$-consistent, and thus can be extended to a repair $\mathcal{R}_\alpha$ that omits α, contradicting our earlier assumption.

For the implication from (c) to (a), suppose that there exists $\mathcal{C} \in \mathsf{causes}(q, \mathcal{K})$ such that $\mathsf{confl}(\mathcal{C}, \mathcal{K}) = \emptyset$. Then $\langle \mathcal{T}, \mathcal{C} \rangle \models q$ and $\mathcal{C} \subseteq \bigcap_{\mathcal{R} \in Rep(\mathcal{K})} \mathcal{R}$, since repairs are maximally consistent and assertions in $\mathcal{C}$ are not involved in any conflicts. We thus obtain $\langle \mathcal{T}, \bigcap_{\mathcal{R} \in Rep(\mathcal{K})} \mathcal{R} \rangle \models q$, i.e. $\mathcal{K} \models_{\text{IAR}} q$. □

By the preceding proposition, a query is entailed under IAR semantics iff there is a cause for the query whose assertions do not participate in any conflict. Therefore, deciding if a query is entailed under IAR semantics can be done by computing its causes with ComputeCauses and the conflicts of the KB with ComputeConflicts, and then checking whether there is a cause without conflicts. For non-Boolean queries, we can again employ the modified version of ComputeCauses to compute all causes of all answers, and then isolate those answers that possess a conflict-free cause to identify the IAR-answers.

AR Semantics For the coNP-complete problem of deciding if a query is entailed under AR semantics, we propose an encoding in terms of propositional unsatisfiability, based upon the following characterization of AR semantics in terms of causes and conflicts:

Proposition 4.10. $\mathcal{K} \not\models_{AR} q$ *iff there exists a* $\mathcal{T}$*-consistent subset* $\mathcal{S} \subseteq \mathcal{A}$ *such that for every* $\mathcal{C} \in \mathsf{causes}(q, \mathcal{K})$*, there is some* $\alpha \in \mathcal{S} \cap \mathsf{confl}(\mathcal{C}, \mathcal{K})$*.*

Proof. First suppose that $\mathcal{K} \not\models_{\text{AR}} q$. By definition, there is some $\mathcal{R} \in Rep(\mathcal{K})$ such that $\langle \mathcal{T}, \mathcal{R} \rangle \not\models q$. Then $\mathcal{R}$ is $\mathcal{T}$-consistent and since it does not entail q, it cannot contain any $\mathcal{C} \in \mathsf{causes}(q, \mathcal{K})$. It follows from the maximality of repairs that for every cause $\mathcal{C}$, the set $\mathcal{R} \cup \mathcal{C}$ is $\mathcal{T}$-inconsistent, and so $\mathcal{R}$ must contain an assertion in conflict with $\mathcal{C}$ since $\mathcal{C}$ is $\mathcal{T}$-consistent by definition.

Conversely, suppose $\mathcal{S} \subseteq \mathcal{A}$ is $\mathcal{T}$-consistent and contains, for every $\mathcal{C} \in \mathsf{causes}(q, \mathcal{K})$, some assertion $\alpha_{\mathcal{C}} \in \mathsf{confl}(\mathcal{C}, \mathcal{K})$. Then there exists a repair $\mathcal{R}$ that extends $\mathcal{S}$ and does not contain any $\mathcal{C} \in \mathsf{causes}(q, \mathcal{K})$ (since it is consistent and contains $\alpha_{\mathcal{C}} \in \mathsf{confl}(\mathcal{C}, \mathcal{K})$). It follows that $\langle \mathcal{T}, \mathcal{R} \rangle \not\models q$, which yields $\mathcal{K} \not\models_{\text{AR}} q$. □

Figure 2 presents the two formulas that are used in our encoding, where variables represent assertions of the ABox. Intuitively, the assertions that correspond to the variables

7. This proposition is similar to Theorem 6 by Lembo et al. (2015). We give a proof here for the sake of completeness and since our formulation differs somewhat.

$$\varphi_{\neg q} = \bigwedge_{\mathcal{C} \in \mathsf{causes}(q,\mathcal{K})} \bigvee_{\beta \in \mathsf{confl}(\mathcal{C},\mathcal{K})} x_\beta \qquad \varphi_{cons} = \bigwedge_{\substack{x_\alpha, x_\beta \in \mathsf{vars}(\varphi_{\neg q}) \\ \beta \in \mathsf{confl}(\{\alpha\},\mathcal{K})}} \neg x_\alpha \vee \neg x_\beta$$

where $\mathsf{vars}(\varphi_{\neg q})$ is the set of variables appearing in $\varphi_{\neg q}$.

Figure 2: SAT encoding $\varphi_{\neg q} \wedge \varphi_{cons}$ for AR query answering.

assigned to true in a truth assignment that satisfies $\varphi_{\neg q} \wedge \varphi_{cons}$ form a consistent subset of the ABox that contains at least one assertion of the conflicts of each cause of the query. By Proposition 4.10, such a subset exists just in the case that $\mathcal{K} \not\models_{\text{AR}} q$.

Theorem 4.11. *Let $\mathcal{K}$ be a DL-Lite$_\mathcal{R}$ KB and q be a Boolean conjunctive query. $\mathcal{K} \not\models_{AR} q$ if and only if $\varphi_{\neg q} \wedge \varphi_{cons}$ is satisfiable, where $\varphi_{\neg q}$ and φ_{cons} are defined in Figure 2.*

Proof. Let ν be a truth assignment of the variables of $\varphi_{\neg q} \wedge \varphi_{cons}$ and $\mathcal{R}_\nu = \{\beta \mid \nu(x_\beta) = \mathsf{true}\}$. The formula $\varphi_{\neg q}$ evaluates to true in ν if and only if for every cause $\mathcal{C}$ of q, there exists an assertion β which is in conflict with some assertion of $\mathcal{C}$ and such that $\nu(x_\beta) = \mathsf{true}$, so is in $\mathcal{R}_\nu$. The formula φ_{cons} evaluates to true in ν if and only if there is no α, β which are in a conflict and such that $\nu(x_\alpha) = \mathsf{true}$ and $\nu(x_\beta) = \mathsf{true}$, so if and only if $\mathcal{R}_\nu$ does not contain any conflict of $\mathcal{K}$, i.e. is $\mathcal{T}$-consistent. Hence $\varphi_{\neg q} \wedge \varphi_{cons}$ evaluates to true in ν if and only if $\mathcal{R}_\nu$ is a consistent subset of $\mathcal{A}$ that contradicts each cause for q. We conclude by Proposition 4.10 that $\varphi_{\neg q} \wedge \varphi_{cons}$ is satisfiable if and only if $\mathcal{K} \not\models_{\text{AR}} q$. □

The following example illustrates the encoding.

Example 13. For the Boolean query $\mathsf{PhD}(ann)$, which has four causes, $\{\mathsf{AProf}(ann)\}$, $\{\mathsf{FProf}(ann)\}$, $\{\mathsf{Advise}(ann, bob)\}$ and $\{\mathsf{Postdoc}(ann)\}$, we have:

$$\begin{aligned}
\mathsf{confl}(\{\mathsf{AProf}(ann)\}, \mathcal{K}) &= \{\mathsf{FProf}(ann), \mathsf{Postdoc}(ann)\} \\
\mathsf{confl}(\{\mathsf{FProf}(ann)\}, \mathcal{K}) &= \{\mathsf{AProf}(ann), \mathsf{Postdoc}(ann)\} \\
\mathsf{confl}(\{\mathsf{Advise}(ann, bob)\}, \mathcal{K}) &= \{\mathsf{Postdoc}(ann)\} \\
\mathsf{confl}(\{\mathsf{Postdoc}(ann)\}, \mathcal{K}) &= \{\mathsf{AProf}(ann), \mathsf{FProf}(ann), \mathsf{Advise}(ann, bob)\}
\end{aligned}$$

so the encoding is as follows:

$$\begin{aligned}
\varphi_{\neg q} =& (x_{\mathsf{FProf}(a)} \vee x_{\mathsf{Postdoc}(a)}) \wedge (x_{\mathsf{AProf}(a)} \vee x_{\mathsf{Postdoc}(a)}) \wedge x_{\mathsf{Postdoc}(a)} \wedge \\
& (x_{\mathsf{AProf}(a)} \vee x_{\mathsf{FProf}(a)} \vee x_{\mathsf{Advise}(a,b)}) \\
\varphi_{cons} =& (\neg x_{\mathsf{FProf}(a)} \vee \neg x_{\mathsf{Postdoc}(a)}) \wedge (\neg x_{\mathsf{AProf}(a)} \vee \neg x_{\mathsf{Postdoc}(a)}) \wedge (\neg x_{\mathsf{Advise}(a,b)} \vee \neg x_{\mathsf{Postdoc}(a)}) \wedge \\
& (\neg x_{\mathsf{AProf}(a)} \vee \neg x_{\mathsf{FProf}(a)})
\end{aligned}$$

Since $x_{\mathsf{Postdoc}(a)}$ cannot be assigned to true together with $x_{\mathsf{AProf}(a)}$, $x_{\mathsf{FProf}(a)}$, or $x_{\mathsf{Advise}(a,b)}$, the formula $\varphi_{\neg q} \wedge \varphi_{cons}$ is unsatisfiable, so $\langle \mathcal{T}, \mathcal{A} \rangle \models_{\text{AR}} \mathsf{PhD}(ann)$.

If we consider instead the query $\mathsf{Prof}(ann)$, we obtain the following encoding:

$$\varphi_{\neg q} = (x_{\mathsf{FProf}(a)} \vee x_{\mathsf{Postdoc}(a)}) \wedge (x_{\mathsf{AProf}(a)} \vee x_{\mathsf{Postdoc}(a)}) \wedge x_{\mathsf{Postdoc}(a)}$$

$$\varphi_{cons} = (\neg x_{\mathsf{FProf}(a)} \vee \neg x_{\mathsf{Postdoc}(a)}) \wedge (\neg x_{\mathsf{AProf}(a)} \vee \neg x_{\mathsf{Postdoc}(a)}) \wedge (\neg x_{\mathsf{AProf}(a)} \vee \neg x_{\mathsf{FProf}(a)})$$

A valuation that assigns $x_{\mathsf{Postdoc}(a)}$ to true and $x_{\mathsf{AProf}(a)}$ and $x_{\mathsf{FProf}(a)}$ to false satisfies $\varphi_{\neg q} \wedge \varphi_{cons}$, so $\langle \mathcal{T}, \mathcal{A} \rangle \not\models_{\mathrm{AR}} \mathsf{Prof}(ann)$. ◁

The interest of the proposed encoding is that it only has as many variables as the number of assertions which are in a conflict with a cause of the query, whereas a naïve encoding that determines a complete repair would need one variable per ABox assertion. Thus, in practice, the size of our encoding may be substantially smaller than the size of the ABox.

By Theorem 4.11, to decide whether $\mathcal{K} \models_{\mathrm{AR}} q$, it suffices to construct the propositional encoding and pass it on to a SAT solver. However, by exploiting the brave and IAR semantics, it is sometimes possible to determine whether a query holds under AR semantics without utilizing the encoding. Indeed, by Proposition 3.9, $\mathcal{K} \not\models_{\mathrm{brave}} q$ implies $\mathcal{K} \not\models_{\mathrm{AR}} q$ and $\mathcal{K} \models_{\mathrm{IAR}} q$ implies $\mathcal{K} \models_{\mathrm{AR}} q$. Thus, even if one were only interested in the AR semantics, it is advantageous to employ the brave and IAR semantics as tractable upper and lower approximations.

Putting everything together, we propose the algorithm ClassifyQuery (Algorithm 3) that determines the strongest semantics (among IAR, AR, brave) under which a Boolean query holds (see next paragraph for discussion of the non-Boolean case). As a first step, ClassifyQuery computes and stores the KB's conflicts (line 1) and uses them to identify the set *ConflAssertions* of assertions that belong to some conflict (line 2). It next generates the causes of q w.r.t. the KB (line 3). If the set of causes is empty, then the query is not entailed under brave semantics, so 'not brave' is output (lines 4-5). Otherwise, each cause is considered in turn (lines 7-12). If a cause has an empty intersection with *ConflAssertions*, then it is entailed under IAR semantics, so we immediately return 'IAR'; else we construct the set of conflicts for that cause and add it to *CausesConfl*. In line 13, we generate the encoding from Figure 2, using the sets of assertions in *CausesConfl* to produce the clauses in $\varphi_{\neg q}$ and the sets of assertions in *Conflicts* to construct the clauses in φ_{cons}. If the encoding is unsatisfiable (as determined by a SAT solver), then by Theorem 2, the query is entailed under AR semantics, so we output 'AR' (line 14). Otherwise, we output 'brave' (line 16), as we have determined the query to be brave (as it has at least one cause) but not to hold under AR semantics. It should be clear that the output of the algorithm indeed corresponds to the strongest semantics under which the query q is entailed.

For a non-Boolean query, the algorithm ClassifyQuery can be slightly modified to classify all tuples that hold under at least one of the semantics. First, we use the non-Boolean version of ComputeCauses to generate the causes for all assignments $\vec{a}$ to the answer variables $\vec{x}$ that admit at least one cause. Note that every such tuple $\vec{a}$ is a brave-answer, and it remains to test whether it holds under IAR or AR semantics. We skip over lines 4-5 as we do not wish to list all tuples that are not brave-answers. Instead, we iterate over all brave-answers $\vec{a}$, and for every such tuple, we perform the operations in lines 7-16, using the computed set of causes for $\vec{a}$. The only other minor modification is that instead of outputting 'IAR' / 'AR', or 'brave', we add tuples to one of the sets IAR, AR, or $brave$, which are returned at the end of the execution. We point out that for non-Boolean queries, the number of tuples that need to be classified can be huge, and thus it is all the more essential to use the brave and IAR semantics to reduce the total number of calls that need to be made to the SAT solver.

ALGORITHM 3: ClassifyQuery

Input: a conjunctive query $q(\vec{x})$, a candidate answer $\vec{a}$, a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: **IAR** if $\langle \mathcal{T}, \mathcal{A} \rangle \models_{\text{IAR}} q(\vec{a})$, **AR** if $\langle \mathcal{T}, \mathcal{A} \rangle \models_{\text{AR}} q(\vec{a})$ and $\langle \mathcal{T}, \mathcal{A} \rangle \not\models_{\text{IAR}} q(\vec{a})$, **brave** if $\langle \mathcal{T}, \mathcal{A} \rangle \models_{\text{brave}} q(\vec{a})$ and $\langle \mathcal{T}, \mathcal{A} \rangle \not\models_{\text{AR}} q(\vec{a})$, and **not brave** otherwise

```
Conflicts ← ComputeConflicts(T, A);
ConflAssertions ← ⋃_{B ∈ Conflicts} B;
Causes ← ComputeCauses(q(a), T, A);
if Causes = ∅ then
    Output not brave
CausesConfl ← ∅;
foreach C ∈ Causes do
    if C ∩ ConflAssertions = ∅ then
        Output IAR
    else
        CausesConfl ← CausesConfl ∪ {ConflictsFor(C, K)}
φ ← ConstructEncoding(CausesConfl, Conflicts);
if Unsat(φ) then                                   /* call to a SAT solver */
    Output AR
else
    Output brave
```

We close this section with a remark on how our SAT-based approach can be extended to handle variants of DL-Lite allowing for n-ary conflicts, such as the language DL-Lite$_{A,id,den}$ considered by Lembo et al. (2015).

Remark 4.12. If we allow in the TBox denial constraints specified using either inclusions of the form $B_1 \sqcap ... \sqcap B_n \sqsubseteq \bot$ or Boolean CQs (which describe situations which should not hold), then conflicts may contain more than two assertions. When conflicts are not binary, we cannot define the conflicts of a cause as a set of assertions, as it may be necessary to use several assertions to contradict a cause. The encoding can be adapted to take into account n-ary conflicts as follows: $\mathcal{K} \not\models_{\text{AR}} q$ iff $\varphi^1_{\neg q} \wedge \varphi^2_{\neg q} \wedge \varphi_{cons}$ is satisfiable, where:

$$\varphi^1_{\neg q} = \bigwedge_{\mathcal{C} \in \mathsf{causes}(q,\mathcal{K})} \bigvee_{\mathcal{B} \in \mathsf{conflicts}(\mathcal{K}), \mathcal{C} \cap \mathcal{B} \neq \emptyset} x_{\mathcal{C},\mathcal{B}}$$

$$\varphi^2_{\neg q} = \bigwedge_{x_{\mathcal{C},\mathcal{B}} \in \mathsf{vars}(\varphi^1_{\neg q})} \bigwedge_{\beta \in \mathcal{B} \setminus \mathcal{C}} \neg x_{\mathcal{C},\mathcal{B}} \vee x_\beta$$

$$\varphi_{cons} = \bigwedge_{x_\alpha \in \mathsf{vars}(\varphi^2_{\neg q})} \bigwedge_{\mathcal{B} \in \mathsf{conflicts}(\mathcal{K}), \alpha \in \mathcal{B}} \bigvee_{\beta \in \mathcal{B}} \neg x_\beta$$

The new variables $x_{\mathcal{C},\mathcal{B}}$ represent the different ways of contradicting $\mathcal{C}$, and $\varphi^1_{\neg q}$ expresses that every cause is contradicted. The formula $\varphi^2_{\neg q}$ ensures that when $x_{\mathcal{C},\mathcal{B}}$ is assigned to true, which means that $\mathcal{C}$ is contradicted with the conflict $\mathcal{B}$, every assertion of $\mathcal{B}$ which does not belong to $\mathcal{C}$ is selected, so that adding $\mathcal{B} \setminus \mathcal{C}$ creates a conflict. As in the original encoding, φ_{cons} enforces consistency of the subset consisting of the assertions whose corresponding variables are assigned to true by preventing all assertions of a conflict to be selected together.

For example, consider the TBox $\mathcal{T} = \{\mathsf{Postdoc} \sqsubseteq \mathsf{PhD}, \mathsf{Prof} \sqsubseteq \mathsf{PhD}, \mathsf{Postdoc} \sqsubseteq \neg\mathsf{Prof}\}$ extended with the constraint that $\exists xy\, \mathsf{Postdoc}(x) \wedge \mathsf{Advise}(x, y) \wedge \mathsf{PhD}(y)$ should not hold and the ABox $\mathcal{A} = \{\mathsf{Postdoc}(ann), \mathsf{Prof}(ann), \mathsf{Postdoc}(bob), \mathsf{Advise}(ann, bob)\}$. The resulting KB is inconsistent and has two conflicts $\mathcal{B}_1 = \{\mathsf{Postdoc}(ann), \mathsf{Prof}(ann)\}$ and $\mathcal{B}_2 = \{\mathsf{Postdoc}(ann), \mathsf{Postdoc}(bob), \mathsf{Advise}(ann, bob)\}$. The Boolean query $\mathsf{PhD}(ann)$ has two causes $\mathcal{C}_1 = \{\mathsf{Postdoc}(ann)\}$ and $\mathcal{C}_2 = \{\mathsf{Prof}(ann)\}$ and the encoding is as follows:

$$\varphi^1_{\neg q} = (x_{\mathcal{C}_1,\mathcal{B}_1} \vee x_{\mathcal{C}_1,\mathcal{B}_2}) \wedge x_{\mathcal{C}_2,\mathcal{B}_1}$$
$$\varphi^2_{\neg q} = (\neg x_{\mathcal{C}_1,\mathcal{B}_1} \vee x_{\mathsf{Prof}(a)}) \wedge (\neg x_{\mathcal{C}_1,\mathcal{B}_2} \vee x_{\mathsf{Postdoc}(b)}) \wedge (\neg x_{\mathcal{C}_1,\mathcal{B}_2} \vee x_{\mathsf{Advise}(a,b)}) \wedge (\neg x_{\mathcal{C}_2,\mathcal{B}_1} \vee x_{\mathsf{Postdoc}(a)})$$
$$\varphi_{cons} = (\neg x_{\mathsf{Postdoc}(a)} \vee \neg x_{\mathsf{Prof}(a)}) \wedge (\neg x_{\mathsf{Postdoc}(a)} \vee \neg x_{\mathsf{Postdoc}(b)} \vee \neg x_{\mathsf{Advise}(a,b)})$$

The formula $\varphi^1_{\neg q} \wedge \varphi^2_{\neg q} \wedge \varphi_{cons}$ is unsatisfiable, so $\langle \mathcal{T}, \mathcal{A} \rangle \models_{\mathrm{AR}} \mathsf{PhD}(ann)$.

5. Explaining Inconsistency-Tolerant Query Answering

In the preceding section, we proposed to use conjointly the brave, AR and IAR semantics to identify answers of different levels of confidence. We now turn our attention to the problem of explaining the obtained query results. Our goal is to improve the usability of inconsistency-tolerant querying systems by helping the user understand the classification of a particular tuple, e.g. why is $\vec{a}$ an AR-answer, and why is it not an IAR-answer? To this end, we introduce notions of explanation for positive and negative query answers under brave, AR, and IAR semantics, and we investigate the computational properties of the resulting explanation framework.

A proof-theoretic approach to explaining positive answers to CQs over consistent DL-Lite$_\mathcal{A}$ KBs was introduced by Borgida et al. (2008). It outputs a single proof, involving both TBox axioms and ABox assertions, that is generated by 'tracing back' the relevant part of the rewritten query, using minimality criteria to select a 'simplest' proof. For negative answers, explanations for why a tuple is not an answer to a CQ are defined by Calvanese et al. (2013) as sets of ABox assertions that can be added to the ABox to make the tuple become an answer. Practical algorithms and an implementation for computing such explanations were described by Du et al. (2014). The latter work was recently extended to the case of inconsistent KBs (Du et al., 2015): essentially the idea is to add a set of ABox assertions that will lead to the answer holding under IAR semantics (in particular, the new assertions must not introduce any inconsistencies).

5.1 Explanations for Query (Non-)Answers

Existing frameworks for explaining (non-)answers to queries over DL KBs are ill adapted to the setting of inconsistency-tolerant query answering. Indeed, it is no longer enough to prove that positive answers are entailed by some part of the ABox together with the TBox (as done by Borgida et al., 2008) to establish that they hold in every repair, as required by AR semantics. Moreover, negative answers do not result from the absence of supporting facts anymore (like in the works by Calvanese et al., 2013; Du et al., 2014), but rather from the presence of conflicting assertions.

As explained in the introduction, this paper targets scenarios in which inconsistencies are due to errors in the ABox, and so understanding the link between (possibly faulty)

ABox assertions and query results is especially important. For this reason, we choose to focus on ABox assertions, rather than TBox axioms. The explanations we consider will therefore take either the form of a set of ABox assertions (viewed as a conjunction) or a set of sets of assertions (interpreted as a disjunction of conjunctions). We shall see that our 'ABox-centric' explanation framework already poses non-trivial computational challenges. To get more detailed explanations, which also take into account the TBox reasoning that led to the results, our approach could be combined with that of Borgida et al. (2008).

We start by considering what are arguably the simplest answers to explain: positive brave- and IAR-answers. Indeed, we have seen that an answer holds under brave semantics just in the case that it possesses some cause, and under IAR semantics just in the case that it has some cause whose assertions do not belong to any conflict. It is therefore natural to use the query's causes as explanations for brave-answers, and the causes that do not participate in any contradiction for IAR-answers.

Definition 5.1 (Explanations for positive brave-answers)**.** A subset $\mathcal{C} \subseteq \mathcal{A}$ is an *explanation for* $\mathcal{K} \models_{brave} q(\vec{a})$ iff $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \mathcal{K})$.

Definition 5.2 (Explanations for positive IAR-answers)**.** A subset $\mathcal{C} \subseteq \mathcal{A}$ is an *explanation for* $\mathcal{K} \models_{IAR} q(\vec{a})$ iff $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \mathcal{K})$ and $\mathcal{C} \subseteq \mathcal{R}$ for every repair $\mathcal{R}$ of $\mathcal{K}$ (equivalently: $\mathsf{confl}(\mathcal{C}, \mathcal{K}) = \emptyset$).

Example 14. There are three causes for $q_1(ann) = \mathsf{Prof}(ann)$:

$$\{\mathsf{AProf}(ann)\} \qquad \{\mathsf{FProf}(ann)\} \qquad \{\mathsf{Advise}(ann, bob)\}$$

and these give us the explanations for $\mathcal{K} \models_{\text{brave}} q_1(ann) = \mathsf{Prof}(ann)$. Observe that each cause is in conflict with an assertion in $\mathcal{A}^*$, so there are no explanations for $\mathcal{K} \models_{\text{IAR}} q_1(ann)$.

If we consider instead $q_3(ann) = \exists y\, \mathsf{Teach}(ann, y)$, then again we have three causes:

$$\{\mathsf{Teach}(ann, c_1)\} \qquad \{\mathsf{Teach}(ann, c_2)\} \qquad \{\mathsf{Teach}(ann, c_3)\}$$

However, each of these causes is without conflict, so they are both explanations for $\mathcal{K} \models_{\text{brave}} q_3(ann)$ and explanations for $\mathcal{K} \models_{\text{IAR}} q_3(ann)$. ◁

To explain why a tuple is an AR-answer, it is no longer sufficient to give a single cause, since the answer may be supported by different causes in different repairs. We will therefore define explanations as (minimal) disjunctions of causes that 'cover' all repairs.

Definition 5.3 (Explanations for AR-answers)**.** An explanation for $\mathcal{K} \models_{\text{AR}} q(\vec{a})$ is a set $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\} \subseteq \mathsf{causes}(q(\vec{a}), \mathcal{K})$ such that (i) every repair $\mathcal{R}$ of $\mathcal{K}$ contains some $\mathcal{C}_i$, and (ii) no proper subset of $\mathcal{E}$ satisfies this property.

Example 15. There are 36 explanations for $\mathcal{K} \models_{\text{AR}} \exists y\, \mathsf{PhD}(ann) \wedge \mathsf{Teach}(ann, y)$, each taking one of the following two forms:

$$\begin{aligned}
\mathcal{E}_{ij} =& (\mathsf{Postdoc}(ann) \wedge \mathsf{Teach}(ann, c_i)) \vee (\mathsf{Advise}(ann, bob) \wedge \mathsf{Teach}(ann, c_j)) \\
\mathcal{E}'_{ijk} =& (\mathsf{Postdoc}(ann) \wedge \mathsf{Teach}(ann, c_i)) \vee (\mathsf{AProf}(ann) \wedge \mathsf{Teach}(ann, c_j)) \\
& \vee (\mathsf{FProf}(ann) \wedge \mathsf{Teach}(ann, c_k))
\end{aligned}$$

for some $i, j, k \in \{1, 2, 3\}$. Indeed, for repair $\mathcal{R}_1$, we must choose a cause of the form $\mathsf{Postdoc}(ann) \wedge \mathsf{Teach}(ann, c_i)$, and to cover the repairs $\mathcal{R}_2$ and $\mathcal{R}_3$, we may either include a cause of the form $\mathsf{Advise}(ann, bob) \wedge \mathsf{Teach}(ann, c_j)$ or a pair of causes of the forms $\mathsf{AProf}(ann) \wedge \mathsf{Teach}(ann, c_j)$ (for $\mathcal{R}_2$) and $\mathsf{FProf}(ann) \wedge \mathsf{Teach}(ann, c_k)$ (for $\mathcal{R}_3$). ◁

We remark that when restricted to consistent KBs, all three notions of explanation for positive answers coincide with the inclusion-minimal subsets of the ABox that, when combined with the TBox, yield the query answer. Such a notion of explanation, defined as a minimal subset that permits the derivation of a target consequence, is broadly similar to the notion of justifications widely used for explaining TBox reasoning (see Section 8 for more discussion of justifications and references).

We next turn to the problem of explaining why a tuple that is returned as a brave-answer does not hold under one of the stronger semantics (for non-brave answers, one can adapt existing work on query abduction, see Section 8.3.3). Brave-answers that do not hold under AR (resp. IAR) semantics will be called *negative AR-answers (resp. IAR-answers)*.

To explain negative AR-answers, a first idea might be to provide a repair in which the query answer does not hold, but this could yield very large explanations with lots of irrelevant assertions. Instead, we propose to give a minimal subset of the ABox that is consistent with the TBox and contradicts every cause of the query, since any such subset can be extended to a repair that omits all causes. For IAR semantics, the formulation is slightly different as we only need to ensure that every cause is contradicted by some consistent subset of the ABox, as this shows that no cause belongs to all repairs.

Definition 5.4 (Explanations for negative AR-answers)**.** An explanation for $\mathcal{K} \not\models_{\mathrm{AR}} q(\vec{a})$ is a $\mathcal{T}$-consistent subset $\mathcal{E} \subseteq \mathcal{A}$ such that: (i) $\langle \mathcal{T}, \mathcal{E} \cup \mathcal{C} \rangle \models \bot$ for every $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \mathcal{K})$, (ii) no proper subset of $\mathcal{E}$ has this property.

Example 16. The unique explanation for $\mathcal{K} \not\models_{\mathrm{AR}} \mathsf{Prof}(ann)$ is $\{\mathsf{Postdoc}(ann)\}$, which contradicts the three causes of $q_1(ann) = \mathsf{Prof}(ann)$. ◁

Definition 5.5 (Explanations for negative IAR-answers)**.** An explanation for $\mathcal{K} \not\models_{\mathrm{IAR}} q(\vec{a})$ is a (possibly $\mathcal{T}$-inconsistent) subset $\mathcal{E} \subseteq \mathcal{A}$ such that: (i) for every $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \mathcal{K})$, there exists a $\mathcal{T}$-consistent subset $\mathcal{E}' \subseteq \mathcal{E}$ with $\langle \mathcal{T}, \mathcal{E}' \cup \mathcal{C} \rangle \models \bot$, (ii) no proper subset of $\mathcal{E}$ has this property.

We remark that for DL-Lite$_{\mathcal{R}}$ and other DLs admitting only binary conflicts, point (i) of the preceding definition can be simplified, as the set $\mathcal{E}'$ can be assumed to be a singleton.

Example 17. Reconsider the query $q_2(x) = \exists y \mathsf{PhD}(x) \wedge \mathsf{Teach}(x, y)$. There are 3 explanations for $\mathcal{K} \not\models_{\mathrm{IAR}} q_2(ann)$: $\{\mathsf{AProf}(ann), \mathsf{Postdoc}(ann)\}$, $\{\mathsf{FProf}(ann), \mathsf{Postdoc}(ann)\}$, and $\{\mathsf{Advise}(ann, bob), \mathsf{Postdoc}(ann)\}$, where the first assertion of each explanation contradicts the causes of $\exists y\, \mathsf{PhD}(x) \wedge \mathsf{Teach}(ann, y)$ that contain $\mathsf{Postdoc}(ann)$, and the second one contradicts those that contain $\mathsf{AProf}(ann)$, $\mathsf{FProf}(ann)$ or $\mathsf{Advise}(ann, bob)$. ◁

The following example illustrates that explanations are more informative than causes and conflicts, since some causes and conflicts may not be involved in the explanations of an answer. It also shows that explanations for negative answers should be accompanied with the explanations for being a brave-answer (i.e. the causes), because otherwise they may be difficult to understand.

Example 18. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ defined as follows:

$$\begin{aligned}
\mathcal{T} = \{&\exists \mathsf{Advise} \sqsubseteq \mathsf{Prof}, \mathsf{Prof} \sqsubseteq \mathsf{Employee}, \mathsf{Postdoc} \sqsubseteq \mathsf{Employee}, \mathsf{Prof} \sqsubseteq \neg \mathsf{Postdoc},\\
&\exists \mathsf{WorkFor} \sqsubseteq \mathsf{Employee}, \exists \mathsf{WorkFor}^- \sqsubseteq \mathsf{Department}, \mathsf{Employee} \sqsubseteq \neg \mathsf{Department},\\
&\exists \mathsf{TakeCourse}^- \sqsubseteq \mathsf{Course}, \exists \mathsf{Advise}^- \sqsubseteq \mathsf{Person}, \mathsf{Course} \sqsubseteq \neg \mathsf{Person}\}\\
\mathcal{A} = \{&\mathsf{Postdoc}(ann), \mathsf{Advise}(ann, bob), \mathsf{Advise}(ann, carl), \mathsf{Teach}(ann, c_1),\\
&\mathsf{TakeCourse}(c_2, carl), \mathsf{WorkFor}(ann, dpt), \mathsf{WorkFor}(dpt, dan)\}
\end{aligned}$$

This KB has the following four conflicts:

$$\{\mathsf{Postdoc}(ann), \mathsf{Advise}(ann, bob)\} \qquad \{\mathsf{Postdoc}(ann), \mathsf{Advise}(ann, carl)\},$$
$$\{\mathsf{Advise}(ann, carl), \mathsf{TakeCourse}(c_2, carl)\} \qquad \{\mathsf{WorkFor}(ann, dpt), \mathsf{WorkFor}(dpt, dan)\}$$

giving rise to the following repairs:

$$\begin{aligned}
\mathcal{R}'_1 &= \{\mathsf{Postdoc}(ann), \mathsf{Teach}(ann, c_1), \mathsf{TakeCourse}(c_2, carl), \mathsf{WorkFor}(ann, dpt)\}\\
\mathcal{R}'_2 &= \{\mathsf{Advise}(ann, bob), \mathsf{Advise}(ann, carl), \mathsf{Teach}(ann, c_1), \mathsf{WorkFor}(ann, dpt)\}\\
\mathcal{R}'_3 &= \{\mathsf{Advise}(ann, bob), \mathsf{TakeCourse}(c_2, carl), \mathsf{Teach}(ann, c_1), \mathsf{WorkFor}(ann, dpt)\}\\
\mathcal{R}'_4 &= \{\mathsf{Postdoc}(ann), \mathsf{Teach}(ann, c_1), \mathsf{TakeCourse}(c_2, carl), \mathsf{WorkFor}(dpt, dan)\}\\
\mathcal{R}'_5 &= \{\mathsf{Advise}(ann, bob), \mathsf{Advise}(ann, carl), \mathsf{Teach}(ann, c_1), \mathsf{WorkFor}(dpt, dan)\}\\
\mathcal{R}'_6 &= \{\mathsf{Advise}(ann, bob), \mathsf{TakeCourse}(c_2, carl), \mathsf{Teach}(ann, c_1), \mathsf{WorkFor}(dpt, dan)\}
\end{aligned}$$

If we evaluate the query $q(x, y) = \mathsf{Employee}(x) \wedge \mathsf{Teach}(x, y)$ using AR semantics, we find that $\mathcal{K} \models_{\mathrm{AR}} q(ann, c_1)$. There are four causes of $q(ann, c_1)$ in $\mathcal{K}$:

$$\{\mathsf{Postdoc}(ann), \mathsf{Teach}(ann, c_1)\} \qquad \{\mathsf{Advise}(ann, bob), \mathsf{Teach}(ann, c_1)\}$$
$$\{\mathsf{Advise}(ann, carl), \mathsf{Teach}(ann, c_1)\} \qquad \{\mathsf{WorkFor}(ann, dpt), \mathsf{Teach}(ann, c_1)\}$$

but only one explanation for $\mathcal{K} \models_{\mathrm{AR}} q(ann, c_1)$:

$$\{\{\mathsf{Postdoc}(ann), \mathsf{Teach}(ann, c_1)\}, \{\mathsf{Advise}(ann, bob), \mathsf{Teach}(ann, c_1)\}\}$$

To see why the cause $\{\mathsf{Advise}(ann, carl), \mathsf{Teach}(ann, c_1)\}$ does not participate in any explanation, observe that if $\mathcal{R}$ is a repair that contains $\{\mathsf{Advise}(ann, carl), \mathsf{Teach}(ann, c_1)\}$, then $\mathcal{R}' = (\mathcal{R} \backslash \{\mathsf{Advise}(ann, carl)\}) \cup \{\mathsf{TakeCourse}(c_2, carl)\}$ is also a repair. As every explanation contains some cause $\mathcal{C}$ included in $\mathcal{R}'$, and $\mathsf{TakeCourse}(c_2, carl)$ does not belong to any cause, it follows that $\mathcal{C} \subseteq \mathcal{R}$, so $\{\mathsf{Advise}(ann, carl), \mathsf{Teach}(ann, c_1)\}$ is not needed to 'cover' $\mathcal{R}$. For a similar reason, the cause $\{\mathsf{WorkFor}(ann, dpt), \mathsf{Teach}(ann, c_1)\}$ does not belong to any explanation $\mathcal{K} \models_{\mathrm{AR}} q(ann, c_1)$.

From the fact that we need two causes to cover the repairs, we can derive that $q(ann, c_1)$ does not hold under IAR semantics. There are two explanations for $\mathcal{K} \not\models_{\mathrm{IAR}} q(ann, c_1)$:

$$\{\mathsf{WorkFor}(dpt, dan), \mathsf{Postdoc}(ann), \mathsf{Advise}(ann, bob)\}$$
$$\{\mathsf{WorkFor}(dpt, dan), \mathsf{Postdoc}(ann), \mathsf{Advise}(ann, carl)\}$$

Note that $\mathsf{TakeCourse}(c_2, carl)$ is not involved in the explanations of $\mathcal{K} \not\models_{\text{IAR}} q(ann, c_1)$, even though it conflicts with the cause $\{\mathsf{Advise}(ann, carl), \mathsf{Teach}(ann, c_1)\}$. This is because $\mathsf{Postdoc}(ann)$ is also a conflict of this cause and is the only assertion that contradicts the other cause $\{\mathsf{Advise}(ann, bob), \mathsf{Teach}(ann, c_1)\}$. It follows that each set of assertions that contradicts every cause must contain $\mathsf{Postdoc}(ann)$, and $\mathsf{TakeCourse}(c_2, carl)$ is not needed (so will not appear in any minimal such set).

If a user receives the explanations for $\mathcal{K} \not\models_{\text{IAR}} q(ann, c_1)$ without the causes of $q(ann, c_1)$, it can be very hard for him to figure out why $\mathsf{WorkFor}(dpt, dan)$ is relevant. Indeed, there is no obvious relation between this assertion and the answer (ann, c_1), since $\mathsf{WorkFor}(dpt, dan)$ does not involve any of the individuals of the answer. Even if the explanations for $\mathcal{K} \models_{\text{AR}} q(ann, c_1)$ are provided, it would not help because neither dpt nor dan appears in them. To understand why $\mathsf{WorkFor}(dpt, dan)$ is part of an explanation for $\mathcal{K} \not\models_{\text{IAR}} q(ann, c_1)$, the user needs to be aware of the cause $\{\mathsf{WorkFor}(ann, dpt), \mathsf{Teach}(ann, c_1)\}$. A solution would be to accompany an explanation for a negative IAR-answer with (one or more) causes that are conflicted by the assertions in that explanation, or alternatively to allow users to ask why a given assertion α appears in an explanation (in which case, cause(s) conflicted by α could be displayed). ◁

When there are a large number of explanations for a given answer, it may be impractical to present them all to the user. In such cases, instead of presenting all explanations, one may choose to rank the explanations according to some preference criteria, and to present one or a small number of most *preferred explanations*. In this work, we will use *cardinality* to rank explanations for brave- and IAR-answers and negative AR- and IAR-answers. For positive AR-answers, we consider two ranking criteria: the *number of disjuncts*, and the total *number of assertions*. Another interesting criterion would be the difficulty of the associated TBox reasoning. For example, we may compute for each cause the minimum number of TBox axioms needed to show that the cause yields the query, and then use this number to rank explanations for brave- and IAR-answers.

Example 19. Reconsider explanations $\mathcal{E}_{1\,1}$ and $\mathcal{E}'_{1\,2\,3}$ for $\mathcal{K} \models_{\text{AR}} q_2(ann)$ from Example 15. There are at least two reasons why $\mathcal{E}_{1\,1}$ may be considered easier to understand than $\mathcal{E}'_{1\,2\,3}$. First, $\mathcal{E}_{1\,1}$ contains fewer disjuncts, hence requires less disjunctive reasoning. Second, both disjuncts of $\mathcal{E}_{1\,1}$ use the same Teach assertion, whereas $\mathcal{E}'_{1\,2\,3}$ uses three different Teach assertions, which may lead the user to (wrongly) believe all are needed to obtain the query result. Preferring explanations having the fewest number of disjuncts, and among them, those involving a minimal set of assertions, leads to focusing on the explanations of the form $\mathcal{E}_{i\,i}$, where $i \in \{1, 2, 3\}$. ◁

A second complementary approach to dealing with a large number of explanations is to concisely summarize the set of explanations in terms of the *necessary assertions* (i.e. appearing in every explanation) and the *relevant assertions* (i.e. appearing in at least one explanation). The advantage of this approach is to present the whole information to the user without overwhelming him with all explanations. This is especially relevant in the case of positive AR and negative answers, where the number of explanations may be exponential in the size of the ABox because of the combination of the different causes for AR-answers, and ways of contradicting each cause for negative answers. Indeed, the set of conflicts of

	brave, IAR	**AR**	**neg. AR**	**neg. IAR**
genOne	in PTime	NP-h	NP-h	in PTime
genBest†	in PTime	Σ_2^p-h‡	NP-h	NP-h*
rel	in AC^0	Σ_2^p-co	NP-co	in AC^0
nec	in AC^0	NP-co	coNP-co	in AC^0
rec	in AC^0	BH_2-co	in AC^0	in AC^0
best rec†	in PTime	Π_2^p-co‡	coNP-co*	coNP-co*

† upper bounds hold for ranking criteria that can be decided in PTime
‡ lower bounds hold for smallest disjunction or fewest assertions
* lower bounds hold for cardinality-minimal explanations

Table 1: Data complexity results for explanations of query answers in DL-Lite$_{\mathcal{R}}$.

each cause can be as large as the ABox. Even for positive IAR or brave-answers, the number of explanations can be very large (imagine for instance a query that retrieves the persons who teach something, advise some students and have some publications).

Example 20. If we tweak the example KB to include n courses taught by ann, then there would be $n^2 + n^3$ explanations of the form $\mathcal{E}_{ij}$ and $\mathcal{E}'_{ijk}$ for $\mathcal{K} \models_{\mathrm{AR}} \exists y \mathsf{PhD}(ann) \wedge \mathsf{Teach}(ann, y)$, built using only $n + 4$ assertions. Presenting the necessary assertions (here: $\mathsf{Postdoc}(ann)$) and the relevant assertions ($\mathsf{AProf}(ann)$, $\mathsf{FProf}(ann)$, $\mathsf{Advise}(ann, bob)$, and $\mathsf{Teach}(ann, c_j)$ for $1 \leq j \leq n$) gives a succinct overview of the set of explanations. ◁

5.2 Complexity Analysis and Algorithms

We study the computational properties of the different notions of explanation for DL-Lite$_{\mathcal{R}}$ KBs. In addition to the problem of generating a single explanation (genOne), or a single best explanation (genBest) according to a given criteria, we consider four related decision problems: decide whether a given assertion appears in some explanation (rel) or in every explanation (nec), decide whether a candidate is an explanation (rec), resp. a best explanation according to a given criterion (best rec).

The remainder of this section will be devoted to proving the following theorem and introducing the procedures that are implemented in the system described in the next section.

Theorem 5.6. *The complexity results displayed in Table 1 hold.*

When showing that a decision problem is hard for a given complexity class, we use standard polynomial-time many-one reductions (also known as Karp reductions), which transform an instance of one decision problem into an instance of a second decision problem.

We consider that a procedure solves the generation task genOne (resp. genBest) if it outputs an explanation (resp. best explanation according to the chosen criterion) when

Type of Explanation	Propositional Equivalent	
Positive AR-answers	Minimal unsatisfiable subsets of $\varphi_{\neg q}$ w.r.t. φ_{cons}	Prop. 5.9
Negative AR-answers	Minimal models of $\varphi_{\neg q} \wedge \varphi_{cons}$	Prop. 5.14
Negative IAR-answers	Minimal models of $\varphi_{\neg q}$	Prop. 5.19

Table 2: Connections between explanations and propositional satisfiability.

there is at least one explanation, and otherwise, it outputs no. To show that a generation task is hard for a class $\mathcal{C}$, we reduce a $\mathcal{C}$-hard decision problem to it. As we cannot use many-one reductions (which relate two decision problems), we will use polynomial-time Turing reductions, that is, we will show how to solve the $\mathcal{C}$-hard decision problem using a polynomial-time Turing machine that can use the generation task as an oracle. Moreover, to prove stronger intractability results, we will only allow a single oracle call.

Many of our upper bounds (and some lower bounds too) are obtained by establishing tight connections between explanations for query (non-)answers and different notions related to propositional satisfiability. These connections are summarized in Table 2.

5.2.1 Positive Brave and IAR-Answers

By Propositions 4.3 and 4.8, the conflicts for a DL-Lite$_\mathcal{R}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and the causes of a query q can be computed in polynomial time w.r.t. data complexity. Since the explanations for positive brave-answers are the causes, and the explanations for positive IAR-answers are the causes that belong to every repair, i.e. those which do not contain any assertion involved in a conflict of $\mathcal{K}$, it is possible to compute the entire set of explanations for positive brave and IAR-answers in polynomial time. This means that genOne is in PTime. For polynomial-time ranking criteria, both genBest and best rec are solvable in polynomial time since we can compare all of the explanations to identify the best ones. The sets of relevant and necessary assertions can be computed in polynomial time by taking the union and intersection of the explanations.

In the appendix, we show that the decision problems rec, rel, and nec are not just in PTime but in fact belong to the much lower AC^0 complexity class. The proof involves defining first-order queries that can be used to decide these problems.

Proposition 5.7. *Regarding explanations for positive brave- and IAR-answers,* rec, rel, *and* nec *are in* AC^0 *w.r.t. data complexity.*

The preceding result is primarily of theoretical interest, as computing the sets of relevant and necessary assertions via rewriting would require us to construct and evaluate a huge number of rather complex first-order queries (one per ABox assertion). For this reason, in our implementation, we adopt the polynomial-time procedures sketched above, as they allow us to easily obtain the relevant and necessary assertions from the causes and conflicts (which have already been computed as part of the preprocessing and querying phases).

5.2.2 Positive AR-Answers

We relate explanations of positive AR-answers to minimal unsatisfiable subsets of a set of propositional clauses.

Definition 5.8 (Minimal Unsatisfiable Subset)**.** Given sets F and H of soft and hard clauses respectively, a subset $M \subseteq F$ is a *minimal unsatisfiable subset* (MUS) of F w.r.t. H if (i) $M \cup H$ is unsatisfiable, and (ii) $M' \cup H$ is satisfiable for every $M' \subsetneq M$.

To explain $\mathcal{K} \models_{\text{AR}} q(\vec{a})$, we exploit the encoding from Figure 2. For the set F of soft clauses, we take the clauses

$$\varphi_{\neg q} = \{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathsf{causes}(q(\vec{a}), \mathcal{K})\} \text{ with } \lambda_{\mathcal{C}} = \bigvee_{\beta \in \mathsf{confl}(\mathcal{C}, \mathcal{K})} x_\beta$$

that aim to contradict every cause, and for the set H of hard clauses, we use the clauses

$$\varphi_{cons} = \{\neg x_\alpha \vee \neg x_\beta \mid x_\alpha, x_\beta \in \mathsf{vars}(\varphi_{\neg q}), \beta \in \mathsf{confl}(\{\alpha\}, \mathcal{K})\}$$

that enforce consistency.

Proposition 5.9. *A set $\mathcal{E} \subseteq \mathsf{causes}(q(\vec{a}), \mathcal{K})$ is an explanation for $\mathcal{K} \models_{AR} q(\vec{a})$ iff the set of clauses $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}\}$ is a MUS of $\varphi_{\neg q}$ w.r.t. φ_{cons}.*

Proof. First suppose that $\mathcal{E}$ is an explanation of $\mathcal{K} \models_{\text{AR}} q(\vec{a})$. By Definition 5.3, this means that every repair of $\mathcal{K}$ contains at least one cause $\mathcal{C}$ from $\mathcal{E}$. It follows that it is not possible to select one conflicting assertion for each cause in $\mathcal{E}$ in a consistent way, i.e. $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}\} \cup \varphi_{cons}$ is inconsistent. Moreover, the minimality condition ensures that for every proper subset $\mathcal{E}' \subsetneq \mathcal{E}$, there is a repair $\mathcal{R}$ that does not contain any cause from $\mathcal{E}'$. We can use $\mathcal{R}$ to select a consistent set of assertions that conflict with every cause in $\mathcal{E}'$, i.e. $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}'\} \cup \varphi_{cons}$ is satisfiable. Thus, $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}\}$ is a MUS of $\varphi_{\neg q}$ w.r.t. φ_{cons}.

Conversely, suppose $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}\}$ is a MUS of $\varphi_{\neg q}$ w.r.t. φ_{cons}. Then $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}\} \cup \varphi_{cons}$ is unsatisfiable, and every $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}'\} \cup \varphi_{cons}$ with $\mathcal{E}' \subsetneq \mathcal{E}$ is satisfiable. The fact that $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}\} \cup \varphi_{cons}$ is unsatisfiable means that every repair must contain one of the causes in $\mathcal{E}$. The satisfiability of $\{\lambda_{\mathcal{C}} \mid \mathcal{C} \in \mathcal{E}'\} \cup \varphi_{cons}$ for $\mathcal{E}' \subsetneq \mathcal{E}$ implies the existence of a repair that omits every cause in $\mathcal{E}'$. We have thus shown that $\mathcal{E}$ satisfies the conditions of Definition 5.3, so it is an explanation of $\mathcal{K} \models_{\text{AR}} q(\vec{a})$. □

Proposition 5.9 directly yields a method of computing explanations for positive AR-answers: simply construct the encoding and call an algorithm for generating MUSes (as implemented e.g. by some SAT solvers). Moreover, by combining Proposition 5.9 and known complexity results for MUSes, we obtain the following upper bounds:

Proposition 5.10. *Regarding explanations for positive AR-answers,* rec *is in* BH_2*,* best rec *is in* Π_2^p*, and* rel *is in* Σ_2^p *w.r.t. data complexity.*

Proof. We recall the following complexity results for MUSes (see Liberatore, 2005):

- Deciding if a set of clauses is a MUS is BH_2-complete.
- Deciding if a clause belongs to some MUS is Σ_2^p-complete.

When combined with Proposition 5.9, the first item yields membership in BH_2 of REC. For BEST REC, we show that an explanation is *not* a best one by guessing a better candidate and checking in BH_2 that it is an explanation. This yields a Σ_2^p procedure for the complement of BEST REC, hence membership in Π_2^p for BEST REC.

For REL, we note that an assertion α is relevant for explaining $\mathcal{K} \models_{\mathrm{AR}} q(\vec{a})$ just in the case that there exists a cause $\mathcal{C}$ for $q(\vec{a})$ w.r.t. $\mathcal{K}$ that contains α and appears in some explanation. By Proposition 5.9, the latter holds iff $\lambda_{\mathcal{C}}$ belongs to some MUS of $\varphi_{\neg q}$ w.r.t. φ_{cons}. By the second item above, deciding whether a particular clause $\lambda_{\mathcal{C}}$ belongs to some MUS can be decided in Σ_2^p. To obtain a Σ_2^p decision procedure for REL, we simply add an initial non-deterministic guess of a cause $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \mathcal{K})$ that contains α. □

We next show the NP upper bound for NEC.

Proposition 5.11. *Regarding explanations for positive AR-answers,* NEC *is in* NP *w.r.t. data complexity.*

Proof. We claim that α belongs to every explanation of $\mathcal{K} \models_{\mathrm{AR}} q(\vec{a})$ iff there exists a repair $\mathcal{R}$ such that every cause for $q(\vec{a})$ included in $\mathcal{R}$ contains α. Indeed, if every repair $\mathcal{R}$ contains a cause $\mathcal{C}_{\mathcal{R}}$ with $\alpha \notin \mathcal{C}_{\mathcal{R}}$, then there would be a minimal disjunction of these $\mathcal{C}_{\mathcal{R}}$ which covers every repair and omits α. Conversely, if there exists such a repair $\mathcal{R}$, then any minimal disjunction of causes that covers every repair must contain α.

It follows that α belongs to every explanation of $\mathcal{K} \models_{\mathrm{AR}} q(\vec{a})$ just in the case that either there are no explanations at all (i.e. $\mathcal{K} \not\models_{\mathrm{AR}} q(\vec{a})$) or there exists a repair $\mathcal{R}$ of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ such that $\langle \mathcal{T}, \mathcal{R} \setminus \{\alpha\} \rangle \not\models q(\vec{a})$. Both conditions can be tested in NP w.r.t. data complexity. Indeed, to decide whether the second condition holds, we simply guess a subset $\mathcal{R} \subseteq \mathcal{A}$ and check (in PTIME w.r.t. data complexity) that $\mathcal{R}$ is a repair and $\langle \mathcal{T}, \mathcal{R} \setminus \{\alpha\} \rangle \not\models q(\vec{a})$. □

The following proposition shows how the connection to MUSes can be exploited to obtain matching lower bounds.

Proposition 5.12. *Regarding explanations for positive AR-answers,* REC *is* BH_2*-hard,* NEC *is* NP*-hard,* REL *is* Σ_2^p*-hard, and* GENONE *is* NP*-hard w.r.t. data complexity. Moreover, if we rank explanations according to the number of causes or number of assertions, then* BEST REC *(resp.* GENBEST*) is* Π_2^p*-hard (resp.* Σ_2^p*-hard) w.r.t. data complexity.*

Proof. We show how the MUSes of a propositional clause set can be captured by explanations of positive AR-answers.

Let $\varphi_0 = \{C_1, ..., C_k\}$ be a set of propositional clauses over $\{X_1, ..., X_n\}$, and consider the KB and query used in the reduction of the proof of coNP-hardness of AR entailment presented by Bienvenu (2012):

$$\mathcal{T}_0 = \{\exists P^- \sqsubseteq \neg\exists N^-, \exists P \sqsubseteq \neg\exists U^-, \exists N \sqsubseteq \neg\exists U^-, \exists U \sqsubseteq A\}$$
$$\mathcal{A}_0 = \{P(c_j, x_i) \mid X_i \in C_j\} \cup \{N(c_j, x_i) \mid \neg X_i \in C_j\} \cup \{U(a, c_j) \mid 1 \leq j \leq k\}$$
$$q_0(x) = A(x)$$

The causes for $q_0(a)$ are given by the assertions $U(a, c_j)$, which are in conflict with assertions of the form $P(c_j, x_i)$ or $N(c_j, x_i)$. It was shown that $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle \models_{\mathrm{AR}} A(a)$ iff φ_0 is unsatisfiable. To prove the proposition, we will require the following stronger claim:

Claim. The following are equivalent:

1. the set of clauses $\{C_{j_1}, ..., C_{j_p}\}$ is unsatisfiable
2. every repair of $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle$ contains some assertion from $\{U(a, c_{j_1}), ..., U(a, c_{j_p})\}$

Proof of claim. It will be more convenient to show that the negations of the two statements are equivalent. First suppose that $\{C_{j_1}, ..., C_{j_p}\}$ is satisfiable, as witnessed by the satisfying assignment ν. Define a repair $\mathcal{R}_\nu$ of $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle$ by including the assertion $P(c_j, v_i)$ if $\nu(v_i) = \mathsf{true}$, including $N(c_j, v_i)$ if $\nu(v_i) = \mathsf{false}$, and then adding as many other assertions as needed to obtain a maximal $\mathcal{T}_0$-consistent subset. Since ν satisfies every clause in $\{C_{j_1}, ..., C_{j_p}\}$, it follows that for every index $\ell \in \{j_1, \ldots, j_p\}$, the clause C_ℓ contains either a positive literal v_ℓ such that $\nu(v_\ell) = \mathsf{true}$ or a negative literal $\neg v_\ell$ such that $\nu(v_\ell) = \mathsf{false}$. In the former case, $\mathcal{R}_\nu$ contains the assertion $P(c_\ell, v_\ell)$, and in the latter case, $\mathcal{R}_\nu$ contains $N(c_\ell, v_\ell)$. In both cases, there is an assertion in $\mathcal{R}_\nu$ that conflicts with $U(a, c_\ell)$, so the latter assertion cannot appear in $\mathcal{R}_\nu$. We have thus shown that $\mathcal{R}_\nu$ does not contain any of the assertions in $\{U(a, c_{j_1}), ..., U(a, c_{j_p})\}$.

Next suppose that there exists a repair $\mathcal{R}$ of $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle$ that has an empty intersection with the set $\{U(a, c_{j_1}), ..., U(a, c_{j_p})\}$. By the maximality of $\mathcal{R}$, it follows that for every $\ell \in \{j_1, \ldots, j_p\}$, there must exist an assertion in $\mathcal{R}$ of the form $P(c_\ell, v_i)$ or $N(c_\ell, v_i)$. Define a (possibly partial) assignment $\nu_\mathcal{R}$ by setting X_i to true if $\mathcal{R}$ contains some $P(c_j, x_i)$ and to false if $\mathcal{R}$ contains some $N(c_j, x_i)$ (recall that $\mathcal{R}$ is consistent with $\mathcal{T}_0$, and so it cannot contain both $P(c_j, x_i)$ and $N(c_{j'}, x_i)$). By construction, $\nu_\mathcal{R}$ satisfies all of the clauses in $\{C_{j_1}, ..., C_{j_p}\}$, i.e. $\{C_{j_1}, ..., C_{j_p}\}$ is satisfiable. (*end proof of claim*)

It follows from the preceding claim that the explanations for $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle \models_{\text{AR}} q_0(a)$, i.e. the minimal sets of causes for $q_0(a)$ that cover all repairs, correspond precisely to the MUSes of φ_0. We can therefore exploit known complexity results for MUSes (Liberatore, 2005):

- Deciding if a clause belongs to a MUS is Σ_2^p-complete, so deciding if $U(a, c_j)$ belongs to an explanation for $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle \models_{\text{AR}} q_0(a)$ is Σ_2^p-hard w.r.t. data complexity. Thus, we have a Σ_2^p lower bound for REL.
- Deciding if a clause belongs to every MUS is NP-complete, so deciding if $U(a, c_j)$ belongs to every explanation for $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle \models_{\text{AR}} q_0(a)$ is NP-hard w.r.t. data complexity. This gives an NP lower bound for NEC.
- REC: Deciding if a set of clauses is a MUS is BH_2-complete, so deciding if $\{\{U(a, c_{j_1})\}, ..., \{U(a, c_{j_p})\}\}$ is an explanation is BH_2-hard w.r.t. data complexity. Hence, REC is BH_2-hard.

The proof by Liberatore (2005) for Σ_2^p-hardness of deciding if there exists a MUS of size at most k also shows that deciding if a set of clauses is a smallest MUS is Π_2^p-hard. It follows that deciding if an explanation for an AR-answer contains a smallest number of causes is Π_2^p-hard. Moreover, since every cause in the considered KB consists of a single assertion, deciding if an explanation for an AR-answer contains a smallest number of assertions is also Π_2^p-hard.

To see why the generation task GENONE is NP-hard, we note that to solve the NP-complete problem of deciding whether $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$, it suffices to call the procedure for GENONE to generate a single explanation for $\mathcal{K} \models_{\text{AR}} q(\vec{a})$. If the procedure outputs 'no' (meaning there is no explanation for $\mathcal{K} \models_{\text{AR}} q(\vec{a})$), then we output 'yes', and if it outputs an explanation, then we return 'no'.

The Σ_2^p-hardness of genBest, when explanations are ranked based upon the number of disjuncts or the number of assertions, follows from the Π_2^p-hardness of best rec for these same criteria. Indeed, to show that an explanation is *not* a best explanation, it suffices to generate a best explanation (genBest) and verify that it has fewer disjuncts / assertions than the explanation at hand. □

5.2.3 Negative AR-Answers

We relate explanations of negative AR-answers to minimal models of $\varphi_{\neg q} \cup \varphi_{cons}$.

Definition 5.13 (Minimal model)**.** Given a clause set ψ over variables X, a set $M \subseteq X$ is a *minimal model* of ψ iff (i) every valuation that assigns true to all variables in M satisfies ψ, (ii) no $M' \subsetneq M$ satisfies this condition. Cardinality-minimal models are defined analogously.

Proposition 5.14. *A set $\mathcal{E}$ is an explanation (resp. cardinality-minimal explanation) for $\mathcal{K} \not\models_{AR} q(\vec{a})$ iff $\{x_\alpha \mid \alpha \in \mathcal{E}\}$ is a minimal (resp. cardinality-minimal) model of $\varphi_{\neg q} \cup \varphi_{cons}$.*

Proof. We recall that the reason why $\mathcal{K} \models_{\text{AR}} q(\vec{a})$ iff $\varphi_{\neg q} \wedge \varphi_{cons}$ is unsatisfiable is because the assertions whose corresponding variables are assigned to true in a valuation that satisfies $\varphi_{\neg q} \wedge \varphi_{cons}$ form a subset of the ABox which: (i) contradicts every cause, since $\varphi_{\neg q}$ states that for every cause, one conflicting assertion is selected, and (ii) is consistent, since φ_{cons} states that two assertions in a conflict cannot be selected together. Thus, the inclusion-minimal models of $\varphi_{\neg q} \wedge \varphi_{cons}$ are precisely the explanations for negative AR-answers. □

Next we show the complexity upper bounds for the decision problems.

Proposition 5.15. *Regarding explanations for negative AR-answers,* rec *is in* AC^0*,* best rec *is in co*NP*,* rel *is in* NP*, and* nec *is in co*NP *w.r.t. data complexity.*

Proof. It follows from Definition 5.4 that deciding whether $\mathcal{E} \subseteq \mathcal{A}$ is an explanation for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ can be done in polynomial time (data complexity) by checking:

- consistency of $\langle \mathcal{T}, \mathcal{E} \rangle$
- inconsistency of $\langle \mathcal{T}, \mathcal{E} \cup \mathcal{C} \rangle$ for every $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \mathcal{K})$
- minimality of $\mathcal{E}$: no proper subset $\mathcal{E}' \subsetneq \mathcal{E}$ satisfies the two previous conditions.

In the appendix, we prove by means of first-order query rewriting an improved AC^0 upper bound for rec.

We can decide in NP that an explanation $\mathcal{E}$ is *not* a best explanation (according to some polynomial-time ranking criterion) by guessing a subset $\mathcal{E}' \subseteq \mathcal{A}$ and verifying in polynomial time w.r.t. data complexity that $\mathcal{E}'$ is an explanation (see previous paragraph) and is better than $\mathcal{E}$ according to the given criterion. This yields a coNP upper bound for best rec.

A simple NP procedure for deciding rel (resp. *not* nec) consists in guessing a subset $\mathcal{E} \subseteq \mathcal{A}$ that contains (does not contain) the considered assertion and checking in polynomial time whether it is an explanation (using the polynomial-time procedure for rec). □

For the purposes of implementation, we propose alternative SAT-based procedures for rel and nec, based upon the following two propositions.

Proposition 5.16. *An assertion α is relevant for explaining $\mathcal{K} \not\models_{AR} q(\vec{a})$ iff the clause set $\varphi_{\neg q} \cup \varphi_{cons} \cup \varphi_\alpha$ is satisfiable, where*

$$\varphi_\alpha = \{ \bigvee_{\mathcal{C} \in \mathsf{causes}(q(\vec{a}),\mathcal{K}), \alpha \in \mathsf{confl}(\mathcal{C},\mathcal{K})} x_\mathcal{C} \} \cup$$
$$\{\neg x_\mathcal{C} \vee \neg x_\beta \mid \mathcal{C} \in \mathsf{causes}(q(\vec{a}),\mathcal{K}), \alpha \in \mathsf{confl}(\mathcal{C},\mathcal{K}), \beta \in \mathsf{confl}(\mathcal{C},\mathcal{K}), \beta \neq \alpha\}$$

Proof. If α is relevant, then there exists an explanation $\mathcal{E}$ such that $\alpha \in \mathcal{E}$. Since $\mathcal{E}$ is minimal, there exists a cause $\mathcal{C}$ such that $\mathcal{C} \cup (\mathcal{E} \backslash \{\alpha\})$ is consistent. It follows that no assertion $\beta \in \mathsf{confl}(\mathcal{C},\mathcal{K})$ belongs to $\mathcal{E}$ except for α. Then the valuation ν such that $\nu(x_\mathcal{C}) = \mathsf{true}$, and for every assertion β, $\nu(x_\beta) = \mathsf{true}$ if $\beta \in \mathcal{E}$, $\nu(x_\beta) = \mathsf{false}$ otherwise, satisfies $\varphi_{\neg q} \cup \varphi_{cons} \cup \varphi_\alpha$.

Conversely, if $\varphi_{\neg q} \cup \varphi_{cons} \cup \varphi_\alpha$ is satisfiable, then it is possible to contradict every cause with a consistent set $\mathcal{E}$ of assertions such that there exists a cause $\mathcal{C}$ such that the only assertion of $\mathcal{E} \cap \mathsf{confl}(\mathcal{C},\mathcal{K})$ is α. Then an explanation that contains α is included in $\mathcal{E}$. □

Proposition 5.17. *An assertion α is necessary for explaining $\mathcal{K} \not\models_{AR} q(\vec{a})$ iff the set of clauses $\varphi_{\neg q} \cup \varphi_{cons} \cup \{\neg x_\alpha\}$ is unsatisfiable.*

Proof. By Proposition 5.14, $\mathcal{E}$ is an explanation for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ iff $\{x_\alpha \mid \alpha \in \mathcal{E}\}$ is a minimal model of $\varphi_{\neg q} \cup \varphi_{cons}$. It follows that α belongs to every explanation for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ just in the case that x_α belongs to every minimal model of $\varphi_{\neg q} \cup \varphi_{cons}$, in which case $\varphi_{\neg q} \cup \varphi_{cons} \cup \{\neg x_\alpha\}$ is unsatisfiable. □

The next proposition establishes matching lower bounds.

Proposition 5.18. *Regarding explanations for negative AR-answers,* NEC *is co*NP*-hard, and* REL, GENONE, *and* GENBEST *(for any ranking criterion) are* NP*-hard w.r.t. data complexity. If explanations are ranked by cardinality, then* BEST REC *is co*NP*-hard w.r.t. data complexity.*

Proof. All reductions are from propositional (un)satisfiability. Let $\varphi_0 = C_1 \wedge ... \wedge C_k$ be a set of clauses over propositional variables $\{X_1, ..., X_n\}$.

• GENONE and GENBEST: Let $\mathcal{T}_0$, $\mathcal{A}_0$, and q_0 be as in Proposition 5.12. We know that φ_0 is satisfiable iff $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle \not\models_{\text{AR}} A(a)$. Thus, to decide the satisfiability of φ_0, we generate a (best) explanation of $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle \not\models_{\text{AR}} A(a)$. If an explanation is produced, then we return 'yes', and if the procedure returns with no explanation, then we output 'no'.

• NEC: We again let $\mathcal{T}_0$, $\mathcal{A}_0$, and q_0 be as in Proposition 5.12. Define a new TBox $\mathcal{T}_1 = \mathcal{T}_0 \cup \{\exists U \sqsubseteq \neg S\}$ and ABox $\mathcal{A}_1 = \mathcal{A}_0 \cup \{S(a)\}$. By construction, the assertion $S(a)$ contradicts every cause for $q_0(a)$, so $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle \not\models_{\text{AR}} q_0(a)$. We show that deciding whether φ_0 is satisfiable is equivalent to deciding if $S(a)$ is *not* necessary for explaining $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle \not\models_{\text{AR}} q_0(a)$. This establishes the coNP-hardness of checking necessity.

($\Rightarrow$) Let ν be a satisfying valuation for φ_0. It can be easily verified that the set $\{P(c_j, v_i) \in \mathcal{A}_0 \mid \nu(v_i) = \mathsf{true}\} \cup \{N(c_j, v_i) \in \mathcal{A}_0 \mid \nu(v_i) = \mathsf{false}\}$ conflicts with every cause of $q_0(a)$, and so by choosing a subset of these assertions, we can construct an explanation for $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle \not\models_{\text{AR}} q_0(a)$ that does not contain $S(a)$.

($\Leftarrow$) An explanation $\mathcal{E}$ that does not contain $S(a)$ forms a $\mathcal{T}_1$-consistent set of P- and N-assertions such that every c_j has an outgoing P- or N-edge. We obtain a (partial) assignment $\nu_{\mathcal{E}}$ that satisfies φ_0 by setting $\nu_{\mathcal{E}}(v_i) = \mathsf{true}$ if $\mathcal{E}$ contains an assertion $P(c_j, v_i)$ and $\nu_{\mathcal{E}}(v_i) = \mathsf{false}$ if $\mathcal{E}$ contains an assertion $N(c_j, v_i)$.

• REL: We use the TBox $\mathcal{T}_1$ and the ABox $\mathcal{A}_2 = \mathcal{A}_1 \cup \{U(a, c_{k+1}), P(c_{k+1}, x_{n+1})\}$. Again, we note that $S(a)$ contradicts every cause for $q_0(a)$, so $\langle \mathcal{T}_1, \mathcal{A}_2 \rangle \not\models_{\text{AR}} \ q_0(a)$. We show that φ_0 is satisfiable iff $P(c_{k+1}, x_{n+1})$ is relevant for explaining $\langle \mathcal{T}_1, \mathcal{A}_2 \rangle \not\models_{\text{AR}} \ q_0(a)$; it follows that relevance is NP-hard.

($\Rightarrow$) If φ_0 is satisfiable, then we can obtain an explanation for $\langle \mathcal{T}_1, \mathcal{A}_2 \rangle \not\models_{\text{AR}} \ q_0(a)$ by adding $P(c_{k+1}, x_{n+1})$ to a minimal subset of the P- and N-assertions corresponding to a satisfying truth assignment for φ_0.

($\Leftarrow$) If φ_0 is unsatisfiable, then every explanation must contain $S(a)$. It follows that $\{S(a)\}$ is the only explanation, so $P(c_{k+1}, x_{n+1})$ is not relevant.

• BEST REC: We consider the following KB:

$$\begin{aligned}\mathcal{T}_3 &= \mathcal{T}_0 \cup \{U_1 \sqsubseteq U, U_2 \sqsubseteq U, \exists U_1^- \sqsubseteq \neg T, \exists U_2 \sqsubseteq \neg S\} \\ \mathcal{A}_3 &= \{P(c_j, x_i) \mid X_i \in C_j\} \cup \{N(c_j, x_i) \mid \neg X_i \in C_j\} \cup \\ &\quad \{U_1(a, c_j), U_2(a, c_j), T(c_j) \mid 1 \leq j \leq k\} \cup \{S(a)\}\end{aligned}$$

We claim that $\mathcal{E} = \{S(a), T(c_1), ..., T(c_k)\}$ is a smallest explanation for $\langle \mathcal{T}_3, \mathcal{A}_3 \rangle \not\models_{\text{AR}} \ q_0(a)$ iff φ_0 is unsatisfiable.

($\Rightarrow$) If φ_0 is satisfiable, then we can use a satisfying truth assignment to define a consistent set of k P- and N-edges such that every c_j has an outgoing edge. This set is an explanation for $\langle \mathcal{T}_3, \mathcal{A}_3 \rangle \not\models_{\text{AR}} \ q_0(a)$, and it has fewer assertions than $\mathcal{E}$.

($\Leftarrow$) If there exists an explanation of size at most k, it contains necessarily only P- and N-edges, since k assertions (P, N or T) are needed to conflict all U_1, and $S(a)$ is needed as soon as one of the U_1-assertions is conflicted only by a T-assertion. It follows that there exists a consistent set of P- and N-assertions such that every c_j has an outgoing edge, from which we can construct a satisfying assignment for φ_0.

Note that role inclusions are not needed for the lower bound, as we can replace the inclusions $U_1 \sqsubseteq U$ and $U_2 \sqsubseteq U$ in the reduction by the following set of concept inclusions: $\exists P \sqsubseteq \neg\exists U_1^-, \exists N \sqsubseteq \neg\exists U_1^-, \exists P \sqsubseteq \neg\exists U_2^-, \exists N \sqsubseteq \neg\exists U_2^-, \exists U_1 \sqsubseteq A, \exists U_2 \sqsubseteq A$. □

5.2.4 Negative IAR-Answers

Similarly to the case of negative AR-answers, we relate explanations of negative IAR-answers to minimal models of the clause set $\varphi_{\neg q}$. Indeed, to show that an answer is not IAR, it is sufficient to contradict every cause, without the consistency constraint.

Proposition 5.19. *A set $\mathcal{E}$ is an explanation (resp. cardinality-minimal explanation) for $\mathcal{K} \not\models_{IAR} q(\vec{a})$ iff $\{x_\alpha \mid \alpha \in \mathcal{E}\}$ is a minimal (resp. cardinality-minimal) model of $\varphi_{\neg q}$.*

Proof. The assertions whose corresponding variables are assigned to true in a valuation that satisfies $\varphi_{\neg q}$ form a subset of the ABox which contradicts every cause, since $\varphi_{\neg q}$ states that for every cause, one conflicting assertion is selected. Thus, the inclusion-minimal (resp. cardinality-minimal) models of $\varphi_{\neg q}$ are precisely the explanations (resp. cardinality-minimal explanations) for negative IAR-answers. □

Importantly, $\varphi_{\neg q}$ does not contain any negative literals, and it is a folklore result that for *positive* clause sets, a single minimal model can be computed in polynomial time: one starts with the model in which all variables are set to true, and then one proceeds to set variables to false so long as the clauses are satisfied. By combining this tractability result with Proposition 5.19, we obtain the following:

Proposition 5.20. *Regarding negative IAR-answers,* GenOne *is in* PTime.

Deciding the necessity problem for monotone CNF is also straightforward due to the following property: v appears in every minimal model of a monotone CNF φ if and only if φ contains the clause v. For the relevance problem, we can use another easy-to-check property (proven by Boros, Elbassioni, Gurvich, and Khachiyan (2000) for the closely related setting of hitting set computation): a variable v is true in some inclusion-minimal model of φ iff it appears in a clause of φ that is not subsumed by any other clause. The following lemmas result from using Proposition 5.19 to transfer these properties to our setting:

Lemma 5.21. *An assertion is necessary for explaining $\mathcal{K} \not\models_{IAR} q(\vec{a})$ just in the case that it is the only conflict of some cause for $q(\vec{a})$.*

Lemma 5.22. *An assertion is relevant for explaining $\mathcal{K} \not\models_{IAR} q(\vec{a})$ just in the case that it is in conflict with a cause $\mathcal{C}$ for $q(\vec{a})$ such that for every other cause $\mathcal{C}'$, if $\mathsf{confl}(\mathcal{C}', \mathcal{K}) \subseteq \mathsf{confl}(\mathcal{C}, \mathcal{K})$, then $\alpha \in \mathsf{confl}(\mathcal{C}', \mathcal{K})$.*

Based upon the preceding lemmas, we introduce a procedure RelNecNegIAR (Algorithm 4) for computing the sets of relevant and necessary assertions for explaining a negative IAR-answer. The procedure starts by computing the causes of $q(\vec{a})$ (line 1), and then iterates over the causes (lines 2-5). When examining cause $\mathcal{C}_i$, it first tests (line 4) whether $|\mathsf{confl}(\mathcal{C}_i, \mathcal{K})| = 1$, and if this is the case, the unique assertion in $\mathsf{confl}(\mathcal{C}_i, \mathcal{K})$ is added to $Necessary$ (as sanctioned by Lemma 5.21). Next, the set $Relevant_i$ is initialized to $Confl_i$ (line 6), and then we iterate (lines 7-10) over all other causes $\mathcal{C}_j$ to filter the assertions in $Relevant_i$ and retain only those that satisfy the conditions from Lemma 5.22. Finally, we let $Relevant$ be the union over the sets $Relevant_i$, and we output the sets $Relevant$ and $Necessary$. Correctness of the procedure follows from Lemmas 5.21 and 5.22.

As causes and conflicts can be computed in polynomial time for data complexity (Propositions 4.3 and 4.8), it is easy to show that RelNecNegIAR runs in polynomial time in $|\mathcal{A}|$. In the appendix, we prove the following improved AC^0 upper bound for rec, nec, and rel by showing how these problems can be solved by means of first-order query rewriting. Again, the AC^0 upper bound is mostly of theoretical interest, as for our implementation, we will prefer to use the polynomial-time RelNecNegIAR procedure rather than FO-rewritings.

Proposition 5.23. *Regarding explanations for negative IAR-answers,* rec, nec, *and* rel *are in* AC^0 *w.r.t. data complexity.*

We next establish the remaining complexity upper bound.

Proposition 5.24. *Regarding explanations for negative IAR-answers,* best rec *is in* coNP *w.r.t. data complexity.*

ALGORITHM 4: RelNecNegIAR

Input: a conjunctive query $q(\vec{x})$, a candidate answer $\vec{a}$, a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: relevant and necessary assertions for explaining $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$

```
Causes ← ComputeCauses(q(a⃗), K);
foreach C_i ∈ Causes do
    Confl_i ← ConflictsFor(C_i, K);
    if |Confl_i| = 1 then
        Necessary ← Necessary ∪ Confl_i;
    Relevant_i ← Confl_i;
    foreach C_j ∈ Causes do
        Confl_j ← ConflictsFor(C_j, K);
        if Confl_j \ Confl_i = ∅ then
            Relevant_i ← Relevant_i ∩ Confl_j;
Relevant ← ⋃ Relevant_i;
Output Relevant, Necessary;
```

Proof. We can decide in NP that an explanation $\mathcal{E}$ is *not* a best explanation (according to some polynomial-time ranking criterion) by guessing a subset $\mathcal{E}' \subseteq \mathcal{A}$ and verifying in polynomial time w.r.t. data complexity that $\mathcal{E}'$ is an explanation and that it is better than $\mathcal{E}$ according to the given criterion. This yields a coNP upper bound for BEST REC. □

Finally, we establish the intractability of BEST REC and GENBEST when explanations are ranked by cardinality.

Proposition 5.25. *Regarding explanations for negative IAR-answers in the case where explanations are ranked by cardinality,* GENBEST *is* NP*-hard, and* BEST REC *is co*NP*-hard w.r.t. data complexity.*

Proof. We give a reduction from the problem of deciding if a truth assignment that satisfies a monotone 2-SAT formula assigns a smallest number of variables to true. This problem is coNP-complete (coNP-hardness can be shown by a straightforward reduction from the complement of the well-known NP-complete vertex cover problem).

Let $\varphi = C_1 \wedge ... \wedge C_k$ be a monotone 2-CNF over the variables $\{X_1, ..., X_n\}$, and let ν be a truth assignment that satisfies φ. Consider the following KB:

$$\mathcal{T} = \{\exists P_r^- \sqsubseteq \neg T \mid 1 \leq r \leq 2\}$$
$$\mathcal{A} = \{T(x_i) \mid 1 \leq i \leq n\} \cup \{P_r(c_j, x_i) \mid X_i \text{ } r^{th} \text{ term of } C_j\}$$
$$q = \exists y z_1 z_2 \, P_1(y, z_1) \wedge P_2(y, z_2)$$

The causes for q take the form $\{P_1(c_j, x_{i_1}), P_2(c_j, x_{i_2})\}$. It follows that an explanation for $\langle \mathcal{T}, \mathcal{A} \rangle \not\models_{\text{IAR}} q$ is a set $\mathcal{E}$ of T-assertions such that for every c_j, there is at least one $X_i \in C_j$ such that $T(x_i) \in \mathcal{E}$. Thus, deciding if ν assigns a minimal number of variables to true is equivalent to deciding if $\mathcal{E} = \{T(x_i) \mid \nu(X_i) = \mathsf{true}\}$ is a smallest explanation. This yields the coNP-hardness of BEST REC, as well as the NP- and coNP-hardness of GENBEST: we can solve the minimum assignment problem - and its complement - by generating a cardinality-minimal explanation and comparing its size with the number of variables set to true by the candidate assignment. □

6. Implementation and Benchmark

In this section, we present the CQAPri system, which implements the query answering and explanation algorithms described in Sections 4 and 5, as well as the benchmark we created to test the system; the results of our experimental evaluation will be presented in the following section. To improve readability, full-page figures and results tables for Sections 6 and 7 are given in the appendix.

6.1 The CQAPri System

We implemented our query answering and explaining framework under AR, IAR and brave semantics over DL-Lite$_\mathcal{R}$ KBs in Java v1.7 within our CQAPri ("Consistent Query Answering with Priorities") tool[8]. CQAPri is built on top of the relational database server PostgreSQL v9.3.2 (www.postgresql.org), the Rapid v1.0 query rewriting engine for DL-Lite (Chortaras, Trivela, & Stamou, 2011), and the SAT4J v2.3.4 SAT solver (Berre & Parrain, 2010). All these building blocks are used with their default settings.

Following the framework developed in Section 4, CQAPri utilizes the brave, AR, and IAR together to classify a query answer $\vec{a}$ into one of 3 classes:

- **Possible**: $\mathcal{K} \models_{\text{brave}} q(\vec{a})$ and $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$
- **Likely**: $\mathcal{K} \models_{\text{AR}} q(\vec{a})$ and $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$
- **(Almost) sure**: $\mathcal{K} \models_{\text{IAR}} q(\vec{a})$

CQAPri handles ABoxes stored in PostgreSQL, while it keeps the TBox in-memory. ABoxes are stored as relational databases with one table per concept and role, of one or two columns (named s for concept and s and o for role) that contains the individuals involved in that concept or role. B-Tree indexes have been created for each table, on s for concepts and (s, o) and (o, s) for roles. The concept, role, and individual names are encoded by integers, and a dictionary table relates each name to its identifier.

CQAPri computes the set of conflicts for the KB in a preprocessing phase since it is query-independent. Conflicts are computed following the steps in Algorithm 1 and stored as a conflict graph (Definition 4.4). More precisely, the SQLized rewritings of the queries looking for counter-examples to the negative TBox inclusions are evaluated over the ABox, their images are retrieved and stored as a graph whose vertices are assertions and edges indicate images, and finally the non-minimal ones are discarded by removing edges between any self-inconsistent assertion and the others to obtain the conflict graph.

At query time, the system implements the steps from the (non-Boolean version of) ClassifyQuery procedure. When a query arrives, CQAPri first evaluates it over the ABox using its SQLized rewriting, to obtain its *candidate answers* and their images. Candidate answers define a superset of the answers holding under the brave, AR and IAR semantics. Among the candidate answers, CQAPri identifies those which are not brave-answers by discarding the inconsistent images, that contain an edge of the conflict graph: an answer that has only such images does not hold under brave semantics. It also identifies the IAR ones, by checking whether there is some image whose assertions do not participate in any edges in the conflict graph, since such an image contains a cause such that none of its

8. Available for download at: https://www.lri.fr/~bourgaux/CQAPri

assertions is involved in a conflict. Finally, for brave-answers that are not found to be IAR-answers, deciding whether they are entailed under the AR semantics is done using the SAT encodings of Figure 2. Using consistent images instead of causes is not a problem here because the set of causes is included in the set of images and every consistent image contains a cause, so it is possible to consistently contradict every cause iff it is possible to consistently contradict every consistent image.

To explain why a query answer $\vec{a}$ belongs to one of the three classes Possible, Likely and Sure that correspond to $\mathcal{K} \models_{\mathrm{S}} q(\vec{a})$ and $\mathcal{K} \not\models_{\mathrm{S}'} q(\vec{a})$ for two semantics S and S', CQAPri provides *all* the explanations for $\vec{a}$ being a positive answer under the first semantics and a *single* explanation for it being a negative answer under the other one (i.e. a counter-example), together with the necessary and relevant assertions for both positive and negative answers. For Possible answers, we additionally provide the necessary and relevant assertions for explaining $\mathcal{K} \not\models_{\mathrm{IAR}} q(\vec{a})$ (which can be used e.g. to identify how to modify the ABox to make an answer hold under IAR semantics). Positive explanations are ranked as explained in Section 5.1: using the number of assertions for negative answers and positive brave and IAR-answers, and numbers of disjuncts and total number of assertions for positive AR-answers; for ranking the latter, the user can choose the priority between the two criteria.

Explanations are computed using the results on positive and negative answers from Section 5.2. We thus need the causes of the query answers as well as their conflicts. For the causes, CQAPri prunes the non-minimal images computed during the query answering phase. The conflicts are directly available from the previous steps.

For positive IAR-answers, CQAPri stores the causes that are without conflict during the query answering phase. Instead of halting at the first cause without conflict, which is enough to show an answer holds under IAR semantics, it passes in review all causes. For positive AR-answers, the SAT encoding is constructed for the query answering phase, and CQAPri uses the solver SAT4J to compute the MUSes (see Section 6, Berre and Parrain, 2010). Necessary and relevant assertions for positive answers are computed simply by taking the intersection and union of the explanations. For negative AR-answers, we rely on SAT4J to compute a smallest model of $\varphi_{\neg q} \wedge \varphi_{cons}$, as well as the necessary and relevant assertions with the encodings presented in Propositions 5.16 and 5.17 that we use to test every potentially relevant assertion, i.e. that appears in $\varphi_{\neg q}$. For negative IAR-answers, we choose to compute by default an arbitrary explanation in polynomial time (cf. Section 7.3 for the reason for this choice), but CQAPri can also provide a smallest explanation using the SAT solver to find a cardinality-minimal model of $\varphi_{\neg q}$. The relevant and necessary assertions are computed using Algorithm 4 that exploits Lemmas 5.21 and 5.22.

6.2 The CQAPri Benchmark

To evaluate CQAPri, we needed a DL-Lite$_{\mathcal{R}}$ KB with a large and inconsistent ABox. Very few experiments have been done on inconsistent KBs, and the only DL-Lite$_{\mathcal{R}}$ benchmark we found was not suitable for our purposes (cf. Section 8.2.2), so we designed our own.

Ontology We built our TBox over the LUBM$^{\exists}_{20}$ TBox (Lutz, Seylan, Toman, & Wolter, 2013), which provides an extended DL-Lite$_{\mathcal{R}}$ version of the well-known $\mathcal{ELI}$ TBox of the Lehigh University Benchmark (LUBM) (Guo, Pan, & Heflin, 2005) that describes the university domain. LUBM$^{\exists}_{20}$ differs from the original LUBM by the removal of the axioms that

go beyond DL-Lite$_\mathcal{R}$ but also by the addition of concept inclusions, many of which having existential restrictions on the right-hand side, and of subconcepts to increase the size of the ontology. LUBM$^\exists_{20}$ comprises 127 concepts, 27 roles and 202 positive inclusions. To enable contradictions, we added negative inclusions to state the disjointness of pairs of concepts or roles having the same closest super-concept or super-role. Such concepts and roles appear at the same level in the TBox (that is, have the same distance to the top concept Thing). We excluded a small number of such inclusions when they did not seem to reflect the intended meaning of the concepts or roles. Figure 3 illustrates how we added negative inclusions for an example concept AssistantProfessor. We added a total of 875 negative inclusions. This apparently huge number of constraints results from the many pairwise disjoint concepts or roles used in the TBox.

Datasets We generated ABoxes of increasing sizes with the Extended University Data Generator (EUGen) provided with LUBM$^\exists_{20}$ by setting its *data completeness* parameter (i.e. the percentage of individuals from a given concept for which roles describing this concept are indeed filled) to its default value of 95%, which seems realistic from the application viewpoint. All the generated ABoxes were found consistent w.r.t. our enriched TBox, which suggests that the added disjointness constraints were faithful to the reused benchmark. The size of these ABoxes ranges from 75,663 to 9,938,139 assertions, which corresponds to 1 to 100 universities in EUGen settings, and each ABox is included in the larger ones: the smallest corresponds to university 0, the largest to universities 0 to 99.

Inconsistencies were introduced by reviewing all of the assertions of the consistent ABox, and contradicting the presence of an individual in a concept assertion with probability p, and the presence of each individual in a role assertion with probability $p/2$. A contradicting assertion is built by stating that the considered individual also belongs to a disjoint but close concept, i.e. the two concepts have the same closest super-concept which is not the top concept Thing. Note that a concept may here be an unqualified existential role restriction. The contradicting assertion is added either explicitly or implicitly by choosing one of its specializations (obtained by query rewriting). The concept or role that is used to build the assertion which will actually be added to the ABox is chosen among all rewritings of all possible contradicting assertions with a uniform probability distribution. We chose to generate such inconsistencies because they seem quite natural in real applications (e.g. using by mistake AssistantProfessor in place of AssociateProfessor). Conflicting assertions thus introduced are in turn processed as described above to create a few more complex conflicts. Additionally, for every role assertion, its individuals are switched with probability $p/10$. We chose to generate such misuses of roles because they also seem quite natural mistakes and may lead to inconsistencies (e.g. inverting the Faculty and Course in a TeacherOf role assertion).

Example 21. Below are four assertions that have been created to conflict some assertions of the original consistent ABoxes[9].

- The assertion Subj4Course($Dept11.Univ1/GradCourse33$) contradicts the assertion Subj20Course($Dept11.Univ1/GradCourse33$) because these concepts are disjoint (Subj20Course $\sqsubseteq \neg$Subj4Course).

9. For readability, we have abbreviated some of the individual names, e.g. replacing *Department* by *Dept*.

- The assertion $\mathsf{Subj3Dept}(Univ462)$ was inserted to contradict the assertion $\mathsf{MastersDegreeFrom}(Dept22.Univ0/Lecturer2, Univ462)$: indeed, the range of $\mathsf{MastersDegreeFrom}$ is Univ ($\exists\mathsf{MastersDegreeFrom}^- \sqsubseteq \mathsf{Univ}$), which is disjoint with Dept ($\mathsf{Univ} \sqsubseteq \neg\mathsf{Dept}$), which has $\mathsf{Subj3Dept}$ as a subconcept ($\mathsf{Subj3Dept} \sqsubseteq \mathsf{Dept}$).

- The assertion $\mathsf{PublicationResearch}(DUMMY\_1\_1\_749, Dept18.Univ1/Course32)$ conflicts with $\mathsf{TakesCourse}(Dept18.Univ1/UndergradStud124, Dept18.Univ1/Course32)$. Indeed, $\mathsf{TakesCourse}$ has Course for range, which is disjoint with $\mathsf{Research}$ which is the range of $\mathsf{PublicationResearch}$.

- Finally $\mathsf{MemberOf}(Dept5.Univ1, Dept5.Univ1/UndergradStud47)$ is obtained by switching the two individuals of an assertion. Note that this may induce that an individual belongs to a totally different concept, since the domain and the role of a concept have often no common super-concept other than Thing. Such inversions generally yield a lot of conflicts. ◁

For each of the 100 universities that constitute our consistent ABoxes, we set $p = 0.002$ and generated 50 batches of conflicting assertions using the method described above. We obtain inconsistent ABoxes with increasing ratios of assertions involved in some conflict by adding the n first batches of conflicting assertions to each university of the original consistent ABox, n ranging from 1 to 50, that roughly leads to a percentage of assertions involved in some conflict varying from about 3% to about 46%. We consider that ABoxes with a few percent of assertions in conflicts are realistic, but we also built ABoxes with a huge number of conflicts in order to study the impact of the data quality on the efficiency of our approach.

Table 5 in the appendix displays the characteristics of the generated ABoxes in terms of size, inserted assertions, and number and percentage of assertions involved in conflicts of the ABoxes of our benchmark. Every ABox's id uXcY indicates the number X of universities generated by EUGen and the number of queries batches used to add conflicts Y. Note that our method ensures that $\mathsf{uXcY} \subseteq \mathsf{uX'cY}$ when $\mathsf{X} \leq \mathsf{X'}$ and $\mathsf{uXcY} \subseteq \mathsf{uXcY'}$ when $\mathsf{Y} \leq \mathsf{Y'}$.

Queries Table 3 summarizes the characteristics of the 20 queries used in our experiments (for the complete queries, consult Figure 4 in the appendix). They have between 1 and 8 atoms, with an average of 4.25 atoms. Their rewritings produced with Rapid have between 2 and 202,710 CQs, 23,185.95 on average. Queries $\mathsf{q14}$ to $\mathsf{q20}$ were used in experiments of the DL-Lite query answering systems[10] by Pérez-Urbina, Horrocks, and Motik (2009), Lutz et al. (2013), and Rosati et al. (2012), and we designed the others ourselves. Our rationale when building queries was to use some concepts that have disjoint specializations to get more chance to get answers that hold under AR semantics and not under IAR semantics (as such cases are the most challenging to handle and thus the most interesting for testing purposes).

10. These queries were available on the websites associated with these systems at the time we developed our benchmark, but the websites are no longer online.

id	shape	#atoms	#variables	#constants	#rewritings	rew.time (ms)
q1	chain	2	2	0	80	4
q2	chain	2	2	0	44	3
q3	tree	3	2	1	58	4
q4	dag	7	6	0	25	3
q5	dag	5	2	1	6,401	88
q6	dag	5	3	1	8,240	742
q7	tree	3	2	1	450	7
q8	tree	2	2	1	155	4
q9	dag	6	4	0	202,579	15,917
q10	dag	6	3	1	202,710	33,865
q11	chain	2	2	0	35	3
q12	atomic	1	1	0	44	3
q13	atomic	1	1	0	44	3
q14	dag	6	3	0	23	3
q15	chain	3	2	0	2	3
q16	dag	8	4	0	3,887	124
q17	chain	6	3	0	14,700	190
q18	tree	5	3	0	667	13
q19	dag	8	4	0	23,552	920
q20	chain	4	2	1	23	3

Table 3: Queries in terms of shape, numbers of atoms, variables, constants, rewritings, and rewriting time (Rapid).

7. Experimental Evaluation

We conducted experiments to empirically study the properties of our query answering and explanation frameworks. We study in particular the impact of varying the size of the ABoxes and the ratio of assertions involved in some conflicts on CQAPri behaviour.

7.1 Experimental Setting

All experiments reported in this work were run on an Intel Xeon X5647 at 2.93 GHz with 16 GB of RAM, running CentOS 6.8. Reported times are averaged over 5 runs.

We did not measure the time that our prototype takes to present the results (i.e. translating the answers back in the original terminology with the dictionary table of the database and printing the results in text files), since the goal of our experiments was to study the properties of the computation of query answers and explanations rather than their presentation (which a full-fledged OMQA system would probably handle in a more refined and efficient way).

We use the benchmark presented in Section 6. To test the explanation component of our tool, we explain all answers of the queries over all the ABoxes of our benchmark, except for those which have more than 200,000 answers, because it yields unreasonable experimental times. We note that in practice, a user will never examine the explanations of more than a

	c1	c5	c10	c20	c30	c50
u1	2,033	2,384	2,498	2,710	2,920	3,113
u5	7,758	8,542	8,920	9,644	10,462	11,230
u20	27,748	29,878	31,792	34,683	36,982	40,586
u50	73,031	78,673	80,563	91,122	100,134	118,375
u100	153,476	166,234	177,441	200,468	221,172	247,191

Table 4: Construction of conflict graph in milliseconds w.r.t. size uX and conflicts cY.

small number of answers, so there would typically be no need to generate explanations for (hundreds of) thousands of answers.

7.2 Experimental Results for Query Answering

We empirically study the properties of our consistent query answering framework by measuring the time spent in the different phases of query answering under IAR, AR and brave semantics, and how it varies with the size of the ABoxes and the ratio of assertions involved in some conflicts. We also consider how the repartition of the number of answers under the three semantics changes as the percentage of assertions in conflict increases.

The time it took CQAPri to be up and ready to answer queries is dominated by the construction of the conflict graph for the ABox (Table 4); it took about 2 seconds to load the TBox, construct the queries that correspond to the violation of negative inclusions, and open the PostgreSQL connection to the ABox. The time for the construction of the conflict graph has a linear behavior w.r.t. the size of the ABox, and the more conflicts there are, the higher the slope.

Table 6 shows, for each query-ABox pair, how many Sure, Likely and Possible answers were identified among the candidate answers. In this table, OOM means that CQAPri ran out of memory. In all of our experiments, we found only a few candidate answers that are not brave: only q9 got such answers, on all ABoxes (up to 802 inconsistent answers on u100conf50). Only 9 queries out of 20 got Likely answers over some ABoxes, 5 of them have only one or two atoms (q1, q2, q11, q12, q13) while the others are more complex (q5, q6, q9, q18). Such answers occur because these queries are general and involve concepts with many disjoint sub-concepts. For 67.5% of our query-ABox pairs, the AR semantics does not provide any additional answers compared to IAR semantics. However, all answers of q5 over the different uXc20 ABoxes hold under AR semantics but not IAR semantics. For such selective queries, using the AR semantics rather than IAR semantics may be necessary to identify some likely answers. Unsurprisingly, for a given ABox size, when the proportion of conflicting assertions increases, the number of Sure answers decreases while the number of Likely and Possible answers increases. This incurs a higher computational cost since a call to the SAT solver is needed for each non-IAR-answer to decide if it holds under AR semantics or not.

Figure 5 shows the evolution of the time spent by CQAPri for the query answering phase (consisting in identifying all Possible, Likely, and Sure answers) w.r.t. the size of the ABox, for three different ratios of assertions in conflict: a few percent (about 4%, uXc1),

as we expect to be the case in most real applications, about 30% (uXc20), or about 45% (uXc50). Figure 6 shows the evolution of the time spent by CQAPri for query answering w.r.t. the proportion of ABox assertions involved in some conflicts, for small (u1cY), intermediate (u20cY), and large (u100cY) ABoxes. Figure 7 shows the proportion of the query answering time spent in rewriting the query, executing the rewritten query to get candidate answers, filtering the IAR- or not brave-answers, and identifying the AR-answers among the remaining answers.

In Figure 5, we remark two outlier queries, q4 and q9, whose answering times have an exponential-like growth w.r.t. ABox size even when only a small fraction of the assertions participate in conflicts. Query q4 is very sensitive both to ABox size and ratio of conflicts, while q9 is rather robust to conflicts. This behaviour seems due to the uncommon characteristics of these queries. Indeed, q9 has 202,579 rewritings, with 4 variables and no constant, that leads to a very costly execution over the database, as illustrated on Figure 7 where almost 90% of the time is spent on executing the rewritten query even in the case of the largest ABox with the highest percentage of conflicts. Query q10 that differs from q9 only by the introduction of a constant, behaves very differently since its answering time stabilizes quickly. As for q4, it has a high number of atoms and variables, which are all free, yielding a huge number of answers (10,362,220 answers on u100c10), which become quickly non-IAR since their causes involve lots of assertions. These two queries are interesting to challenge our system, but are not realistic, especially q4.

For the other queries, CQAPri scales up to large ABoxes when the proportion of assertions involved in some conflict is only a few percent, and even a few tens percent for most of the queries. The increase in the number of non-IAR-answers when the proportion of conflicts increases generally significantly augments the time spent in this last phase. This explains our observation that the lower the ratio of conflicts, the more query answering time shows a linear behaviour w.r.t. the ABox size.

The average time to classify an answer is generally under the millisecond, and at most 7 ms (for q19 on u20c50). The average time to decide whether a brave and non-IAR-answer holds under AR semantics is generally a little higher but also under the millisecond.

Overall, CQAPri scales well in settings ranging from ones we consider realistic to more artificial ones. Our experiments thus demonstrate that the AR-answers can be computed in practice, and our analysis shows that this is due to the fact that the IAR semantics often constitutes a very good approximation of the AR semantics.

7.3 Experimental Results for Query Answer Explanation

To assess the practical interest of our explanation framework, we empirically study the properties of our implementation, in particular: the impact of varying the percentage of assertions in conflict, the typical number and size of explanations, and the extra effort required to generate cardinality-minimal explanations for negative IAR-answers rather than arbitrary ones.

Tables 7-9 show the number of answers from each class for each query, as well as the distribution of the explanation times for these answers, for ABoxes of growing sizes and ratios of conflicts. (We recall that for our explanation experiments, we exclude queries that have more than 200,000 answers, which is why queries q8, q10, q17, and q19 do not

appear in these tables.) Figures 8 and 9 show the time spent in explaining *all* query answers w.r.t. ABox size and proportion of conflicting assertions respectively. Figure 10 displays the proportion of time spent in the different phases to explain all query answers for ABoxes of growing difficulty. The explanation cost, given by the two upper bars, consists in pruning non-minimal consistent subsets of the ABox entailing the answers to get the causes, and computing the explanations from the causes and conflicts. The three lower bars relate to the query answering phase, which consists in rewriting and executing the query to get the candidate answers, and identifying Sure, Likely, and Possible answers (classify).

We summarize the general tendencies we observed. Regarding the time needed to produce explanations, the main conclusion is that explaining a *single* query answer is always feasible and fast ($\leq$1s) when there are a *few* percent of conflicts in the ABox (Tables 7-9, uXc1 case), as is likely to be the case in most real applications. Even with a *high* percentage of conflicts, the longest time observed is below 20s (19.5s for explaining a Possible answer of q9 on u100c50), and remains lower than 1s for *small* ABoxes (up to u20cY case, i.e. 2 million assertions), and lower than 8s for a *significant* percentage of conflicts (uXc20 case, i.e. 30% of assertions in conflict). In *all* the experiments we made, explaining a *single* answer typically takes less than 10ms, rarely more than 1s. However, computing explanations of *all* answers can be prohibitively expensive when there are very many answers, which is why we do not produce them all by default.

Adding conflicts to the ABox complicates the explanations of answers, due to their shift from the Sure to the Likely and Possible classes. Explaining such answers indeed comes at higher computational cost, as can be seen in Figures 8, 9 and 10. Compared to query answering, we found that explaining all query answers is more sensitive both to ABox size and ratio of conflicts.

We observed that the average number of explanations per answer is often reasonably low, although some answers do have a large number of explanations. For instance, on the ABoxes u100cY, there were less than 10 explanations on average, but this number varies from about 1 to more than 400, and for u100c50, we got up to 4560 explanations for an AR-answer (and up to 741 causes for a brave-answer). Even on small ABoxes (u1cY), we got up to 693 explanations for a brave-answer and 243 explanations for an AR-answer. Regarding the size of explanations of AR-answers, the number of causes in the disjunction was up to 25 (for a q12 Likely answer on u100c50; up to 5 on the u1cY ABoxes), showing the practical interest of ranking the explanations.

7.3.1 Explaining Negative Answers

Our prototype is able to explain $\mathcal{K} \not\models_{S'} q(\vec{a})$ by providing a (possibly smallest) explanation for $\mathcal{K} \not\models_{S'} q(\vec{a})$, together with the relevant and necessary assertions for $\mathcal{K} \not\models_{S'} q(\vec{a})$. We explain here why we chose to compute an arbitrary explanation for $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$ by default rather than a cardinality-minimal one. We also give some insight into the explanation times for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ and the contribution of the computation of necessary and relevant assertions to the overall runtime.

We considered the following four cases in our experiments:

- Case 1 is our default setting, in which we compute an arbitrary explanation for negative IAR-answers, a smallest explanation for negative AR-answers, and the necessary and relevant assertions for explaining negative answers,
- Case 2 differs from Case 1 in omitting the computation of the necessary and relevant assertions for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ and $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$,
- Case 3 differs from Case 1 in omitting the computation of the relevant assertions for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$,
- Case 4 differs from Case 1 in computing a cardinality-minimal explanation for $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$ instead of using a polynomial-time procedure to generate an arbitrary one.

Table 10 displays, for each query, the number of Likely and Possible answers the query possesses, and the distribution of the times for explaining $\mathcal{K} \not\models_{S'} q(\vec{a})$ in our default setting (Case 1). If we compare these distributions with those of Tables 7-9, we can see that for many queries, there is the same number of answers having the longest explanation times (columns $[0.1, 1[$ and >1) when only the negative answer is explained as in the case where both $\mathcal{K} \models_S q(\vec{a})$ and $\mathcal{K} \not\models_{S'} q(\vec{a})$ are explained. This shows that for many queries, the difficulty comes from explaining $\mathcal{K} \not\models_{S'} q(\vec{a})$.

Cost of relevant and necessary assertions Table 11 shows the same information as Table 10, in the case where the necessary and relevant assertions for explaining $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$ or $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ are not computed. In this case, almost all negative answers are explained in less than 10ms. This shows that the main part of the explanation time for negative answers is spent in computing these assertions. Note that for negative IAR-answers (Likely answers), most of the explanation times were already below 10ms in our default case.

Computation of relevant assertions for negative AR-answers Since computing the necessary and relevant assertions for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ appears to be costly, we investigate further to see how this cost is distributed and if it is possible to reduce the cost of negative explanation without losing too much information. For negative AR-answers, deciding if an assertion α appears in some explanation is an NP-complete problem, and deciding if it appears in all explanations is coNP-complete. In practice the problem of finding the necessary assertions can be solved efficiently because for a negative AR-answer, the SAT solver has already found a model of $\varphi_{\neg q} \cup \varphi_{cons}$ during the query answer classification phase, so checking whether $\varphi_{\neg q} \cup \varphi_{cons} \cup \{\neg x_\alpha\}$ is unsatisfiable is trivial when α does not appear in this model, and if it does, the SAT solver may reuse what it has already computed for a closely related problem. Deciding if α is relevant is more difficult because the encoding we use differs more from $\varphi_{\neg q} \cup \varphi_{cons}$. Indeed, if we compare the execution times with (Table 10, for Possible answers) and without (Table 12) the computation of the relevant assertions for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$, we observe significant differences: for all queries and ABoxes, at least 60% of the negative AR-answers are explained in less than 10ms (only 6.45% for q19 on u100c50 in Case 1), and less than 0.15% of them need more than 1s (12.72% of the negative AR-answers of q12 on u100c50 in Case 1). Moreover, the longest time required to explain an answer in Case 1 is generally significantly longer than in Case 3 (for instance on u100c20, the longest time required to explain an answer is 8s in Case 1, while it is 5s for Case 3).

We therefore tried to obtain an approximation of these assertions that is fast to compute. The assertions relevant for $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$ can be computed very quickly and provide a

superset of those relevant for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$, since the explanations for $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ are the consistent explanations for $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$. However, in our experiments, those two sets of assertions differ quite significantly, and when they do the difference may be huge (hundreds of assertions instead of one to four assertions for some answers of q12 on u100c20). When the ABox size and ratio of conflicts increase, the proportion of answers having additional relevant assertions for $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$ and the difference between the two sets of relevant assertions for $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$ and $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ increase. The two sets always coincide on u1c1, while on the uXc50 ABoxes, they differ in up to 100% of the Possible answers of a query (up to 27 assertions for u1c50, up to 651 assertions for u100c50). On u100c1 they differ in up to 5.6% of the Possible answers of a query, and up to 501 assertions. This shows that there is no straightforward way to reduce the computation of relevant assertions for negative AR-answers without sacrificing too much information.

Cardinality of explanations for negative IAR-answers Although smallest explanations for negative answers may be preferred, we found it worthwhile to use a polynomial-time method to obtain an arbitrary explanation for a negative IAR-answer rather than relying on the SAT solver to generate a smallest such explanation.

Indeed, computing an arbitrary explanation for $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$ always takes less than 100ms (see Table 10, Likely answers), and in almost all cases less than 10ms.

By contrast, when a *smallest* explanation is computed (Table 13), more time may be needed. A striking case is that of query q12: on u20c20, almost 19 minutes are spent in computing a smallest explanation for *one* negative IAR-answer, and on u100c20, it takes around 50 minutes, while computing an arbitrary explanation was done in less than 10ms for every ABox and negative IAR-answer of q12 (Table 10). This even leads to a time-out for ABoxes from u20c30. This long explanation time is due to the unusual size of the explanation (18 assertions for the answer on u100c20, whereas other explanations for negative answers typically contained only a few assertions).

As for the size of explanations, we found that on u100c20, the arbitrary explanations generated for all negative IAR-answers of q5, q6, q11, and q12 have exactly the same size as the smallest explanations found with the SAT solver, that only one negative IAR-answer of q13 had a suboptimal explanation (with 4 assertions instead of 3), as well as about 61% of the negative IAR-answers of q9, whose explanations contain at most two assertions more than the smallest ones.

The possibly very high additional cost of computing a smallest explanation for a limited benefit in terms of the size of explanations led us to adopt arbitrary explanations for negative IAR-answers as the default setting in our system. However, note that this very high additional cost concerns very few answers: for instance all the other negative IAR-answers of q12 on u20c20 are explained in less than 10ms. One could therefore envision a mixed approach in which one allocates a short amount of time to try to find a smallest explanation, falling back to arbitrary explanations if no smallest explanation is identified within the allotted amount of time.

8. Related Work

This section provides an overview of different lines of research that are related to our own.

8.1 Inconsistency-Tolerant Semantics

There have been several different inconsistency-tolerant semantics that have been proposed for DL KBs. We will only give a brief discussion of the literature here and refer the reader to the recent survey by Bienvenu and Bourgaux (2016) for further details and references.

The survey by Bienvenu and Bourgaux compares different semantics w.r.t. the properties they satisfy and the set of answers they produce. Two key semantic properties are considered: Consistent Support and Consistent Results. A semantics $\mathcal{S}$ (with entailment relation $\models_{\mathcal{S}}$) satisfies the Consistent Support property if for every KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, query q, and tuple $\vec{a}$, if $\mathcal{K} \models_{\mathcal{S}} q(\vec{a})$, then there exists a $\mathcal{T}$-consistent subset $C \subseteq \mathcal{A}$ such that $\langle \mathcal{T}, C \rangle \models q(\vec{a})$ (in our terminology, there is a cause for $q(\vec{a})$). This is a desirable property as it means that a query answer can always be justified by exhibiting a consistent subset of the KB which yields the answer. The brave, AR, and IAR semantics all satisfy this property, with the brave semantics being the most permissive semantics with this property. By contrast, variants of the AR and IAR semantics based upon repairs of the closed ABox (the so-called CAR and ICAR semantics, introduced by Lembo et al., 2010) do not satisfy this property. The second property, Consistent Results, holds for a semantics $\mathcal{S}$ just in the case that for every KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $\mathcal{I} \models q(\vec{a})$ for every $q(\vec{a})$ with $\mathcal{K} \models_{\mathcal{S}} q(\vec{a})$. This property thus requires that query results obtained from a KB are jointly consistent with the TBox, meaning that users can safely combine query results without risking contradiction. As discussed in Section 3, this property is verified by both AR and IAR semantics, but not by the brave semantics.

The comparison of semantics by Bienvenu and Bourgaux (2016) also considers the relationships of over- and under-approximation that hold between the different semantics. We have seen in Section 3 that the IAR and brave semantics provide respectively over- and under-approximations of the AR semantics, a fact that we exploit in our system. To obtain finer approximations, the families of k-support and k-defeater semantics were proposed (Bienvenu & Rosati, 2013). It was shown that the k-support semantics (resp. k-defeater semantics) contain the IAR (resp. brave) semantics as special cases, converge to the AR semantics in the limit, and possess the same desirable semantic and computational properties (in particular, AC^0 data complexity) as the IAR (resp. brave) semantics. We observe that these semantics can be elegantly defined in terms of our notions of explanations: a tuple $\vec{a}$ is an answer to $q(\vec{x})$ under the k-support semantics iff $\mathcal{K} \models_{\text{AR}} q(\vec{a})$ has an explanation consisting of at most k causes, and it is an answer to $q(\vec{a})$ under the k-defeater semantics iff there is no explanation of $\mathcal{K} \not\models_{\text{AR}} q(\vec{a})$ of size at most k. Thus, the input parameter k used by these semantics is naturally interpreted as the size of explanations for $q(\vec{a})$ being a (non)-answer under AR semantics.

We note in passing that another parameterized family of semantics, the k-lazy semantics (Lukasiewicz, Martinez, & Simari, 2012), has been proposed as a compromise to interpolate between the IAR and AR semantics, but these semantics have less desirable semantic and computational properties than the k-support semantics (as has been argued by Bienvenu & Rosati, 2013; Bienvenu & Bourgaux, 2016). We also point out that a recent work by Benferhat, Bouraoui, Croitoru, Papini, and Tabia (2016) introduces a new semantics, called non-objection inference, that is an over-approximation of the AR semantics and an under-approximation of the brave semantics. This semantics is defined as follows: a tuple $\vec{a}$ is

an answer to $q(\vec{x})$ under non-objection semantics iff (i) there is some repair $\mathcal{R}$ such that $\langle \mathcal{T}, \mathcal{R} \rangle \models q(\vec{a})$ and (ii) for every repair $\mathcal{R}$, there is a model $\mathcal{I}$ of $\langle \mathcal{T}, \mathcal{R} \rangle$ in which $q(\vec{a})$ holds. Interestingly, unlike the brave semantics, the non-objection semantics satisfies the Consistent Results property. Moreover, ground CQ answering (i.e., Boolean CQs without existentially quantified variables) is in PTime w.r.t. data complexity for DL-Lite.

Most of the semantics discussed so far are based upon the notion of repair, defined as an inclusion-minimal $\mathcal{T}$-consistent subset of the ABox. However, just as we introduced preferences to isolate the most interesting explanations, it is natural to use preferences to focus on the most likely repairs. In the DL setting, the idea was first explored by Du, Qi, and Shen (2013), who considered a weight-based variant of the AR semantics to query highly expressive description logics. Closer to the present work, variants of the IAR, brave, and AR semantics based upon several notions of preferred repair (namely, cardinality, prioritized set inclusion, prioritized cardinality, and weights) have been considered by the authors of this paper (Bienvenu et al., 2014; Bourgaux, 2016). We showed that in most cases the addition of preferences increases the complexity of query answering over DL-Lite KBs. However, for preferred repairs based upon prioritized set inclusion, query answering under modified AR and IAR semantics is coNP-complete w.r.t. data complexity, and thus no higher than for 'plain' AR semantics. This led us to extend the SAT encoding from Section 4 to these semantics and to implement the resulting encodings in CQAPri. The main result of the experiments was that the addition of preferences makes query answering more challenging. We note that a variant of non-objection semantics based upon cardinality-maximal repairs has been considered by Benferhat et al. (2016) and unsurprisingly, it was shown to have intractable data complexity.

In this work, we chose to focus on the AR semantics, as it is the most widely studied semantics and arguably the most reasonable. We additionally considered the IAR and brave semantics, which can be seen as the most and least permissive semantics, in order to classify answers based upon their reliability and to aid us in computing the answers under AR semantics. It would be interesting to experiment with other approximations of AR semantics that offer polynomial data complexity, like the k-support semantics, k-defeater semantics, and the non-objection semantics.

8.2 Systems and Benchmarks for Inconsistency-tolerant Query Answering

We next discuss previous implementations and testing of algorithms for inconsistency-tolerant query answering.

8.2.1 Systems for Inconsistency-tolerant Query Answering

In terms of implemented tools and benchmarks for inconsistency-tolerant query answering over DL KBs, we are aware of three systems: the QuID system (Rosati et al., 2012; Lembo et al., 2015) that handles IAR semantics for CQs and DL-Lite$_\mathcal{A}$ KBs, the SaQAI system (Tsalapati et al., 2016) that handles IAR and ICAR semantics for CQs and DL-Lite KBs, and the system by Du et al. (2013) for querying $\mathcal{SHIQ}$ KBs under a variant of AR semantics with weight on ABox assertions that handles CQs without non-distinguished variables (which reduces to the simpler task of ABox assertion entailment). None of these

systems is directly comparable to our own, since they employ different semantics, and in the case of the system of Du et al. target different DLs and queries.

The QuID system implements three approaches for IAR query answering: query rewriting, annotated ABox, and ABox cleaning. The query rewriting approach consists in rewriting the query and evaluating the rewritten query over the original inconsistent ABox (see Section 3 for the intuition on how rewritings for IAR semantics can be constructed). In the annotated ABox approach, the ABox is annotated with information about assertions that belong to some conflict, and at query time, we enrich the original query with conditions to filter out such assertions. The cleaning approach evaluates the original query over a 'cleaned' version of the ABox, in which every assertion involved in some conflict has been removed (which yields precisely the intersection of repairs). Our approach regarding IAR semantics is closer to the last two approaches, since our preprocessing phase where we compute and store the conflicts is similar in spirit to the annotation and extraction of the intersection of the repairs. The main difference is that we do not modify the query to retrieve only IAR-answers at evaluation time but rather filter out IAR-answers from candidate answers by checking a posteriori that they have causes without conflict.

The SaQAI system computes the intersection of the repairs by computing the conflicts and removing them from the ABox, then saturates the ABox obtained before performing standard query evaluation. The experiments reported by Tsalapati et al. (2016) show that for IAR query answering, CQAPri performances are comparable to those of QuID and often better, while SaQAI is much more efficient. This is not surprising given that SaQAI uses saturation while the others use query rewriting. However, cleaning and saturating the ABox is not always possible (if the ABox cannot be modified). Moreover, such an approach is not compatible with the AR semantics, which avoids the loss of information that results from discarding all assertions involved in conflicts.

We can observe some high-level similarities with Du et al.'s system which also has a preprocessing phase that compiles the KB, then employs SAT solvers and uses a reachability analysis to identify a query-relevant portion of the KB to do query answering.

There are also a few systems for querying inconsistent relational databases. Most relevant to our work is EQUIP (Kolaitis, Pema, & Tan, 2013), which reduces AR conjunctive query answering in the presence of primary key constraints to binary integer programming (BIP). Similarly to our system, EQUIP first computes the IAR-answers and the causes of the answers with their conflicts. The encoding for AR semantics consists in a first part that encodes the repairs, enforcing that exactly one tuple of each group of same-key tuple is selected, and a second part that ensures that the repair contains no cause. The main difference with CQAPri is that instead of building and solving one encoding for each answer, only one is built using variables to represent the different answers, so that setting them to 1 trivializes the equations related to the causes of this answer. The answers that do not hold under AR semantics are computed iteratively, by minimizing the sum of the answers variables, that are then set to 1 when found not AR, until the system becomes unsatisfiable. We considered using BIP rather than SAT solvers but were not convinced by our preliminary experiments. Some systems focus on cases where consistent query answering is tractable, for restricted types of constraints or queries, using first-order rewritings (ConQuer, Fuxman & Miller, 2005; Fuxman, Fazli, & Miller, 2005) or conflict-hypergraphs (Hippo, Chomicki et al., 2004a, 2004b). Other systems handle more general constraints that can lead to dif-

ferent kinds of repairs, since for databases it may be necessary to insert or modify tuples to restore consistency. For instance, ConsEx (Marileo & Bertossi, 2010) reduces AR query answering to answer set programming (ASP) by building repair programs such that there is a one-to-one correspondence between stable models and repairs. Experiments reported by Kolaitis et al. (2013) show that EQUIP outperforms ConsEx on its restricted setting that is closer to our own.

8.2.2 Experimental Settings Involving Inconsistent DL-Lite KBs

The QuID benchmark QuID is evaluated using the LUBM TBox containing 43 concepts, 25 roles, 7 attributes and about 200 positive inclusions, approximated in DL-Lite by eliminating the inclusions that go beyond DL-Lite, then enriched with 10 negative inclusions, 5 identifications (that state that some sets of properties identify the instances of some basic concepts), and 3 denial constraints given by Boolean CQs (of the form $\exists \vec{y}\, \psi(\vec{y})$, where ψ is a conjunction of atoms using variables from $\vec{y}$) which have to be false.

Datasets are generated with the UBA Data Generator provided with LUBM, for 1, 5, 10 and 20 universities, leading to a size varying from about 100K to about 2 million assertions. For each of these consistent ABoxes, four inconsistent ABoxes are constructed by adding 1%, 5%, 10% and 20% of assertions that are in conflict. The main difference with our setting is the way conflicts are generated. Indeed, the original ABox produced by the generator is left consistent, while each assertion that is added is in conflict with others. In practice, for a growing n, the same n fresh individuals are assigned to all eleven concepts that appear in some negative inclusion, and the same n pairs of fresh individuals are assigned all five roles (or inverse roles) that appear in some negative inclusion. While such way of adding conflicts may make sense for evaluating the QuID system that implements IAR semantics, which only needs to ignore the assertions that are in some conflicts, it is not realistic at all because the new individuals are inserted in many concepts that are semantically very far from each other, like Person, Publication, Course, and Organization. We could therefore not use these datasets to evaluate CQAPri, since there is no chance obtaining query answer that hold under AR semantics but not under IAR semantics.

Du et al.'s benchmark In Du et al.'s later work (2015) on abduction over inconsistent DL-Lite KBs, the Semintec and LUBM TBoxes without the non-DL-Lite axioms are used. The authors added negative inclusions to LUBM in a similar fashion to ours.

Regarding data, Semintec has a small ABox of about 65K assertions and the number of universities used for LUBM datasets ranges from 1 to 100 as in our experimental setting. Du et al. (2013) present a tool called Injector to insert conflicts in ABoxes. Given a consistent KB and a number of conflicts to be inserted, Injector randomly selects a functional role or an atomic concept that has disjoint atomic concepts. If the KB already entails assertions that correspond to that role or concept, Injector randomly selects such an assertion and adds an assertion that conflicts it: in the case of a functional role, it relates the corresponding individual of the ABox to a new individual with that role, and in case of concept assertions, it assigns the individual to one of the disjoint atomic concepts. Otherwise, Injector adds two assertions that are in conflict in the same way using fresh individuals. This way of adding conflicts is much more realistic than that of QuID. Even if it is similar in spirit to ours, since it tries to distribute randomly conflicts over the ABox, there

are some differences. First, we randomly select assertions of the initial ABox rather than concepts or roles that may be contradicted, which leads to a repartition of the conflicts that respects the structure of the data (since there may be lots of assertions for some concepts and few or no assertions for others). Second, we do not use only atomic concepts to build contradictions but also unqualified existential role restrictions. Finally we take into account another kind of possible errors by switching role individuals. Du et al. (2015) added 0 to 400 "conflicts" (i.e. assertions that contradict an original assertion, or inconsistent pairs of assertions) to the consistent ABoxes. We prefer to quantify the degree of inconsistency in terms of assertions involved in some conflicts rather than in terms of assertions added to the consistent ABox, since we can compute the ratio of conflicts (as we define them) of a real dataset, but not its ratio of erroneous facts. However, note that we added many more assertions than Du et al. for original datasets of the same size.

8.3 Related Work on Explanation

We now discuss related work on explaining entailments and query (non-)answers.

8.3.1 Justifications of Entailed Axioms

As mentioned in the introduction, there has been significant interest in equipping DL reasoning systems with explanation facilities. The earliest work proposed formal proof systems as a basis for explaining concept subsumptions (McGuinness & Borgida, 1995; Borgida, Franconi, & Horrocks, 2000), while the post-2000 literature mainly focuses on *axiom pinpointing* (Schlobach & Cornet, 2003; Kalyanpur, Parsia, Sirin, & Hendler, 2005; Horridge, Parsia, & Sattler, 2012), in which the problem is to generate minimal subsets of the KB that yield a given (surprising or undesirable) consequence. Such subsets are often called *justifications*. It should be noted that work on axiom pinpointing has thus far focused on explaining entailed TBox axioms (or possibly ABox assertions), and in particular on TBox debugging by explaining unsatisfiable classes. In our work, we assume that the TBox has been properly debugged, so is consistent and correct, i.e. all consequences of the TBox are desirable. This work on axiom pinpointing can therefore be seen as a first step that allows us to be sure that the errors stem from the data.

For the lightweight DL $\mathcal{EL}+$, justifications have been shown to correspond to minimal models of propositional Horn formulas and can be computed using SAT solvers (Sebastiani & Vescovi, 2009); a polynomial algorithm has been proposed to compute one justification (Baader, Peñaloza, & Suntisrivaraporn, 2007). In DL-Lite, the problem is simpler: all justifications can be enumerated in polynomial delay (Peñaloza & Sertkaya, 2010).

Beside computing justifications efficiently, several works addressed the problem of making them understandable for the user, either by studying their cognitive complexity (Horridge, Bail, Parsia, & Sattler, 2011), or by grouping justifications that have a similar structure to help to handle large number of justifications (Bail, Parsia, & Sattler, 2013). Our experiments showed that a query answer can possess a very large number of explanations, many of which are quite similar in structure. It could therefore be interesting to investigate ways of improving the presentation of explanations, e.g. by identifying and grouping similar explanations as has been done for justifications, or by adopting a factorized representation (like in the work by Olteanu & Zavodny, 2012).

8.3.2 Explanation of Query Answers

The problem of explaining answers to conjunctive queries over DL-Lite KBs is considered by Borgida et al. (2008) who provide a proof-theoretic approach to explaining positive answers. The proof of an answer involves the ABox assertions and TBox axioms used to derived it. As mentioned in Section 5, the difficulty of such proofs could provide an additional criteria for ranking explanations, and the work on the cognitive complexity of justifications may give clues on this difficulty.

Probably the closest related work is by Arioua, Tamani, and Croitoru (2015) who introduce an argumentation framework for explaining positive and negative answers under the inconsistency-tolerant semantics ICR. Their motivations are quite similar to our own, and there are some high-level similarities in the definition of explanations (e.g. to explain positive ICR-answers, they consider sets of arguments that minimally cover the preferred extensions, whereas for positive AR-answers, we use sets of causes that minimally cover the repairs). They propose to compute one explanation for a positive or negative ICR-answer with a hitting set algorithm, applied either on the sets of supporting arguments (which correspond to our causes) present in each extension (corresponding to the repair), or on the set of attacking arguments (which correspond to the conflicts of the causes). Our work differs from theirs by considering different semantics and by providing detailed complexity analysis, in which we do not assume that the set of repairs is given, and an implemented prototype. Another argumentation framework has been proposed for ground BCQ explanation under IAR and brave semantics by Arioua and Croitoru (2016a). They consider that a brave-answer is explained if an argument supporting it has been found and that an IAR-answer is explained if an argument without attacking argument has been found. Contrary to our work, the focus is not on the computation of the explanations, since the arguments are considered given, but on the characterization of the dialogue between a proponent and an opponent to the answer exchanging arguments, depending on the semantics under which it is entailed. The same kind of study has then been conducted for the AR semantics (Arioua & Croitoru, 2016b). This framework has been implemented in the DALEK prototype (Arioua, Croitoru, & Buche, 2016). To support the explanatory dialogue and build (counter)arguments, DALEK computes the set of conflicts of the KB in a similar way to our system, as well as the causes of the claim. However, the explanations themselves are not computed but result from the interaction between the user and the system.

Finally, we note that the problem of explaining query results has been studied in the database community (cf. Cheney, Chiticariu, & Tan, 2009, for a survey). The *lineage* of a query answer is the set of tuples of the database that contribute to produce the answer, i.e. the union of the images for the answer. The *why-provenance* corresponds to the images and the *minimal witness basis* to the set of causes of the answer. The *how-provenance* describes how a result was produced from the tuples. The *where-provenance* provides the location (i.e. relation, tuple and attribute) of the values in the answer tuple.

8.3.3 Query Abduction

Since we defined explanations for a negative AR- or IAR-answer using assertions of the ABox that conflict its causes, defining explanations for negative brave-answers (which have no cause) would significantly differ in spirit. The problem of explaining negative answers

has been primarily seen as the problem of finding a minimal dataset to be added to the data to get the missing answers, i.e. as *query abduction*, both in the DL literature (see e.g. Calvanese et al., 2013; Du et al., 2014, for DL-Lite and Wang, Chitsaz, Wang, and Du, 2015, for $\mathcal{ELH}_\bot$) and in the database arena (see e.g. Herschel & Hernández, 2010). In both settings, restrictions on the signature of the explanation are allowed. Note that a different approach related to query debugging was proposed in the database context (Bidoit, Herschel, & Tzompanaki, 2014), and focuses on finding subqueries responsible for pruning the missing answer from a query result. This approach is less relevant to our setting since conjunctive queries over DL KBs are much simpler and less error-prone than SQL queries.

Calvanese et al. (2013) studied the complexity of the decision problems related to explaining negative answers in DL-Lite (recognition and existence of an explanation, necessity and relevance of an assertion). Du et al. (2014) presented an implementation that computes explanations in DL-Lite and later treated the case of inconsistent KBs (2015). In this case, an explanation is a set of assertions to add that will lead to the answer holding under IAR semantics. These three papers tackle the issue of preferred explanations in different ways. Calvanese et al. consider subset- or cardinality-minimal explanations. Du et al. introduce the notion of representative explanation, which is an explanation that is minimal (when allowing renaming of fresh individuals to compare explanations) and is not subsumed by any other (e.g. if $\mathsf{Advise}(ann, ann)$, $\mathsf{Advise}(ann, bob)$, and $\mathsf{Advise}(ann, ind)$ where ind is a fresh individual are the explanations for ann not being an answer to $\exists y \mathsf{Advise}(x, y)$, the last one is the unique representative explanation). Du et al. also consider cardinality-minimal preferred explanations for a preference relation based on cardinality-preserving substitutions.

We could build on the work on query abduction to define explanations for negative brave-answers as minimal sets of assertions $\mathcal{E}$ such that $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{E} \rangle \models_{\text{brave}} q(\vec{a})$. Note that if we define explanations in this way, then $\mathcal{E}$ is an explanation for $q(\vec{a})$ being a negative brave answer in $\langle \mathcal{T}, \mathcal{A} \rangle$ iff $\mathcal{E}$ is an inclusion-minimal explanation for $q(\vec{a})$ being a negative answer in $\langle \mathcal{T}_P, \mathcal{A} \rangle$, where $\mathcal{T}_P$ is the set of positive inclusions in $\mathcal{T}$, *and* $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{E} \rangle \models_{\text{brave}} q(\vec{a})$. Since Du et al. have shown that computing all minimal explanations is polynomial w.r.t. data complexity, we can build all explanations for $\langle \mathcal{T}, \mathcal{A} \rangle \not\models_{\text{brave}} q(\vec{a})$ by first computing all minimal explanations for $\langle \mathcal{T}_P, \mathcal{A} \rangle \not\models q(\vec{a})$, and then pruning those that does not respect the condition $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{E} \rangle \models_{\text{brave}} q(\vec{a})$. It follows that all of the decision and generation problems that we consider are in PTime w.r.t. data complexity, and the implementation would be a straightforward adaptation of the method of Du et al. However, as argued by Du et al., the number of such explanations can be too large to be computed in practice, and it would probably be relevant to adopt their notion of representative explanations.

The idea of representative explanation could also be used for presenting the notions of explanations proposed in the present paper, by treating the individuals that are mapped to existentially quantified variables in the same way as Du et al. treat fresh individuals.

9. Conclusion and Perspectives

Prior work on inconsistency-tolerant querying in DL-Lite left open whether the IAR constitutes a good approximation of AR semantics, as well as whether the AR semantics can be feasibly computed in practice. Encouraged by the performance of modern-day SAT solvers,

we proposed a practical SAT-based approach for query answering in DL-Lite$_\mathcal{R}$ under the AR semantics, using the IAR and brave semantics as tractable lower and upper bounds. We then devised a framework for explaining query (non-)answers over DL KBs under these three semantics and studied the computational properties of our framework, focusing on DL-Lite$_\mathcal{R}$. For intractable explanation tasks, we exhibited tight connections with variants of propositional satisfiability. We implemented both inconsistency-tolerant query answering and the explanation framework in our CQAPri system and built a benchmark for inconsistency-tolerant query answering upon a well-established DL-Lite$_\mathcal{R}$ benchmark. Our experiments show that CQAPri scales up to large ABoxes for realistic to challenging ratios of conflicting assertions. We thus show that (i) the AR semantics can be computed in practice and that this is due to the fact that the IAR semantics often constitutes a very good approximation of the AR semantics, and (ii) our explanation framework is of practical interest since explanations of query (non-)answers are generated fast overall.

We focused on DL-Lite$_\mathcal{R}$ for simplicity and because it is the basis for the W3C standard OWL 2 QL. Most of our results actually hold for other dialects of the DL-Lite family, but the problem of reasoning with the inconsistency-tolerant semantics we consider becomes considerably harder for many other well-known DLs. Regarding DL-Lite dialects, we recall that DL-Lite$_{\mathsf{core}}$ is the core language of the family and amounts to DL-Lite$_\mathcal{R}$ without role inclusions, and DL-Lite$_\mathcal{F}$ extends DL-Lite$_{\mathsf{core}}$ with functionality axioms on roles or on their inverses of the form (funct S). DL-Lite$_\mathcal{A}$ extends DL-Lite$_{\mathsf{core}}$ with both role inclusions and functionality with the restriction that functional roles cannot be used positively on the right-hand side of a role inclusion, and also allows for attributes (binary relations between objects and values from concrete domains, e.g. strings, integers). The complexity of query answering, consistency checking and computation of the causes for a query and the conflicts of a knowledge base are the same for all these languages, and the size of the conflicts is at most two in all cases. Therefore, all complexity upper bounds presented hold for DL-Lite$_{\mathsf{core}}$, DL-Lite$_\mathcal{R}$, DL-Lite$_\mathcal{F}$ and DL-Lite$_\mathcal{A}$. Moreover, since all our algorithms only need the causes and conflicts, they can be used without any modification. To make our prototype CQAPri able to handle KBs expressed in DL-Lite$_\mathcal{F}$ or DL-Lite$_\mathcal{A}$, we simply need to modify the computation of the conflicts of the knowledge base that currently only search for violation of disjointness TBox axioms in order to also find pairs of assertions that contradict functionality axioms (as well as datatype restrictions, in the case of DL-Lite$_\mathcal{A}$). Regarding complexity lower bounds, all hardness results of Section 5 hold for KBs expressed in DL-Lite$_{\mathsf{core}}$, hence for these four DL-Lite dialects. For Horn versions of DL-Lite, which allow the use of conjunctions in the concepts appearing in the left-hand side of TBox inclusions, the size of causes is not bounded by $|q|$ anymore but is bounded by $|q|*|\mathcal{T}|$, and the size of conflicts is bounded by $|\mathcal{T}|$, so conflicts and causes can still be computed in polynomial time w.r.t. data complexity. For AR (non)-answers, the extension of the SAT encoding presented in Remark 4.12 can be used, and the MUSes of $\varphi^1_{\neg q}$ w.r.t. $\varphi^2_{\neg q} \wedge \varphi_{cons}$ would allow us to generate the explanations for AR-answers, while the minimal models of $\varphi^1_{\neg q} \wedge \varphi^2_{\neg q} \wedge \varphi_{cons}$ (resp. of $\varphi^1_{\neg q} \wedge \varphi^2_{\neg q}$) can be connected to the explanations of negative AR (resp. IAR) answers. It is also possible to adapt the rewritings presented in the appendix to obtain the AC^0 upper bounds by taking into account the higher possible size of the conflicts. When moving to other DLs outside the DL-Lite family, query answering under the AR semantics, or even the IAR and brave semantics, becomes hard (Rosati, 2011). For

$\mathcal{ALC}$, which is the prototypical expressive description logic, query answering under these three semantics is in the second level of the polynomial hierarchy w.r.t. data complexity, while it is coNP-complete under classical semantics. Even for $\mathcal{EL}_\bot$, which extends the lightweight DL $\mathcal{EL}$ with the ability to express disjointness using $\bot$, query answering under the IAR and brave semantics is intractable w.r.t. data complexity, while query answering is in PTime under classical semantics. However, for $\mathcal{EL}_\bot$, query answering under AR semantics has the same complexity as in DL-Lite. It would be interesting to see if query answering under these semantics can be done efficiently for $\mathcal{EL}_\bot$, since the IAR semantics does not provide a tractable approximation of AR in this setting.

Regarding the performance of inconsistency-tolerant query answering, we see two possible directions to investigate. First, we could try to lower the cost of query evaluation. Indeed, our approach is based on the computation of the causes of the answers, which are obtained straightforwardly using the UCQ-rewritings of the initial queries. However, database management systems perform poorly on large UCQs, and it is not uncommon for UCQ-rewritings to be (very) large (there were more than 200,000 CQs in the rewritings of some queries of our benchmark!). That is why FO-rewritings that can be evaluated more efficiently in practice than the standard UCQs have been proposed (see e.g. Rosati & Almatelli, 2010; Eiter, Ortiz, Simkus, Tran, & Xiao, 2012; Thomazo, 2013; Bursztyn, Goasdoué, & Manolescu, 2015, 2016). To improve the performance of our system for queries that have a long evaluation time, we could try to use such optimized techniques to find the candidate answers, and then evaluate only the Boolean UCQ-rewritings instantiated with these answers to retrieve their causes. Indeed, such Boolean specializations would typically contain fewer variables and so would be easier to evaluate than the original UCQs. An open question is whether it is possible to compute the causes more directly, without employing UCQ-rewritings. Second, we could try alternative methods for AR query answering based on the pre-computation of the set of facts that hold under AR semantics that could provide some AR answers without using the SAT encoding, or by using other lower and upper approximations of AR semantics in place of the IAR and brave semantics.

There are several natural directions to explore related to explaining query (non-)answers. First, it would be useful to accompany our explanations with details on the TBox reasoning involved, using the work by Borgida et al. (2008) on proofs of positive answers as a starting point. The difficulty of such proofs could provide an additional criteria for ranking explanations (cf. the work on the cognitive complexity of justifications by Horridge et al., 2011). Second, our experiments showed that an answer can have a huge number of explanations, many of which are quite similar in structure. We thus plan to investigate ways of improving the presentation of explanations, e.g. by identifying and grouping similar explanations (cf. work on comparing justifications by Bail et al., 2013), or by defining a notion of representative explanation (Du et al., 2014). Third, we plan to experiment with other methods of generating explanations of negative answers, by comparing alternative encodings and using tools for computing hitting sets or diagnoses.

Acknowledgments

This work was supported by the ANR project PAGODA (ANR-12-JS02-007-01).

Appendix A. Omitted Proofs of Membership in AC^0

Thoughout this section, we will be working with a TBox $\mathcal{T}$, a CQ q, a candidate answer $\vec{a}$, and a UCQ-rewriting Q of $q(\vec{a})$ w.r.t. $\mathcal{T}$ and consistent ABoxes. We let $\Sigma = \Sigma_\mathsf{C} \cup \Sigma_\mathsf{R}$ denote the signature of $\mathcal{T}$ and q, with Σ_C (resp. Σ_R) is the set of concept names (resp. role names) occurring in $\mathcal{T}$ and/or q.

We begin with some technical preliminaries. Recall that $\mathcal{I}_\mathcal{A}$ is the interpretation isomorphic to ABox $\mathcal{A}$, i.e. having the individuals from $\mathcal{A}$ as its universe and interpreting all concept and role names so that an assertion holds iff it appears in $\mathcal{A}$. For any CQ Ψ, we can define a corresponding ABox $\mathcal{A}_\Psi$ consisting of the assertions obtained by replacing each variable v in Ψ by a fresh individual a_v, and then let $\mathcal{I}_\Psi$ be the interpretation isomorphic to $\mathcal{A}_\Psi$. Given a first-order formula Φ and an interpretation $\mathcal{I}$, the notation $\mathcal{I} \models_\pi \Phi$ will denote that Φ is satisfied in the interpretation $\mathcal{I}$ under a variable assignment π that maps every variable in Φ to an element of the universe of $\mathcal{I}$ (which is an individual if we consider an interpretation of the form $\mathcal{I}_\mathcal{A}$). We will use $\pi(\Phi)$ to denote the result of replacing each variable v in Φ by $\pi(v)$. Note that when Φ is a CQ, $\pi(\Phi)$ is a conjunction of assertions, which can be viewed as an ABox, continuing our convention of treating CQs either as conjunctive formulas or sets of atoms, depending on which is more convenient.

To establish the AC^0 membership results from Propositions 5.7, 5.15, and 5.23, we show how to construct FO-formulas that solve the recognition, relevance, and necessity tasks from these propositions. To render our constructions shorter and more readable, we will build these formulas in a modular way, with subformulas introduced for expressions that are used in multiple rewritings. For example, we will construct subformulas that check whether the image of a CQ defines a cause or whether it conflicts with all causes of $q(\vec{a})$. Our constructions are very loosely based upon the rewritings for k-support and k-defeater semantics by Bienvenu and Rosati (2013).

We start by defining some basic formulas that can be used to check whether an image of a CQ contains a certain assertion, is $\mathcal{T}$-(in)consistent, entails the query answer, or has a conflict in the ABox.

Lemma A.1. *Given a CQ Ψ and an atom α, one can construct FO-formulas* $\textsc{contains}(\Psi, \alpha)$, $\textsc{cons}(\Psi)$, $\textsc{incons}(\Psi)$, $\textsc{hasMatch}(\Psi)$, *and* $\textsc{hasConflict}(\Psi)$ *such that the following statements hold for every ABox $\mathcal{A}$ and variable assignment π with $\pi(\Psi) \subseteq \mathcal{A}$:*

1. $\mathcal{I}_\mathcal{A} \models_\pi \textsc{contains}(\Psi, \alpha)$ *iff* $\pi(\Psi)$ *contains* $\pi(\alpha)$
2. $\mathcal{I}_\mathcal{A} \models_\pi \textsc{incons}(\Psi)$ *iff* $\pi(\Psi)$ *is $\mathcal{T}$-inconsistent*
3. $\mathcal{I}_\mathcal{A} \models_\pi \textsc{cons}(\Psi)$ *iff* $\pi(\Psi)$ *is $\mathcal{T}$-consistent*
4. $\mathcal{I}_\mathcal{A} \models_\pi \textsc{hasMatch}(\Psi)$ *iff* $\langle \mathcal{T}, \pi(\Psi) \rangle \models q(\vec{a})$
5. $\mathcal{I}_\mathcal{A} \models_\pi \textsc{hasConflict}(\Psi)$ *iff* $\mathsf{confl}(\pi(\Psi), \langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$

Proof. For (1), we show how to define the formula $\textsc{contains}(\Psi, \alpha)$ when α is a role atom $R(t, t')$ (with t, t' terms). If $R(t_1, t'_1), \ldots, R(t_n, t'_n)$ is an enumeration of all of the R-atoms occurring in the CQ Ψ, then it suffices to set

$$\textsc{contains}(\Psi, R(t, t')) = \bigvee_{i=1}^{n} (t = t_i \wedge t' = t'_i)$$

The construction is analogous but simpler when α is a concept atom.

For the second statement, observe that some assignments may yield a consistent image while others lead to an inconsistent image. We therefore need to consider the different types of images that can be generated from Ψ by applying an assignment. To this end, we consider all possible equivalence relations $\equiv_1, \ldots, \equiv_M$ on the terms in Ψ such that no two individuals belong to the same equivalence class. Intuitively, an equivalence relation $\equiv_i$ represents a class of assignments, with $t_1 \equiv_i t_2$ meaning that t_1 and t_2 are mapped to the same individual. For each equivalence relation $\equiv_i$, the induced CQ Ψ'_i is obtained from Ψ by unifying all terms that occur in the same equivalence class of $\equiv_i$. The CQs Ψ'_i represent all possible images of Ψ under an assignment. Finally, we let EqUnsat be the set of all $\equiv_i$ such that $\mathcal{A}_{\Psi'_i}$ is $\mathcal{T}$-inconsistent, and we set

$$\textsc{incons}(\Psi) = \bigvee_{\equiv_i \in \textsc{EqUnsat}} \Big(\bigwedge_{t \equiv_i t'} t = t' \wedge \bigwedge_{t \not\equiv_i t'} t \neq t' \Big)$$

To see why this formula gives the desired result, consider an assignment π with $\pi(\Psi) \subseteq \mathcal{A}$. If $\mathcal{I}_\mathcal{A} \models_\pi \textsc{incons}(\Psi)$, then there must exist an equivalence relation $\equiv_i \in$ EqUnsat such that $\pi(t) = \pi(t')$ (resp. $\pi(t) \neq \pi(t')$) whenever $t \equiv_i t'$ (resp. $t \not\equiv_i t'$). It follows that $\pi(\Psi)$ is isomorphic to $\mathcal{A}_{\Psi'_i}$. As $\mathcal{A}_{\Psi'_i}$ is $\mathcal{T}$-inconsistent, and inconsistency is preserved under isomorphisms, $\pi(\Psi)$ is also $\mathcal{T}$-inconsistent. Conversely, suppose $\pi(\Psi)$ is $\mathcal{T}$-inconsistent, and let $\equiv_i$ be the equivalence relation on the terms of Ψ obtained by putting t, t' in the same equivalence class whenever $\pi(t) = \pi(t')$ (here we assume π maps individuals to themselves). By construction, $\mathcal{A}_{\Psi'_i}$ is isomorphic to $\pi(\Psi)$, so $\mathcal{A}_{\Psi'_i}$ is $\mathcal{T}$-inconsistent. It follows that $\equiv_i$ belongs to EqUnsat, so π satisfies the disjunct of $\textsc{incons}(\Psi)$ corresponding to $\equiv_i$, yielding $\mathcal{I}_\mathcal{A} \models_\pi \textsc{incons}(\Psi)$.

Statement (3) is an immediate corollary of (2), as we can set $\textsc{cons}(\Psi) = \neg\textsc{incons}(\Psi)$. For (4), the construction is broadly similar to that of (2). We consider all possible equivalence relations $\equiv_1, \ldots, \equiv_K$ over the set of terms in Ψ *and* the individuals occurring in $q(\vec{a})$, again with the restriction that each class can contain at most one individual. As before, each $\equiv_i$ induces a corresponding CQ Ψ'_i. We let EqMatch be the set of $\equiv_i$ such that $\langle \mathcal{T}, \mathcal{A}_{\Psi'_i} \rangle \models q(\vec{a})$. We then set

$$\textsc{hasMatch}(\Psi) = \bigvee_{\equiv_i \in \textsc{EqMatch}} \Big(\bigwedge_{t \equiv_i t'} t = t' \wedge \bigwedge_{t \not\equiv_i t'} t \neq t' \Big)$$

Correctness can be shown using a similar argument to that for (2). The only relevant difference is that since $q(\vec{a})$ contains individuals, we must argue that $\pi(\Psi)$ and $\mathcal{A}_{\Psi'_i}$ are isomorphic w.r.t. an isomorphism that is the identity on the individuals in $q(\vec{a})$ (this is why we consider equivalence relations involving the individuals in $q(\vec{a})$).

Finally, let us prove statement (5). Unlike the previous formulas, which only contained (in)equality atoms (and whose satisfaction depended only on π rather than the contents of the ABox), to check whether an image has a conflict in the ABox, we will need to verify whether certain assertions occur in the ABox. To define $\textsc{hasConflict}(\Psi)$, let $q^\mathcal{T}_{unsat} = \Gamma_1 \vee \ldots \vee \Gamma_L$ be the UCQ-rewriting of unsatisfiability constructed in the manner described in Section 2. We recall that each CQ Γ_i in $q^\mathcal{T}_{unsat}$ contains at most 2 atoms, and by repeating

atoms, we may assume they all contain precisely 2 atoms, so $\Gamma_i = \exists \vec{z}\, \gamma_i^1 \wedge \gamma_i^2$. We may also assume for convenience that every Γ_i has the same initial existential quantifier: $\exists \vec{z}$. The desired formula can then be defined as follows:

$$\textsc{hasConflict}(\Psi) = \bigvee_{i=1}^{L} \bigvee_{j=1}^{2} \exists \vec{z} \Big(\gamma_i^1 \wedge \gamma_i^2 \wedge \textsc{contains}(\Psi, \gamma_i^j) \wedge \textsc{contains}(\gamma_i^1, \gamma_i^2) \vee (\textsc{cons}(\gamma_i^1) \wedge \textsc{cons}(\gamma_i^2)) \Big)$$

The first line of the formula holds if for some i, $\pi(\Gamma_i)$ is present in the ABox and one of its assertions occurs in $\pi(\Psi)$. As Γ_i is a disjunct of $q^{\mathcal{T}}_{unsat}$, we know that $\pi(\Gamma_i)$ is $\mathcal{T}$-inconsistent. The second line ensures that $\pi(\Gamma_i)$ is a conflict, which trivially holds if it consists of a single assertion (which we can test using $\textsc{contains}(\gamma_i^1, \gamma_i^2)$) or if each of the component assertions is $\mathcal{T}$-consistent (here we can use the $\textsc{cons}$ formula from item (2)). Correctness follows from the definition of conflicts and the properties given in items (1) and (2). □

Next, we use the preceding building blocks to construct a formula that identifies causes:

$$\textsc{isCause}(\Psi) = \Psi \wedge \textsc{cons}(\Psi) \wedge \textsc{hasMatch}(\Psi) \wedge \bigwedge_{\gamma \in \Psi} (\textsc{contains}(\Psi \setminus \{\gamma\}, \gamma) \vee \neg \textsc{hasMatch}(\Psi \setminus \{\gamma\}))$$

This formula closely follows the definition of causes: it checks that the assertions in (the image of) Ψ are present in the ABox, that they are $\mathcal{T}$-consistent and entail the query answer, and that the query answer cannot be obtained from any proper subset. To verify the latter, for each atom in Ψ, we require that either removing γ does not change the image of the CQ (which can be the case when some variables are mapped to the same individuals) or the removal of the atom leads to the query answer no longer holding. The following lemma, which is a straightforward consequence of the definition of causes and items (2) and (4) of Lemma A.1, states that the formula has the intended meaning:

Lemma A.2. *For every ABox $\mathcal{A}$ and variable assignment π: $\mathcal{I}_\mathcal{A} \models_\pi \textsc{isCause}(\Psi)$ iff $\pi(\Psi)$ is a cause of $q(\vec{a})$ in the KB $\langle \mathcal{T}, \mathcal{A} \rangle$.*

For explanations of negative AR- and IAR-answers, we will need to be able to check whether a set of assertions conflicts all the causes of $q(\vec{a})$. To do so, we introduce the following formula, where E is the set of assertions we wish to test:

$$\textsc{conflictsAllCauses}(E) = \bigwedge_{\Psi(\vec{z}) \in Q} \forall \vec{z}\, (\textsc{isCause}(\Psi(\vec{z})) \rightarrow \bigvee_{\alpha \in E, \langle \mathcal{T}, \{\alpha\} \rangle \not\models \bot} \textsc{incons}(\Psi(\vec{z}) \wedge \alpha))$$

The preceding formula considers every CQ $\Psi(\vec{z})$ in the UCQ-rewriting of $q(\vec{a})$ and requires that for every possible instantiation of $\vec{z}$, if the resulting set of assertions is a cause, then it is conflicted by some $\mathcal{T}$-consistent assertion in the set E. Correctness is given in the following lemma.

Lemma A.3. *For every ABox $\mathcal{A}$, subset $E \subseteq \mathcal{A}$, and variable assignment π: $\mathcal{I}_\mathcal{A} \models_\pi \textsc{conflictsAllCauses}(E)$ iff for every $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \langle \mathcal{T}, \mathcal{A} \rangle)$, there is $\alpha \in E$ such that $\{\alpha\}$ is $\mathcal{T}$-consistent and $\mathcal{C} \cup \{\alpha\}$ is $\mathcal{T}$-inconsistent.*

Proof. First suppose $\mathcal{I}_\mathcal{A} \models_\pi \textsc{conflictsAllCauses}(E)$, and let $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \langle \mathcal{T}, \mathcal{A}\rangle)$. By Proposition 4.7, there exists $\Psi(\vec{z}) \in Q$ such that $\mathcal{C}$ is the image of $\Psi(\vec{z})$ under some assignment of the variables $\vec{z}$. From $\mathcal{I}_\mathcal{A} \models_\pi \textsc{conflictsAllCauses}(E)$, it follows that there exists a $\mathcal{T}$-consistent assertion $\alpha \in E$ such that $\mathcal{C} \cup \{\alpha\}$ is $\mathcal{T}$-inconsistent.

For the other direction, suppose that for every $\mathcal{C} \in \mathsf{causes}(q(\vec{a}), \langle \mathcal{T}, \mathcal{A}\rangle)$, there is $\alpha_\mathcal{C} \in E$ such that $\{\alpha_\mathcal{C}\}$ is $\mathcal{T}$-consistent and $\mathcal{C} \cup \{\alpha_\mathcal{C}\}$ is $\mathcal{T}$-inconsistent. Consider some $\Psi(\vec{z}) \in Q$ and some assignment ρ of the variables $\vec{z}$, and set $S = \rho(\Psi(\vec{z}))$. If S is not a cause, then the implication for $\Psi(\vec{z})$ and ρ is trivially satisfied. If $S \in \mathsf{causes}(q(\vec{a}), \langle \mathcal{T}, \mathcal{A}\rangle)$, then we can use the assertion α_S to show that $\bigvee_{\alpha \in E, \langle \mathcal{T}, \{\alpha\}\rangle \not\models \bot} \textsc{incons}(\Psi(\vec{z}) \wedge \alpha)$ is satisfied. □

We will also require a formula that checks whether the set of conflicts associated with one cause is contained in the set of conflicts of another cause. To this end, given two CQs Ψ and Ψ', we define a formula $\textsc{compareConflicts}(\Psi, \Psi')$ as follows:

$$\bigwedge_{A \in \Sigma_\mathsf{C}} \forall v \Big((A(v) \wedge \textsc{cons}(A(v)) \wedge \textsc{incons}(\Psi'(\vec{z}') \wedge A(v))) \rightarrow \textsc{incons}(\Psi(\vec{z}) \wedge A(v)) \Big) \wedge$$

$$\bigwedge_{R \in \Sigma_\mathsf{C}} \forall v,v' \Big((R(v,v') \wedge \textsc{cons}(R(v,v')) \wedge \textsc{incons}(\Psi'(\vec{z}') \wedge R(v,v'))) \rightarrow \textsc{incons}(\Psi(\vec{z}) \wedge R(v,v')) \Big)$$

The first line amounts to verifying that every $\mathcal{T}$-consistent concept assertion $A(a)$ in the ABox that conflicts the image of the second CQ also conflicts the image of the first CQ. The second line is similar, but formulated for role assertions. The next lemma formally states the properties of this formula:

Lemma A.4. *Let Ψ and Ψ' be CQs. For every ABox $\mathcal{A}$ and variable assignment π such that $\pi(\Psi)$ and $\pi(\Psi')$ are $\mathcal{T}$-consistent subsets of $\mathcal{A}$: $\mathcal{I}_\mathcal{A} \models_\pi \textsc{compareConflicts}(\Psi, \Psi')$ iff $\mathsf{confl}(\pi(\Psi'), \langle \mathcal{T}, \mathcal{A}\rangle) \subseteq \mathsf{confl}(\pi(\Psi), \langle \mathcal{T}, \mathcal{A}\rangle)$.*

Proof. Fix an ABox $\mathcal{A}$ and variable assignment π such that $\pi(\Psi)$ and $\pi(\Psi')$ are $\mathcal{T}$-consistent subsets of $\mathcal{A}$. First suppose $\mathcal{I}_\mathcal{A} \not\models_\pi \textsc{compareConflicts}(\Psi, \Psi')$. Then there exists $A \in \Sigma_\mathsf{C}$ or $R \in \Sigma_\mathsf{R}$ such that the corresponding subformula is not satisfied. Assume this is the case for $R \in \Sigma_\mathsf{R}$ (the argument is analogous for the case of $A \in \Sigma_\mathsf{C}$). Then there is some $R(a,b) \in \mathcal{A}$ such that $\{R(a,b)\}$ is $\mathcal{T}$-consistent, $\pi(\Psi'(\vec{z}')) \cup \{R(a,b)\}$ is $\mathcal{T}$-inconsistent, and $\pi(\Psi(\vec{z}')) \cup \{R(a,b)\}$ is $\mathcal{T}$-consistent. Thus, $R(a,b)$ is a conflict for $\pi(\Psi')$ but not for $\pi(\Psi)$, proving $\mathsf{confl}(\pi(\Psi'), \langle \mathcal{T}, \mathcal{A}\rangle) \not\subseteq \mathsf{confl}(\pi(\Psi), \langle \mathcal{T}, \mathcal{A}\rangle)$.

For the second direction, suppose $\mathsf{confl}(\pi(\Psi'), \langle \mathcal{T}, \mathcal{A}\rangle) \not\subseteq \mathsf{confl}(\pi(\Psi), \langle \mathcal{T}, \mathcal{A}\rangle)$, as witnessed by the assertion $R(a,b)$ which conflicts with $\pi(\Psi')$ but not for $\pi(\Psi)$ (the argument is analogous if $A(a)$ is the witness assertion). It follows that the subformula for role R and assignment π to the variables $\vec{z}$ is not satisfied in $\mathcal{I}_\mathcal{A}$, so $\mathcal{I}_\mathcal{A} \not\models_\pi \textsc{compareConflicts}(\Psi, \Psi')$. □

With these building blocks in hand, we can now proceed to the definition of the desired formulas for testing recognition, relevance, and necessity.

Positive brave-answers As the explanations for positive brave-answers are causes, we can use the formula for identifying causes to recognize positive brave-answers:

$$Q^{\text{REC}}_{\mathsf{brave}}(E) = \textsc{isCause}(E)$$

For relevance, we know from Proposition 4.7 that every cause is an image of a CQ appearing as a disjunct in the rewriting Q. We can thus take a disjunction over all CQs $\Psi(\vec{z})$ in Q and test whether there is some assignment to the variables $\vec{z}$ that yields an image that is a cause containing the considered assertion:

$$Q^{\text{REL}}_{\mathsf{brave}}(\alpha) = \bigvee_{\Psi(\vec{z}) \in Q} \exists \vec{z}\, (\textsc{isCause}(\Psi(\vec{z})) \wedge \textsc{contains}(\Psi(\vec{z}), \alpha))$$

A similar idea can be used for the necessity problem. An assertion α is necessary if there is no cause that omits α. Thus, we take a conjunction over all CQs $\Psi(\vec{z})$ in Q and check that for every assignment, if the resulting set of assertions is a cause, then it contains α.

$$Q^{\text{NEC}}_{\mathsf{brave}}(\alpha) = \bigwedge_{\Psi(\vec{z}) \in Q} \forall \vec{z}\, (\textsc{isCause}(\Psi(\vec{z})) \rightarrow \textsc{contains}(\Psi(\vec{z}), \alpha))$$

Positive IAR-answers A first-order formula for recognizing positive IAR-answers is easily obtained by combining the earlier formulas for identifying causes and determining whether a set of assertions has a conflict:

$$Q^{\text{REC}}_{\mathsf{IAR}}(E) = \textsc{isCause}(E) \wedge \neg \textsc{hasConflict}(E)$$

For relevance, we proceed as for the brave semantics: take a disjunction over all CQs in Q and check whether one such CQ has an image that is a cause that is without conflicts. This yields the following formula:

$$Q^{\text{REL}}_{\mathsf{IAR}}(\alpha) = \bigvee_{\Psi(\vec{z}) \in Q} \exists \vec{z}\, (\textsc{isCause}(\Psi(\vec{z})) \wedge \textsc{contains}(\Psi(\vec{z}), \alpha) \wedge \neg \textsc{hasConflict}(\Psi(\vec{z})))$$

For necessity, we can again follow the brave case, considering all CQs in the rewriting Q and requiring that every image that is a cause and is without conflicts contains α:

$$Q^{\text{NEC}}_{\mathsf{IAR}}(\alpha) = \bigwedge_{\Psi(\vec{z}) \in Q} \forall \vec{z}\, (\textsc{isCause}(\Psi(\vec{z})) \rightarrow \textsc{hasConflict}(\Psi(\vec{z})) \vee \textsc{contains}(\Psi(\vec{z}), \alpha))$$

Negative AR-answers For negative AR-answers, we consider only the recognition task, as relevance and necessity were proven intractable. The following formula expresses the conditions for a set of assertions to be an explanation for a negative AR-answer:

$$Q^{\text{REC}}_{\neg\mathsf{AR}}(E) = E \wedge \textsc{cons}(E) \wedge \textsc{conflictsAllCauses}(E)$$

Correctness immediately follows from Lemmas A.2 and A.3.

Negative IAR-answers To recognize explanations for negative IAR-answers, we can simply drop the consistency requirement from the formula for the AR case:

$$Q^{\text{REC}}_{\neg\mathsf{IAR}}(E) = E \wedge \textsc{conflictsAllCauses}(E)$$

For the relevance problem, we exploit the characterization given in Proposition 5.22, which states that an assertion α is relevant iff there exists a cause $\mathcal{C}$ such that α conflicts $\mathcal{C}$, and for every other cause $\mathcal{C}'$, if $\mathsf{confl}(\mathcal{C}', \langle \mathcal{T}, \mathcal{A}\rangle) \subseteq \mathsf{confl}(\mathcal{C}, \langle \mathcal{T}, \mathcal{A}\rangle)$, then $\alpha \in \mathsf{confl}(\mathcal{C}', \mathcal{K})$. We translate this condition into first-order logic as follows:

$$\begin{aligned} Q^{\text{REL}}_{\neg\mathsf{IAR}}(\alpha) = \bigvee_{\Psi(\vec{z})\in Q} \exists \vec{z} \Big(& \textsc{isCause}(\Psi(\vec{z})) \wedge \textsc{incons}(\Psi(\vec{z}) \wedge \alpha) \\ & \wedge \bigwedge_{\Psi'(\vec{z'})\in Q} \forall \vec{z'} \Big((\textsc{isCause}(\Psi'(\vec{z'})) \wedge \textsc{compareConflicts}(\Psi, \Psi')) \\ & \qquad \rightarrow \textsc{incons}(\Psi'(\vec{z'}) \wedge \alpha) \Big) \Big) \end{aligned}$$

In the first line, we take a disjunction over all CQs in Q (representing the choices for the shape of the first cause) and check whether there is some assignment such that the resulting set of assertions, call it $\mathcal{C}$, is a cause satisfying the preceding requirements. We may assume that $\{\alpha\}$ is $\mathcal{T}$-consistent, since otherwise α cannot be relevant (so we could simply set $Q^{\text{REL}}_{\neg\mathsf{IAR}}(\alpha) = \bot$). This, together with the fact that $\mathcal{C}$ is a cause (hence $\mathcal{T}$-consistent), means that α conflicts with $\mathcal{C}$ iff $\mathcal{C} \cup \{\alpha\}$ is $\mathcal{T}$-inconsistent, which is checked in the first line of the formula. In the second and third lines, we verify that every cause $\mathcal{C}'$ satisfied the aforementioned conditions. To do so, we take the conjunction over all CQs in Q (in order to capture all possible causes), and ask that for every assignment that yields a cause whose set of conflicts is contained in the conflicts of $\mathcal{C}$ has the assertion α as a conflict. Correctness follows from Proposition 5.22, Lemmas A.1, A.2 and A.4.

For the necessity task, we make use of the characterization given in Proposition 5.21, which states that an assertion α appears in all explanations of a negative IAR-answer iff there exists a cause for that answer that is only conflicted by α. We thus create a formula that checks for the existence of a cause that is conflict with the assertion α and such that every other assertion that is not equal to α (either because it uses a different concept or role name, or because it uses different individuals), is consistent with the cause. The formulation is slightly different depending on whether α is a concept or role assertion. We give here the formulation for the case of role assertions:

$$\begin{aligned} Q^{\text{NEC}}_{\neg\mathsf{IAR}}(R(a,b)) = \bigvee_{\Psi(\vec{z})\in Q} \exists \vec{z} \Big(& \textsc{isCause}(\Psi(\vec{z})) \wedge \neg\textsc{cons}(\Psi(\vec{z}) \cup \{R(a,b)\}) \\ & \wedge \forall v, v' \left((v = a \wedge v' = b) \vee \neg R(v,v') \vee \textsc{cons}(\Psi(\vec{z}) \cup \{R(v,v')\}) \right) \\ & \wedge \bigwedge_{P\in\Sigma_{\mathsf{R}}, P\neq R} \forall v, v' \left(\neg P(v,v') \vee \textsc{cons}(\Psi(\vec{z}) \cup \{P(v,v')\}) \right) \\ & \wedge \bigwedge_{B\in\Sigma_{\mathsf{C}}} \forall v \left(\neg B(v) \vee \textsc{cons}(\Psi(\vec{z}) \cup \{B(v)\}) \right) \Big) \end{aligned}$$

Note that v and v' are fresh variables not occurring in $\vec{z}$. Correctness follows from Proposition 5.21, together with Lemmas A.1 and A.2.

Appendix B. Tables and Figures from Sections 6 and 7

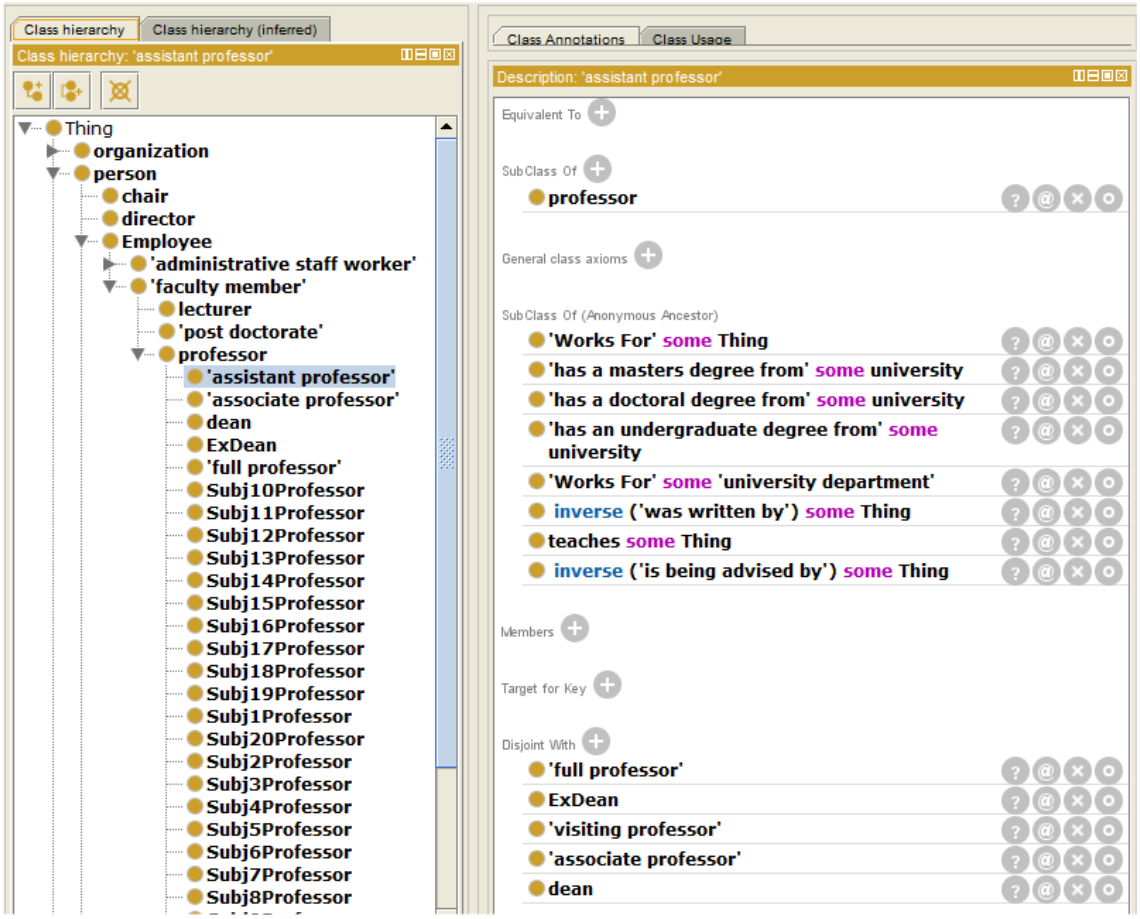


Figure 3: Screenshot from Protégé ontology editor (https://protege.stanford.edu/). The left side displays a part of the concept hierarchy of the $\text{LUBM}^{\exists}_{20}$ ontology, and the upper part of the right side shows of which concepts AssistantProfessor is a subconcept (here the single concept Professor). The lower part of the right side displays the negative inclusions added between AssistantProfessor and concepts with the same closest super-concept Professor: FullProfessor, ExDean, VisitingProfessor, AssociateProfessor and Dean. We did not add disjointness axioms with the concepts SubjXProfessor, because such concepts indicate the domain of a professor, which is independent from its seniority.

ABox id	size	% assertions added to uXc0	# assertions in some conflict	% assertions in some conflict
u1c0	75,663	0	0	0
u1c1	75,724	0.08	2,373	3
u1c5	75,951	0.38	6,412	8
u1c10	76,201	0.70	10,891	14
u1c20	76,821	1.51	20,175	26
u1c30	77,447	2.30	26,086	34
u1c50	78,593	3.73	34,814	44
u5c0	463,325	0	0	0
u5c1	463,691	0.08	12,191	3
u5c5	465,157	0.39	45,906	10
u5c10	466,919	0.77	83,263	18
u5c20	470,674	1.56	137,836	29
u5c30	474,368	2.33	172,245	36
u5c50	481,400	3.75	221,900	46
u20c0	1,981,872	0	0	0
u20c1	1,983,493	0.08	69,597	4
u20c5	1,989,788	0.40	253,141	13
u20c10	1,997,445	0.78	408,398	20
u20c20	2,013,048	1.55	610,271	30
u20c30	2,028,069	2.28	748,664	37
u20c50	2,056,957	3.65	946,819	46
u50c0	4,934,691	0	0	0
u50c1	4,938,737	0.08	224,131	5
u50c5	4,954,494	0.40	686,159	14
u50c10	4,973,292	0.78	1,034,226	21
u50c20	5,010,776	1.52	1,517,499	30
u50c30	5,046,802	2.22	1,865,679	37
u50c50	5,115,473	3.53	2,353,739	46
u100c0	9,938,139	0	0	0
u100c1	9,946,144	0.08	546,708	5
u100c5	9,977,656	0.40	1,381,298	14
u100c10	10,014,894	0.77	2,077,201	21
u100c20	10,087,801	1.48	3,069,321	30
u100c30	10,157,192	2.16	3,755,732	37
u100c50	10,289,863	3.42	4,728,588	46

Table 5: Characteristics of ABoxes used in experiments.

$$
\begin{aligned}
\mathsf{q1}(x,y) =& \mathsf{Person}(x) \wedge \mathsf{takesCourse}(x,y)\\
\mathsf{q2}(x,y) =& \mathsf{Employee}(x) \wedge \mathsf{publicationAuthor}(y,x)\\
\mathsf{q3}(x,y) =& \mathsf{Professor}(x) \wedge \mathsf{teacherOf}(x,y) \wedge \mathsf{worksFor}(x, Dept0.Univ0)\\
\mathsf{q4}(x,y,z,u,v,w) =& \mathsf{FullProfessor}(x) \wedge \mathsf{publiAuthor}(y,x) \wedge \mathsf{teacherOf}(x,z) \wedge \mathsf{advisor}(u,x) \wedge\\
& \mathsf{graduateStudent}(u) \wedge \mathsf{degreeFrom}(x,v) \wedge \mathsf{degreeFrom}(u,w)\\
\mathsf{q5}(x) =& \exists y \mathsf{Person}(Dept2.Univ0/GradStudent131) \wedge\\
& \mathsf{takesCourse}(Dept2.Univ0/GradStudent131, y) \wedge\\
& \mathsf{GraduateCourse}(y) \wedge \mathsf{takesCourse}(x,y) \wedge \mathsf{Person}(x)\\
\mathsf{q6}(x,z) =& \exists y \mathsf{Employee}(x) \wedge \mathsf{publicationAuthor}(y,x) \wedge \mathsf{memberOf}(x, Dept4.Univ0) \wedge\\
& \mathsf{Employee}(z) \wedge \mathsf{publicationAuthor}(y,z) \wedge \mathsf{memberOf}(z, Dept4.Univ0)\\
\mathsf{q7}(y) =& \exists x \mathsf{Employee}(x) \wedge \mathsf{memberOf}(x, Dept2.Univ0) \wedge \mathsf{degreeFrom}(x,y)\\
\mathsf{q8}(x) =& \exists y \mathsf{teacherOf}(x,y) \wedge \mathsf{degreeFrom}(x, Univ532)\\
\mathsf{q9}(x,z) =& \exists y u \mathsf{Employee}(x) \wedge \mathsf{memberOf}(x,u) \wedge \mathsf{degreeFrom}(x,y) \wedge\\
& \mathsf{Employee}(z) \wedge \mathsf{memberOf}(z,u) \wedge \mathsf{degreeFrom}(z,y)\\
\mathsf{q10}(x,z) =& \exists u \mathsf{Employee}(x) \wedge \mathsf{memberOf}(x,u) \wedge \mathsf{degreeFrom}(x, Univ532) \wedge\\
& \mathsf{Employee}(z) \wedge \mathsf{memberOf}(z,u) \wedge \mathsf{degreeFrom}(z, Univ532)\\
\mathsf{q11}(x) =& \exists y \mathsf{Faculty}(x) \wedge \mathsf{publicationAuthor}(y,x)\\
\mathsf{q12}(x) =& \mathsf{Organization}(x)\\
\mathsf{q13}(x) =& \mathsf{Employee}(x)\\
\mathsf{q14}(x,y,z) =& \mathsf{Student}(x) \wedge \mathsf{advisor}(x,y) \wedge \mathsf{Faculty}(y) \wedge \mathsf{takesCourse}(x,z) \wedge\\
& \mathsf{teacherOf}(y,z) \wedge \mathsf{Course}(z)\\
\mathsf{q15}(x,y) =& \mathsf{Person}(x) \wedge \mathsf{worksFor}(x,y) \wedge \mathsf{Organization}(y)\\
\mathsf{q16}(x,y) =& \exists z u \mathsf{Student}(x) \wedge \mathsf{takesCourse}(x,y) \wedge \mathsf{Course}(y) \wedge \mathsf{teacherOf}(z,y) \wedge\\
& \mathsf{Faculty}(z) \wedge \mathsf{worksFor}(z,u) \wedge \mathsf{Department}(u) \wedge \mathsf{memberOf}(x,u)\\
\mathsf{q17}(x) =& \exists y z \mathsf{Faculty}(x) \wedge \mathsf{degreeFrom}(x,y) \wedge \mathsf{University}(y) \wedge\\
& \mathsf{subOrganizationOf}(z,y) \wedge \mathsf{Department}(z) \wedge \mathsf{memberOf}(x,z)\\
\mathsf{q18}(x) =& \exists y z \mathsf{Publication}(x) \wedge \mathsf{publicationAuthor}(x,y) \wedge \mathsf{Professor}(y) \wedge\\
& \mathsf{publicationAuthor}(x,z) \wedge \mathsf{Student}(z)\\
\mathsf{q19}(x,y) =& \exists z u \mathsf{University}(x) \wedge \mathsf{University}(y) \wedge \mathsf{memberOf}(z,x) \wedge \mathsf{Student}(z) \wedge\\
& \mathsf{University}(y) \wedge \mathsf{memberOf}(u,y) \wedge \mathsf{Professor}(u) \wedge \mathsf{advisor}(z,u)\\
\mathsf{q20}(x,y) =& \mathsf{takesCourse}(x,y) \wedge \mathsf{Student}(x) \wedge \mathsf{Course}(y) \wedge\\
& \mathsf{teacherOf}(Dept0.Univ0/AssociateProf0, y)
\end{aligned}
$$

Figure 4: Queries used in experiments.

	Sure	uXc1 Likely	Possible	Sure	uXc20 Likely	Possible	Sure	uXc50 Likely	Possible
u1cY									
q1	20029	0	380	12538	7	7864	6646	19	13747
q2	7215	20	12	6284	402	734	4728	887	2087
q3	85	0	0	0	0	85	0	0	87
q4	78101	0	5636	24545	0	60236	4806	0	80839
q5	10	0	0	0	10	0	0	0	10
q6	235	0	0	177	14	110	0	0	342
q7	136	0	1	0	0	138	0	0	149
q8	0	0	0	0	0	0	0	0	0
q9	1291	3	80	1002	68	406	783	116	741
q10	0	0	3	0	0	6	0	0	7
q11	534	0	4	471	4	89	385	7	236
q12	1180	11	10	999	174	117	802	345	350
q13	1069	3	8	966	71	122	783	169	351
q14	191	0	4	98	0	97	36	0	159
q15	405	0	102	99	0	409	0	0	515
q16	13545	0	3987	2052	0	15480	0	0	17532
q17	0	0	1	0	0	1	0	0	1
q18	3107	0	66	2302	0	872	1319	0	1871
q19	0	0	0	0	0	0	0	0	0
q20	50	0	0	25	0	25	0	0	50
u20cY									
q1	532331	0	12776	344391	190	200539	187504	758	356922
q2	189186	228	1019	163260	9927	22489	127098	23819	52117
q3	85	0	0	0	0	85	0	0	87
q4	1123422	0	944364	118	0	2082472	0	0	2114007
q5	10	0	0	0	10	0	0	0	10
q6	235	0	0	177	14	110	0	0	342
q7	91	0	46	0	0	138	0	0	149
q8	0	0	31	0	0	31	0	0	32
q9	33433	60	2714	25701	59	13282	21462	267	21419
q10	0	0	58	0	0	62	0	0	66
q11	14331	0	145	12613	42	2781	10329	267	6373
q12	7082	218	251	5830	880	3881	5395	991	8769
q13	28891	64	204	25791	1780	4028	21471	4185	9430
q14	4785	0	166	2539	0	2412	1007	0	3944
q15	12050	0	1702	1715	0	12143	54	0	13946
q16	396411	0	72138	33936	0	434613	585	0	467964
q17	27	0	10	0	0	37	0	0	39
q18	81760	0	1342	58294	0	24959	34795	0	48770
q19	0	0	1	0	0	8	0	0	20
q20	50	0	0	25	0	25	0	0	50
u100cY									
q1	2675887	3	65067	1732258	961	1007794	934012	3751	1803546
q2	946599	1003	5807	819168	50118	109934	637443	124113	254737
q3	85	0	0	0	0	85	0	0	87
q4	702009	0	9614733	OOM	OOM	OOM	OOM	OOM	OOM
q5	10	0	0	0	10	0	0	0	10
q6	235	0	0	177	14	110	0	0	342
q7	34	0	103	0	0	138	0	0	149
q8	0	0	187	0	0	188	0	0	190
q9	152404	110	27107	128616	192	64820	107220	1300	105450
q10	0	0	293	0	0	310	0	0	326
q11	71756	0	739	63411	192	13778	51791	1300	31764
q12	31955	566	1109	29074	1166	18162	27002	2849	41644
q13	144313	308	1014	129083	8902	19737	107258	21279	46553
q14	23330	0	777	12390	0	11717	4942	0	19165
q15	61189	0	7584	7693	0	61492	221	0	69599
q16	2029091	0	323720	150512	0	2202299	2374	0	2350437
q17	28	0	190	0	0	221	0	0	226
q18	406817	0	7330	291909	0	123116	174140	17	242252
q19	0	0	5	0	0	56	0	0	124
q20	50	0	0	25	0	25	0	0	50

Table 6: Number of answers per category for three sizes of ABoxes and three ratios of conflicts (about 4%, 30%, and 45% of assertions involved in some conflict).

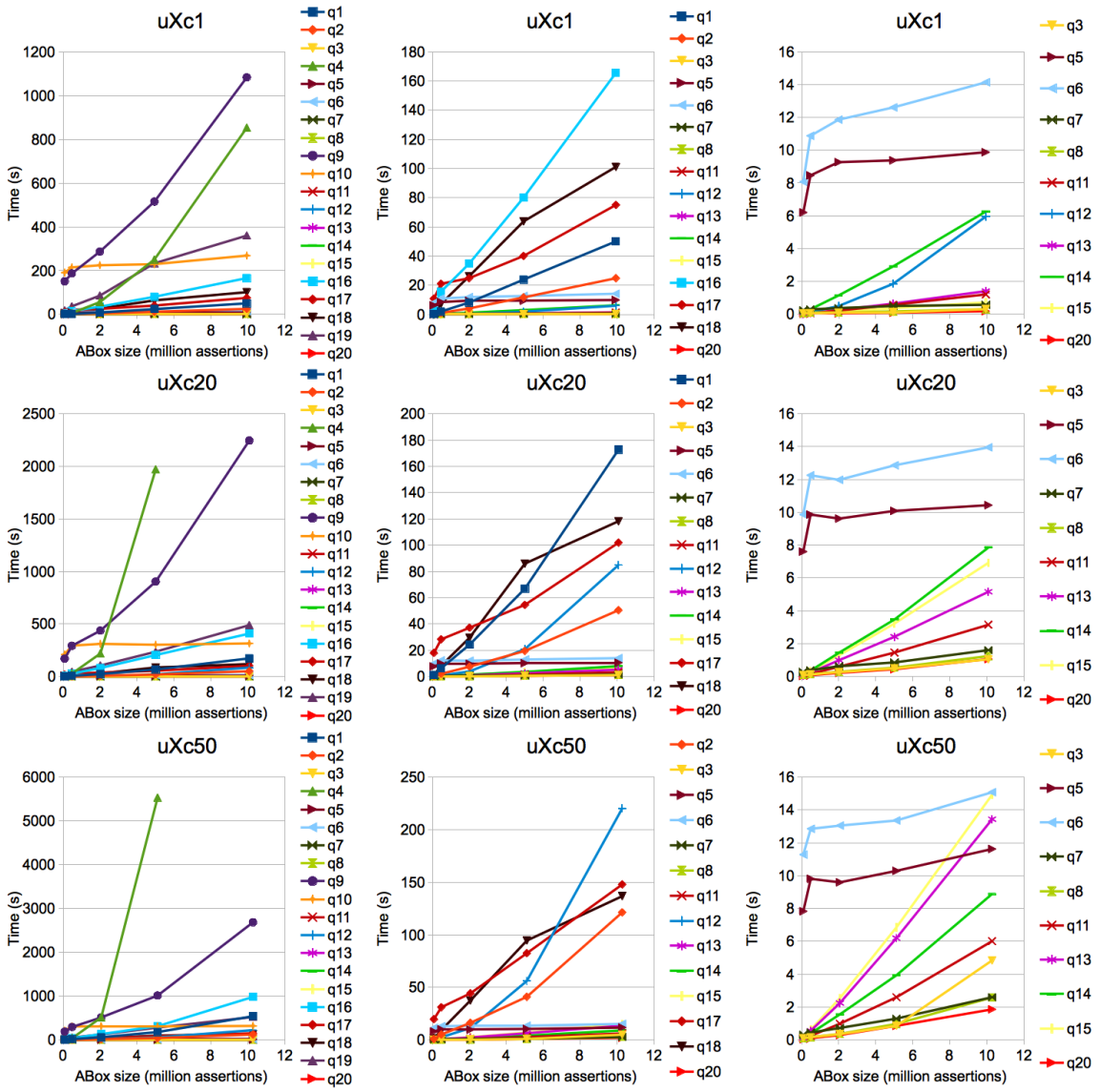


Figure 5: Time in seconds for query answering w.r.t. the size of the ABox for three ratios of conflicts (about 4%, 30%, and 45% of assertions in some conflict). For readability, the center and right columns focus on the queries whose answering times are lower and whose behaviours are thus not visible in the left column.

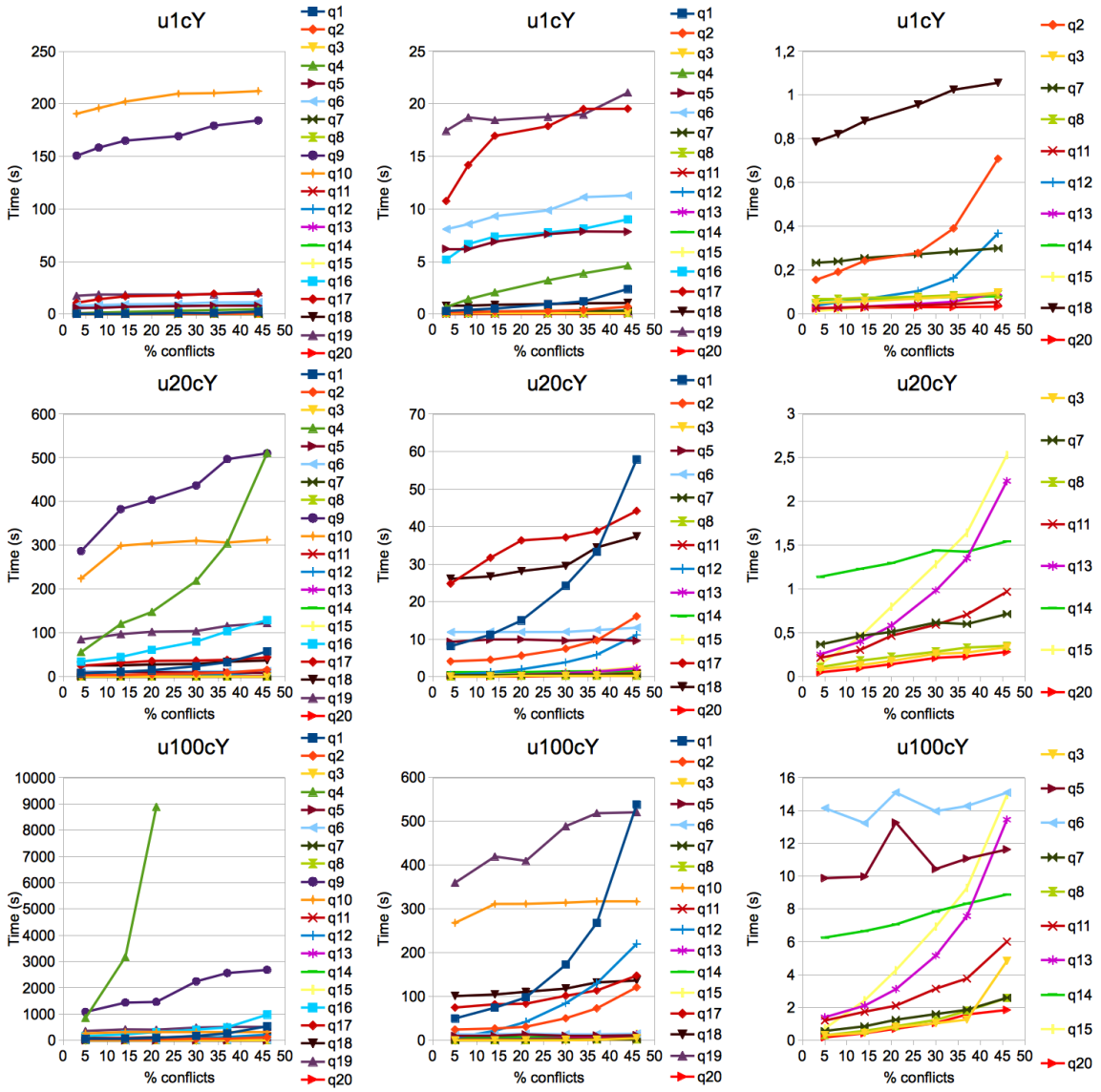


Figure 6: Time in seconds for query answering w.r.t. the ratios of conflicts for three ABox sizes (about 76 thousand, 2 million, and 10 million assertions). The center and right columns focus on the queries whose answering times are lower and whose behaviours are thus not visible in the left column.

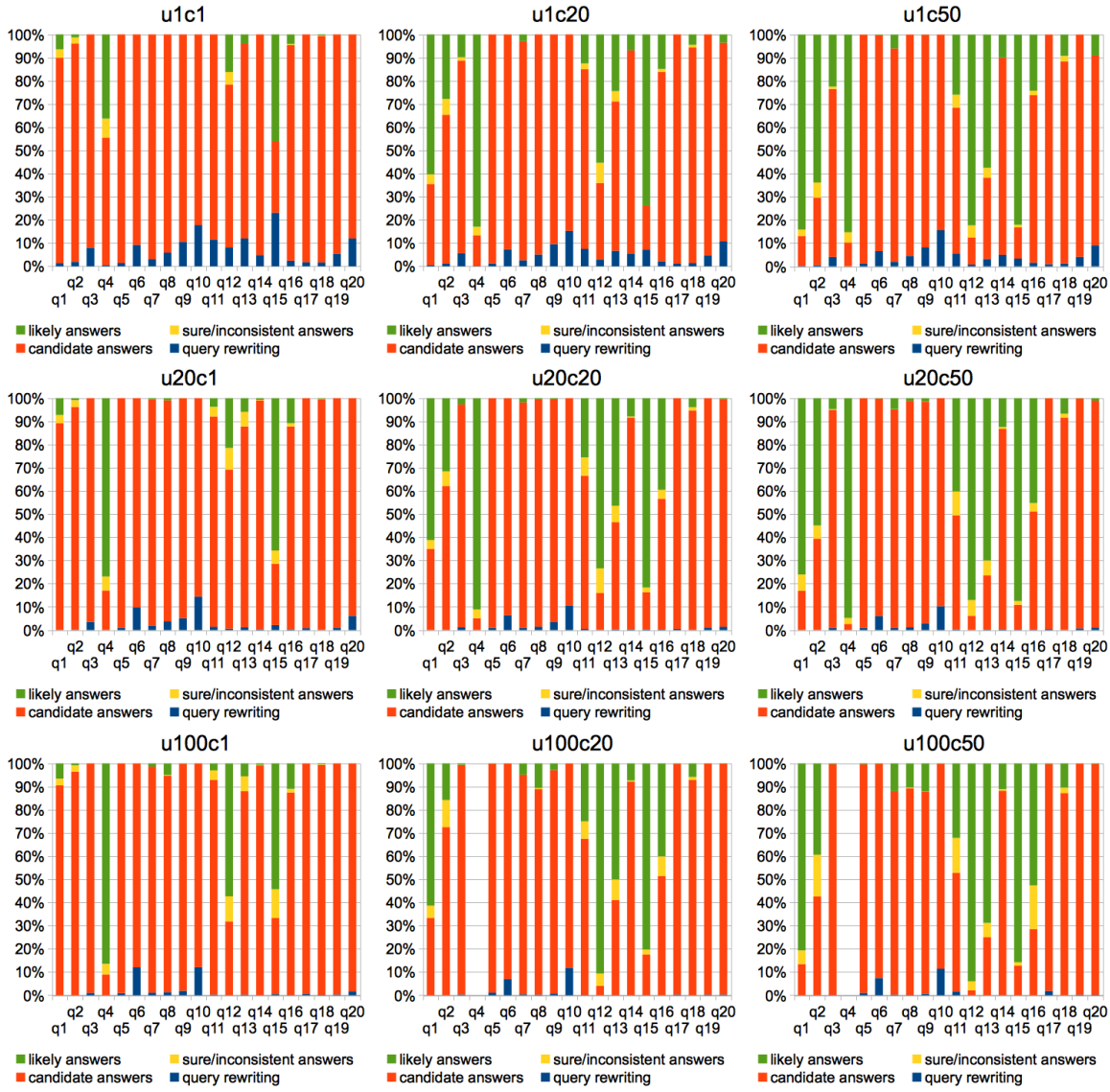


Figure 7: Proportion of time spent by CQAPri in the different phases of query answering on 9 ABoxes. The two lower bars are the time for rewriting the query and executing the rewritten query to get candidate answers, and the two upper bars represent the time needed to classify such answers, by identifying the IAR- and non-brave-answers in a first step ('sure/inconsistent' in the legend), then the AR-answers ('likely answers').

		u1c1					u1c20					u1c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q1	Sure	20029	100	0	0	0	12538	100	0	0	0	6646	100	0	0	0
	Likely	0					7	100	0	0	0	19	100	0	0	0
	Poss.	380	100	0	0	0	7864	100	0	0	0	13747	99.96	0.04	0	0
q2	Sure	7215	100	0	0	0	6284	100	0	0	0	4728	100	0	0	0
	Likely	20	100	0	0	0	402	99.75	0.25	0	0	887	99.89	0.11	0	0
	Poss.	12	100	0	0	0	734	100	0	0	0	2087	99.95	0.05	0	0
q3	Sure	85	100	0	0	0	0					0				
	Poss.	0					85	100	0	0	0	87	100	0	0	0
q4	Sure	78101	100	0	0	0	24545	100	0	0	0	4806	100	0	0	0
	Poss.	5636	99.96	0.04	0	0	60236	99.99	0.01	0	0	80839	99.98	0.01	<0.01	0
q5	Sure	10	100	0	0	0	0					0				
	Likely	0					10	60	40	0	0	0				
	Poss.	0					0					10	100	0	0	0
q6	Sure	235	100	0	0	0	177	100	0	0	0	0				
	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Sure	136	100	0	0	0	0					0				
	Poss.	1	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q9	Sure	1291	100	0	0	0	1002	99.99	0.01	0	0	783	100	0	0	0
	Likely	3	100	0	0	0	68	100	0	0	0	116	100	0	0	0
	Poss.	80	100	0	0	0	406	100	0	0	0	741	99.87	0.13	0	0
	Poss.	3	100	0	0	0	6	100	0	0	0	7	100	0	0	0
q11	Sure	534	100	0	0	0	471	100	0	0	0	385	100	0	0	0
	Likely	0					4	100	0	0	0	7	100	0	0	0
	Poss.	4	100	0	0	0	89	100	0	0	0	236	99.58	0.42	0	0
q12	Sure	1180	100	0	0	0	999	100	0	0	0	802	100	0	0	0
	Likely	11	100	0	0	0	174	100	0	0	0	345	100	0	0	0
	Poss.	10	40	20	40	0	117	69.23	4.27	26.50	0	350	73.43	0	26.57	0
q13	Sure	1069	100	0	0	0	966	100	0	0	0	783	100	0	0	0
	Likely	3	100	0	0	0	71	100	0	0	0	169	100	0	0	0
	Poss.	8	100	0	0	0	122	99.18	0	0.82	0	351	98.58	0.28	1.14	0
q14	Sure	191	100	0	0	0	98	100	0	0	0	36	100	0	0	0
	Poss.	4	100	0	0	0	97	100	0	0	0	159	100	0	0	0
q15	Sure	405	100	0	0	0	99	100	0	0	0	0				
	Poss.	102	100	0	0	0	409	99.76	0	0.24	0	515	99.03	0	0.97	0
q16	Sure	13545	99.99	0.01	0	0	2052	100	0	0	0	0				
	Poss.	3987	100	0	0	0	15480	99.99	0.01	0	0	17532	99.98	0.02	0	0
	Poss.	1	100	0	0	0	1	100	0	0	0	1	100	0	0	0
q18	Sure	3107	100	0	0	0	2302	100	0	0	0	1319	100	0	0	0
	Poss.	66	100	0	0	0	872	99.89	0.11	0	0	1871	100	0	0	0
q20	Sure	50	100	0	0	0	25	100	0	0	0	0				
	Poss.	0					25	100	0	0	0	50	100	0	0	0

Table 7: Number of answers of each class with distribution (in %) of their explanation times (in second) per query over ABoxes with 1 university and three different ratios of conflicts.

		u20c1					u20c20					u20c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q2	Sure	189186	>99.99	<0.01	0	0	163260	>99.99	<0.01	0	0	127098	>99.99	<0.01	0	0
	Likely	228	100	0	0	0	9927	99.98	0.02	0	0	23819	99.97	0.03	0	0
	Poss.	1019	100	0	0	0	22489	99.96	0.04	0	0	52117	99.95	0.05	0	0
q3	Sure	85	100	0	0	0	0					0				
	Poss.	0					85	100	0	0	0	87	100	0	0	0
q5	Sure	10	100	0	0	0	0					0				
	Likely	0					10	70	30	0	0	0				
	Poss.	0					0					10	90	10	0	0
q6	Sure	235	100	0	0	0	177	100	0	0	0	0				
	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Sure	91	100	0	0	0	0					0				
	Poss.	46	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q8	Poss.	31	100	0	0	0	31	100	0	0	0	32	100	0	0	0
q9	Sure	33433	>99.99	<0.01	0	0	25701	>99.99	<0.01	0	0	21462	99.96	0.04	0	0
	Likely	60	100	0	0	0	59	100	0	0	0	267	99.63	0.37	0	0
	Poss.	2714	100	0	0	0	13282	99.89	0.11	0	0	21419	93.01	6.99	0	0
q10	Poss.	58	100	0	0	0	62	100	0	0	0	66	100	0	0	0
q11	Sure	14331	100	0	0	0	12613	100	0	0	0	10329	100	0	0	0
	Likely	0					42	100	0	0	0	267	100	0	0	0
	Poss.	145	100	0	0	0	2781	99.89	0.11	0	0	6373	99.70	0.30	0	0
q12	Sure	7082	100	0	0	0	5830	100	0	0	0	5395	100	0	0	0
	Likely	218	11.93	88.07	0	0	880	25.68	74.32	0	0	991	58.83	40.06	1.11	0
	Poss.	251	47.41	23.90	28.69	0	3881	50.40	18.50	31.10	0	8769	54.29	11.68	34.03	0
q13	Sure	28891	100	0	0	0	25791	100	0	0	0	21471	100	0	0	0
	Likely	64	100	0	0	0	1780	100	0	0	0	4185	99.95	0.05	0	0
	Poss.	204	98.53	0	1.47	0	4028	98.01	0.05	1.94	0	9430	98.30	0.22	1.48	0
q14	Sure	4785	100	0	0	0	2539	100	0	0	0	1007	100	0	0	0
	Likely	0					0					0				
	Poss.	166	100	0	0	0	2412	100	0	0	0	3944	100	0	0	0
q15	Sure	12050	100	0	0	0	1715	100	0	0	0	54	100	0	0	0
	Poss.	1702	99.82	0	0.18	0	12143	99.28	0.02	0.70	0	13946	98.69	0.03	1.28	0
q17	Sure	27	100	0	0	0	0					0				
	Poss.	10	100	0	0	0	37	100	0	0	0	39	82.05	17.95	0	0
q18	Sure	81760	>99.99	<0.01	0	0	58294	99.99	0.01	0	0	34795	99.98	0.02	0	0
	Poss.	1342	100	0	0	0	24959	99.87	0.13	0	0	48770	99.91	0.09	0	0
q19	Poss.	1	0	100	0	0	8	0	87.50	12.50	0	20	0	45	55	0
q20	Sure	50	100	0	0	0	25	100	0	0	0	0				
	Poss.	0					25	100	0	0	0	50	100	0	0	0

Table 8: Number of answers of each class with distribution (in %) of their explanation times (in second) per query over ABoxes with 20 universities and three different ratios of conflicts.

		u100c1					u100c20					u100c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q3	Sure	85	100	0	0	0	0					0				
	Poss.	0					85	100	0	0	0	87	98.85	0	0	1.15
q5	Sure	10	100	0	0	0	0					0				
	Likely	0					10	50	50	0	0	0				
	Poss.	0					0					10	0	100	0	0
q6	Sure	235	100	0	0	0	177	100	0	0	0	0				
	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Sure	34	100	0	0	0	0					0				
	Poss.	103	100	0	0	0	138	99.28	0	0.72	0	149	78.52	20.81	0.67	0
q8	Poss.	187	100	0	0	0	188	100	0	0	0	190	10.53	88.95	0.52	0
q9	Sure	152404	99.99	0.01	0	0	128616	99.98	0.02	0	0	107220	99.99	0.01	0	0
	Likely	110	100	0	0	0	192	100	0	0	0	1300	70.46	29.54	0	0
	Poss.	27107	99.95	0.05	0	0	64820	86.54	13.46	0	0	105450	63.68	32.92	3.25	0.15
q10	Poss.	293	100	0	0	0	310	96.45	3.55	0	0	326	33.13	66.87	0	0
q11	Sure	71756	>99.99	<0.01	0	0	63411	>99.99	0	<0.01	0	51791	>99.99	<0.01	0	0
	Likely	0					192	100	0	0	0	1300	99.92	0.08	0	0
	Poss.	739	100	0	0	0	13778	99.92	0.06	0.02	0	31764	99.76	0.24	0	0
q12	Sure	31955	100	0	0	0	29074	>99.99	<0.01	0	0	27002	100	0	0	0
	Likely	566	0.53	0	99.47	0	1166	86.19	0	5.57	8.23	2849	99.16	0.03	0	0.81
	Poss.	1109	43.28	12.17	44.55	0	18162	52.16	2.31	38.46	7.07	41644	56.85	0.34	30.08	12.73
q13	Sure	144313	>99.99	<0.01	0	0	129083	>99.99	<0.01	0	0	107258	>99.99	<0.01	0	0
	Likely	308	100	0	0	0	8902	99.98	0.02	0	0	21279	99.99	0.01	0	0
	Poss.	1014	98.82	0	1.18	0	19737	98.56	0.06	1.38	0	46553	98.56	0.16	1.28	0
q14	Sure	23330	100	0	0	0	12390	100	0	0	0	4942	100	0	0	0
	Poss.	777	100	0	0	0	11717	100	0	0	0	19165	99.99	0	0	0.01
q15	Sure	61189	100	0	0	0	7693	100	0	0	0	221	100	0	0	0
	Poss.	7584	99.83	0.01	0.16	0	61492	99.52	0.01	0.47	0	69599	98.99	0.01	1.00	0
q17	Sure	28	100	0	0	0	0					0				
	Poss.	190	100	0	0	0	221	81.90	18.10	0	0	226	3.10	95.57	1.33	0
q19	Poss.	5	0	100	0	0	56	0	98.21	1.79	0	124	0	65.32	34.68	0
q20	Sure	50	100	0	0	0	25	100	0	0	0	0				
	Poss.	0					25	100	0	0	0	50	100	0	0	0

Table 9: Number of answers of each class with distribution (in %) of their explanation times (in second) per query over ABoxes with 100 universities and three different ratios of conflicts.

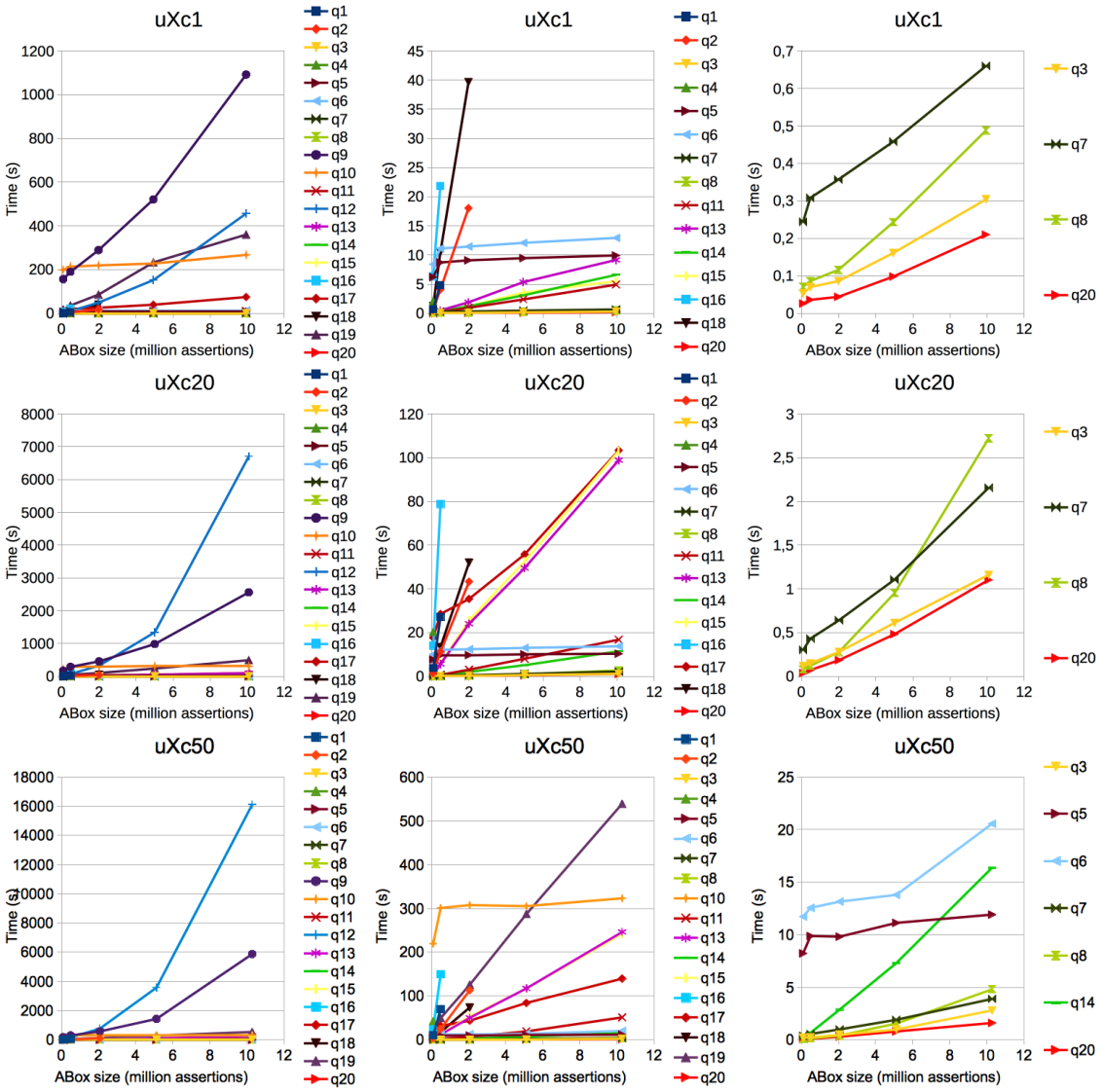


Figure 8: Time in seconds for explaining all query answers w.r.t. the size of the ABox for three ratios of conflicts (about 4%, 30%, and 45% of assertions involved in some conflict). The center and right columns focus on the queries whose answering times are lower and whose behaviours are thus not visible in the left column.

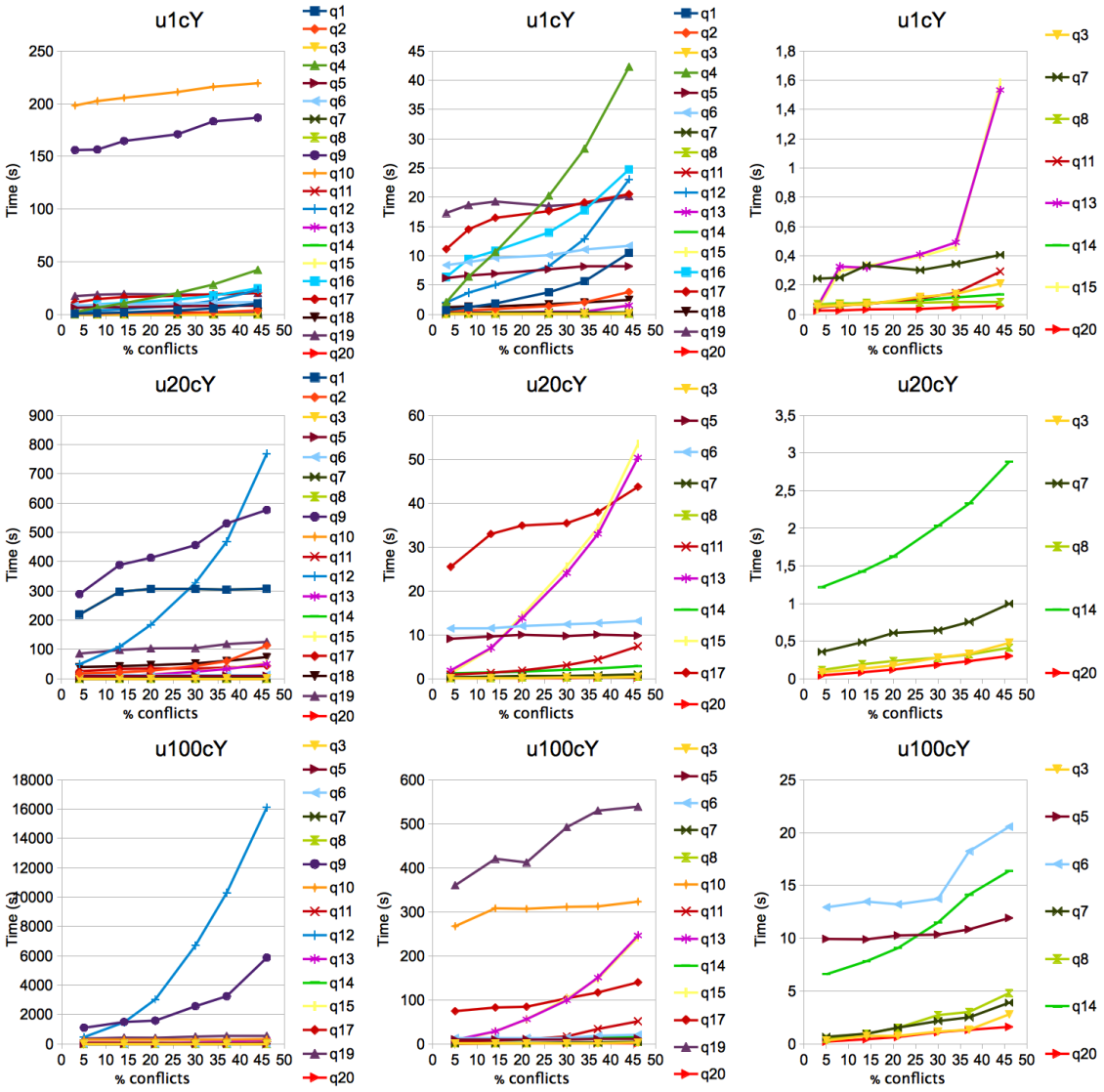


Figure 9: Time in seconds for explaining all query answers w.r.t. the ratios of conflicts for three ABox sizes (about 76 thousand, 2 million, and 10 million assertions). The center and right columns focus on the queries whose answering times are lower and whose behaviours are thus not visible in the left column.

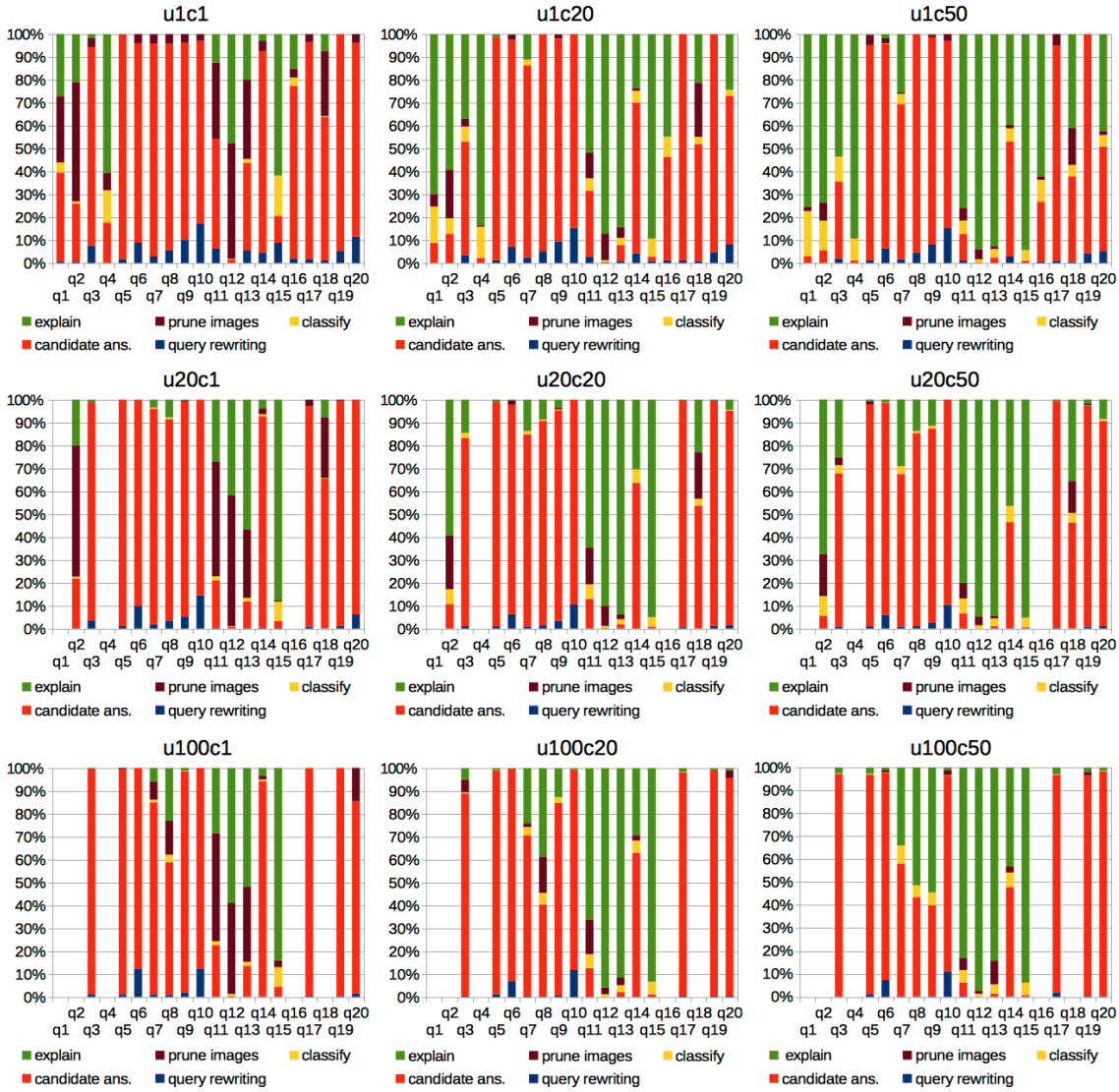


Figure 10: Proportion of time spent by CQAPri in the different phases of query answers explanation on 9 ABoxes of differing sizes and proportions of assertions in conflicts. The two lower bars are the time for rewriting the query and executing the rewritten query to get candidate answers, the middle bar is the time needed to classify such answers, and the two upper bars give the added cost of generating explanations, which is divided into the time spent computing the causes by pruning the non-minimal causes and the time needed to compute the explanations from the causes and conflicts.

		u1c1					u1c20					u1c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q1	Likely	0					7	100	0	0	0	19	100	0	0	0
	Poss.	380	100	0	0	0	7864	100	0	0	0	13747	99.96	0.04	0	0
q2	Likely	20	100	0	0	0	402	100	0	0	0	887	100	0	0	0
	Poss.	12	100	0	0	0	734	100	0	0	0	2087	99.95	0.05	0	0
q3	Poss.	0					85	100	0	0	0	87	100	0	0	0
q4	Poss.	5636	99.96	0.04	0	0	60236	100	0	0	0	80839	99.99	<0.01	<0.01	0
q5	Likely	0					10	100	0	0	0	0				
	Poss.	0					0					10	100	0	0	0
q6	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Poss.	1	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q9	Likely	3	100	0	0	0	68	100	0	0	0	116	100	0	0	0
	Poss.	80	100	0	0	0	406	100	0	0	0	741	99.87	0.13	0	0
q10	Poss.	3	100	0	0	0	6	100	0	0	0	7	100	0	0	0
q11	Likely	0					4	100	0	0	0	7	100	0	0	0
	Poss.	4	100	0	0	0	89	100	0	0	0	236	99.58	0.42	0	0
q12	Likely	11	100	0	0	0	174	100	0	0	0	345	100	0	0	0
	Poss.	10	70	0	30	0	117	79.49	0	20.51	0	350	75.42	2.29	22.29	0
q13	Likely	3	100	0	0	0	71	100	0	0	0	169	100	0	0	0
	Poss.	8	100	0	0	0	122	99.18	0	0.82	0	351	98.58	0.28	1,14	0
q14	Poss.	4	100	0	0	0	97	100	0	0	0	159	100	0	0	0
q15	Poss.	102	100	0	0	0	409	99.76	0	0.24	0	515	99.03	0	0.97	0
q16	Poss.	3987	100	0	0	0	15480	99.99	0.01	0	0	17532	99.99	0.01	0	0
q17	Poss.	1	100	0	0	0	1	100	0	0	0	1	100	0	0	0
q18	Poss.	66	100	0	0	0	872	100	0	0	0	1871	100	0	0	0
q20	Poss.	0					25	100	0	0	0	50	100	0	0	0

		u20c1					u20c20					u20c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q2	Likely	228	100	0	0	0	9927	100	0	0	0	23819	>99.99	<0.01	0	0
	Poss.	1019	100	0	0	0	22489	>99.99	<0.01	0	0	52117	99.97	0.03	0	0
q3	Poss.	0					85	100	0	0	0	87	100	0	0	0
q5	Likely	0					10	100	0	0	0	0				
	Poss.	0					0					10	100	0	0	0
q6	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Poss.	46	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q8	Poss.	31	100	0	0	0	31	100	0	0	0	32	100	0	0	0
q9	Likely	60	100	0	0	0	59	100	0	0	0	267	100	0	0	0
	Poss.	2714	100	0	0	0	13282	99.92	0.08	0	0	21419	94.47	5.53	0	0
q10	Poss.	58	100	0	0	0	62	100	0	0	0	66	100	0	0	0
q11	Likely	0					42	100	0	0	0	267	100	0	0	0
	Poss.	145	100	0	0	0	2781	99.93	0.07	0	0	6373	99.76	0.24	0	0
q12	Likely	218	100	0	0	0	880	100	0	0	0	991	100	0	0	0
	Poss.	251	66.53	13.55	19.92	0	3881	59.26	17.39	23.35	0	8769	56.43	14.85	28.72	0
q13	Likely	64	100	0	0	0	1780	100	0	0	0	4185	100	0	0	0
	Poss.	204	98.53	0	1.47	0	4028	98.01	0.05	1.94	0	9430	98.35	0.17	1.48	0
q14	Poss.	166	100	0	0	0	2412	100	0	0	0	3944	100	0	0	0
q15	Poss.	1702	99.82	0	0.18	0	12143	99.28	0.02	0.70	0	13946	98.72	0.01	1.27	0
q17	Poss.	10	100	0	0	0	37	100	0	0	0	39	100	0	0	0
q18	Poss.	1342	100	0	0	0	24959	99.97	0.03	0	0	48770	99.99	0.01	0	0
q19	Poss.	1	100	0	0	0	8	75	25	0	0	20	10	90	0	0
q20	Poss.	0					25	100	0	0	0	50	100	0	0	0

Table 10: Distribution of the times (in second) for explaining $\mathcal{K} \not\models_{\mathrm{S}} q(\vec{a})$ in Case 1.

		u100c1					u100c20					u100c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q3	Poss.	0					85	100	0	0	0	87	98.85	0	0	1.15
q5	Likely	0					10	100	0	0	0	0				
	Poss.	0					0					10	10	90	0	0
q6	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Poss.	103	100	0	0	0	138	99.28	0	0.72	0	149	95.30	4.03	0.67	0
q8	Poss.	187	100	0	0	0	188	100	0	0	0	190	42.63	56.84	0.53	0
q9	Likely	110	100	0	0	0	192	100	0	0	0	1300	100	0	0	0
	Poss.	27107	99.99	0.01	0	0	64820	90.70	9.30	0	0	105450	75.55	22.30	2	0.15
q10	Poss.	293	100	0	0	0	310	98.39	1.61	0	0	326	51.23	48.77	0	0
q11	Likely	0					192	100	0	0	0	1300	100	0	0	0
	Poss.	739	100	0	0	0	13778	99.97	0.03	0	0	31764	99.84	0.16	0	0
q12	Likely	566	100	0	0	0	1166	100	0	0	0	2849	100	0	0	0
	Poss.	1109	64.65	0.09	35.26	0	18162	61.67	0.22	31.26	6.84	41644	58.84	2.84	25.60	12.72
q13	Likely	308	100	0	0	0	8902	100	0	0	0	21279	100	0	0	0
	Poss.	1014	98.82	0	1.18	0	19737	98.60	0.02	1.38	0	46553	98.62	0.10	1.28	0
q14	Poss.	777	100	0	0	0	11717	100	0	0	0	19165	99.99	0	0	0.01
q15	Poss.	7584	99.84	0	0.16	0	61492	99.53	<0.01	0.47	0	69599	98.99	0.01	1.00	0
q17	Poss.	190	100	0	0	0	221	95.93	4.07	0	0	226	14.60	84.52	0.88	0
q19	Poss.	5	100	0	0	0	56	91.07	8.93	0	0	124	6.45	93.55	0	0
q20	Poss.	0					25	100	0	0	0	50	100	0	0	0

Table 10: Distribution of the times (in second) for explaining $\mathcal{K} \not\models_S q(\vec{a})$ in Case 1.

		u1c1					u1c20					u1c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q1	Likely	0					7	100	0	0	0	19	100	0	0	0
	Poss.	380	100	0	0	0	7864	100	0	0	0	13747	100	0	0	0
q2	Likely	20	100	0	0	0	402	100	0	0	0	887	100	0	0	0
	Poss.	12	100	0	0	0	734	100	0	0	0	2087	100	0	0	0
q3	Poss.	0					85	100	0	0	0	87	100	0	0	0
q4	Poss.	5636	100	0	0	0	60236	100	0	0	0	80839	100	0	0	0
q5	Likely	0					10	100	0	0	0	0				
	Poss.	0					0					10	100	0	0	0
q6	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Poss.	1	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q9	Likely	3	100	0	0	0	68	100	0	0	0	116	100	0	0	0
	Poss.	80	100	0	0	0	406	100	0	0	0	741	100	0	0	0
q10	Poss.	3	100	0	0	0	6	100	0	0	0	7	100	0	0	0
q11	Likely	0					4	100	0	0	0	7	100	0	0	0
	Poss.	4	100	0	0	0	89	100	0	0	0	236	100	0	0	0
q12	Likely	11	100	0	0	0	174	100	0	0	0	345	100	0	0	0
	Poss.	10	100	0	0	0	117	100	0	0	0	350	100	0	0	0
q13	Likely	3	100	0	0	0	71	100	0	0	0	169	100	0	0	0
	Poss.	8	100	0	0	0	122	100	0	0	0	351	100	0	0	0
q14	Poss.	4	100	0	0	0	97	100	0	0	0	159	100	0	0	0
q15	Poss.	102	100	0	0	0	409	100	0	0	0	515	100	0	0	0
q16	Poss.	3987	100	0	0	0	15480	100	0	0	0	17532	100	0	0	0
q17	Poss.	1	100	0	0	0	1	100	0	0	0	1	100	0	0	0
q18	Poss.	66	100	0	0	0	872	100	0	0	0	1871	100	0	0	0
q20	Poss.	0					25	100	0	0	0	50	100	0	0	0

Table 11: Distribution of the times (in second) for explaining $\mathcal{K} \not\models_S q(\vec{a})$ in Case 2.

		u20c1					u20c20					u20c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q2	Likely	228	100	0	0	0	9927	100	0	0	0	23819	100	0	0	0
	Poss.	1019	100	0	0	0	22489	100	0	0	0	52117	100	0	0	0
q3	Poss.	0					85	100	0	0	0	87	100	0	0	0
q5	Likely	0					10	100	0	0	0	0				
	Poss.	0					0					10	100	0	0	0
q6	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Poss.	46	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q8	Poss.	31	100	0	0	0	31	100	0	0	0	32	100	0	0	0
q9	Likely	60	100	0	0	0	59	100	0	0	0	267	100	0	0	0
	Poss.	2714	100	0	0	0	13282	100	0	0	0	21419	99.94	0.06	0	0
q10	Poss.	58	100	0	0	0	62	100	0	0	0	66	100	0	0	0
q11	Likely	0					42	100	0	0	0	267	100	0	0	0
	Poss.	145	100	0	0	0	2781	100	0	0	0	6373	100	0	0	0
q12	Likely	218	100	0	0	0	880	100	0	0	0	991	100	0	0	0
	Poss.	251	100	0	0	0	3881	100	0	0	0	8769	100	0	0	0
q13	Likely	64	100	0	0	0	1780	100	0	0	0	4185	100	0	0	0
	Poss.	204	100	0	0	0	4028	100	0	0	0	9430	100	0	0	0
q14	Poss.	166	100	0	0	0	2412	100	0	0	0	3944	100	0	0	0
q15	Poss.	1702	100	0	0	0	12143	100	0	0	0	13946	100	0	0	0
q17	Poss.	10	100	0	0	0	37	100	0	0	0	39	100	0	0	0
q18	Poss.	1342	100	0	0	0	24959	100	0	0	0	48770	100	0	0	0
q19	Poss.	1	100	0	0	0	8	100	0	0	0	20	100	0	0	0
q20	Poss.	0					25	100	0	0	0	50	100	0	0	0

		u100c1					u100c20					u100c50				
		#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q3	Poss.	0					85	100	0	0	0	87	100	0	0	0
q5	Likely	0					10	100	0	0	0	0				
	Poss.	0					0					10	100	0	0	0
q6	Likely	0					14	100	0	0	0	0				
	Poss.	0					110	100	0	0	0	342	100	0	0	0
q7	Poss.	103	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q8	Poss.	187	100	0	0	0	188	100	0	0	0	190	100	0	0	0
q9	Likely	110	100	0	0	0	192	100	0	0	0	1300	100	0	0	0
	Poss.	27107	100	0	0	0	64820	97.86	2.14	0	0	105450	93.76	6.07	0.02	0.15
q10	Poss.	293	100	0	0	0	310	100	0	0	0	326	100	0	0	0
q11	Likely	0					192	100	0	0	0	1300	100	0	0	0
	Poss.	739	100	0	0	0	13778	100	0	0	0	31764	100	0	0	0
q12	Likely	566	100	0	0	0	1166	100	0	0	0	2849	100	0	0	0
	Poss.	1109	100	0	0	0	18162	100	0	0	0	41644	>99.99	<0.01	0	0
q13	Likely	308	100	0	0	0	8902	100	0	0	0	21279	100	0	0	0
	Poss.	1014	100	0	0	0	19737	100	0	0	0	46553	100	0	0	0
q14	Poss.	777	100	0	0	0	11717	100	0	0	0	19165	100	0	0	0
q15	Poss.	7584	100	0	0	0	61492	100	0	0	0	69599	100	0	0	0
q17	Poss.	190	100	0	0	0	221	100	0	0	0	226	100	0	0	0
q19	Poss.	5	100	0	0	0	56	100	0	0	0	124	100	0	0	0
q20	Poss.	0					25	100	0	0	0	50	100	0	0	0

Table 11: Distribution of the times (in second) for explaining $\mathcal{K} \not\models_{\mathrm{S}} q(\vec{a})$ in Case 2.

	u1c1					u1c20					u1c50				
	#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q1 Poss.	380	100	0	0	0	7864	100	0	0	0	13747	100	0	0	0
q2 Poss.	12	100	0	0	0	734	100	0	0	0	2087	100	0	0	0
q3 Poss.	0					85	100	0	0	0	87	100	0	0	0
q4 Poss.	5636	100	0	0	0	60236	100	0	0	0	80839	>99.99	<0.01	0	0
q5 Poss.	0					0					10	100	0	0	0
q6 Poss.	0					110	100	0	0	0	342	100	0	0	0
q7 Poss.	1	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q9 Poss.	80	100	0	0	0	406	100	0	0	0	741	100	0	0	0
q10 Poss.	3	100	0	0	0	6	100	0	0	0	7	100	0	0	0
q11 Poss.	4	100	0	0	0	89	100	0	0	0	236	100	0	0	0
q12 Poss.	10	70	20	10	0	117	79.49	16.24	4.27	0	350	77.71	18.29	4.00	0
q13 Poss.	8	100	0	0	0	122	100	0	0	0	351	98.87	0.85	0.28	0
q14 Poss.	4	100	0	0	0	97	100	0	0	0	159	100	0	0	0
q15 Poss.	102	100	0	0	0	409	>99.99	<0.01	0	0	515	99.03	0.78	0.19	0
q16 Poss.	3987	100	0	0	0	15480	100	0	0	0	17532	>99.99	<0.01	0	0
q17 Poss.	1	100	0	0	0	1	100	0	0	0	1	100	0	0	0
q18 Poss.	66	100	0	0	0	872	100	0	0	0	1871	100	0	0	0
q20 Poss.	0					25	100	0	0	0	50	100	0	0	0

	u20c1					u20c20					u20c50				
	#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q2 Poss.	1019	100	0	0	0	22489	100	0	0	0	52117	>99.99	<0.01	0	0
q3 Poss.	0					85	100	0	0	0	87	100	0	0	0
q5 Poss.	0					0					10	100	0	0	0
q6 Poss.	0					110	100	0	0	0	342	100	0	0	0
q7 Poss.	46	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q8 Poss.	31	100	0	0	0	31	100	0	0	0	32	100	0	0	0
q9 Poss.	2714	100	0	0	0	13282	99.96	0.04	0	0	21419	99.69	0.31	0	0
q10 Poss.	58	100	0	0	0	62	100	0	0	0	66	100	0	0	0
q11 Poss.	145	100	0	0	0	2781	100	0	0	0	6373	99.98	0.02	0	0
q12 Poss.	251	79.28	13.95	6.77	0	3881	76.04	18.01	5.95	0	8769	71.05	22.98	5.97	0
q13 Poss.	204	98.53	0.98	0.49	0	4028	98.06	1.29	0.65	0	9430	98.51	1.04	0.45	0
q14 Poss.	166	100	0	0	0	2412	100	0	0	0	3944	100	0	0	0
q15 Poss.	1702	99.82	0.12	0.06	0	12143	99.29	0.50	0.21	0	13946	98.72	0.92	0.35	0
q17 Poss.	10	100	0	0	0	37	100	0	0	0	39	100	0	0	0
q18 Poss.	1342	100	0	0	0	24959	100	0	0	0	48770	100	0	0	0
q19 Poss.	1	100	0	0	0	8	100	0	0	0	20	100	0	0	0
q20 Poss.	0					25	100	0	0	0	50	100	0	0	0

	u100c1					u100c20					u100c50				
	#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q3 Poss.	0					85	100	0	0	0	87	100	0	0	0
q5 Poss.	0					0					10	100	0	0	0
q6 Poss.	0					110	100	0	0	0	342	100	0	0	0
q7 Poss.	103	100	0	0	0	138	100	0	0	0	149	100	0	0	0
q8 Poss.	187	100	0	0	0	188	100	0	0	0	190	100	0	0	0
q9 Poss.	27107	100	0	0	0	64820	95.40	4.60	0	0	105450	93.64	6.13	0.08	0.15
q10 Poss.	293	100	0	0	0	310	100	0	0	0	326	99.69	0.31	0	0
q11 Poss.	739	100	0	0	0	13778	100	0	0	0	31764	99.99	0.01	0	0
q12 Poss.	1109	64.74	29.94	5.32	0	18162	61.88	27.74	10.38	<0.01	41644	61.68	27.59	10.72	0.01
q13 Poss.	1014	98.82	0.88	0.30	0	19737	98.63	0.84	0.53	0	46553	98.71	0.78	0.50	0.01
q14 Poss.	777	100	0	0	0	11717	100	0	0	0	19165	99.99	0	0.01	0
q15 Poss.	7584	99.84	0.12	0.04	0	61492	99.53	0.29	0.18	0	69599	99.00	0.63	0.37	0
q17 Poss.	190	100	0	0	0	221	100	0	0	0	226	100	0	0	0
q19 Poss.	5	100	0	0	0	56	100	0	0	0	124	100	0	0	0
q20 Poss.	0					25	100	0	0	0	50	100	0	0	0

Table 12: Distribution of the times (in second) for explaining $\mathcal{K} \not\models_{\mathrm{AR}} q(\vec{a})$ in Case 3.

	u1c1					u1c20					u1c50				
	#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q1 Likely	0					7	100	0	0	0	19	100	0	0	0
q2 Likely	20	100	0	0	0	402	100	0	0	0	887	100	0	0	0
q5 Likely	0					10	100	0	0	0	0				
q6 Likely	0					14	100	0	0	0	0				
q9 Likely	3	100	0	0	0	68	100	0	0	0	116	100	0	0	0
q11 Likely	0					4	100	0	0	0	7	100	0	0	0
q12 Likely	11	100	0	0	0	174	100	0	0	0	345	100	0	0	0
q13 Likely	3	100	0	0	0	71	100	0	0	0	169	100	0	0	0

	u20c1					u20c20					u20c50				
	#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q2 Likely	228	100	0	0	0	9927	100	0	0	0	23819	99.82	0.18	0	0
q5 Likely	0					10	100	0	0	0	0				
q6 Likely	0					14	100	0	0	0	0				
q9 Likely	60	100	0	0	0	59	100	0	0	0	267	100	0	0	0
q11 Likely	0					42	100	0	0	0	267	100	0	0	0
q12 Likely	218	100	0	0	0	880	>99.99	0	0	<0.01	991	TO	TO	TO	TO
q13 Likely	64	100	0	0	0	1780	100	0	0	0	4185	100	0	0	0

	u100c1					u100c20					u100c50				
	#ans	0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1	#ans	<0.01	[0.01, 0.1[	[0.1, 1[	>1
q5 Likely	0					10	100	0	0	0	0				
q6 Likely	0					14	100	0	0	0	0				
q9 Likely	110	100	0	0	0	192	91.15	1.04	0	7.81	1300	99.84	0.08	0	0.08
q11 Likely	0					192	100	0	0	0	1300	100	0	0	0
q12 Likely	566	100	0	0	0	1166	89.28	10.63	0	0.09	2849	TO	TO	TO	TO
q13 Likely	308	100	0	0	0	8902	99.80	0.20	0	0	21279	99.44	0.56	0	0

Table 13: Distribution of the times (in second) for explaining $\mathcal{K} \not\models_{\text{IAR}} q(\vec{a})$ in Case 4.

References

Arenas, M., Bertossi, L. E., & Chomicki, J. (1999). Consistent query answers in inconsistent databases. In *Proceedings of PODS*.

Arioua, A., & Croitoru, M. (2016a). Dialectical characterization of consistent query explanation with existential rules. In *Proceedings of FLAIRS*.

Arioua, A., & Croitoru, M. (2016b). A dialectical proof theory for universal acceptance in coherent logic-based argumentation frameworks. In *Proceedings of ECAI*.

Arioua, A., Croitoru, M., & Buche, P. (2016). DALEK: A tool for dialectical explanations in inconsistent knowledge bases. In *Proceedings of COMMA*.

Arioua, A., Tamani, N., & Croitoru, M. (2014). On conceptual graphs and explanation of query answering under inconsistency. In *Proceedings of ICCS*.

Arioua, A., Tamani, N., & Croitoru, M. (2015). Query answering explanation in inconsistent Datalog+/- knowledge bases. In *Proceedings of DEXA*.

Arora, S., & Barak, B. (2009). *Computational Complexity - A Modern Approach*. Cambridge University Press.

Artale, A., Calvanese, D., Kontchakov, R., & Zakharyaschev, M. (2009). The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)*, *36*, 1–69.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. F. (Eds.). (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F., Peñaloza, R., & Suntisrivaraporn, B. (2007). Pinpointing in the description logic EL^+. In *Proceedings of KI*.

Bail, S., Parsia, B., & Sattler, U. (2013). The logical diversity of explanations in OWL ontologies. In *Proceedings of CIKM*.

Benferhat, S., Bouraoui, Z., Croitoru, M., Papini, O., & Tabia, K. (2016). Non-objection inference for inconsistency-tolerant query answering. In *Proceedings of IJCAI*.

Berre, D. L., & Parrain, A. (2010). The sat4j library, release 2.2. *JSAT*, *7*(2-3), 59–64.

Bertossi, L. E. (2006). Consistent query answering in databases. *SIGMOD Record*, *35*(2), 68–76.

Bertossi, L. E. (2011). *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Bidoit, N., Herschel, M., & Tzompanaki, K. (2014). Query-based why-not provenance with nedexplain. In *Proceedings of EDBT*.

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

Bienvenu, M., & Bourgaux, C. (2016). Inconsistency-tolerant querying of description logic knowledge bases. In *Reasoning Web, Tutorial Lectures*, pp. 156–202.

Bienvenu, M., Bourgaux, C., & Goasdoué, F. (2014). Querying inconsistent description logic knowledge bases under preferred repair semantics. In *Proceedings of AAAI*.

Bienvenu, M., Bourgaux, C., & Goasdoué, F. (2016). Explaining inconsistency-tolerant query answering over description logic knowledge bases. In *Proceedings of AAAI*.

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

Borgida, A., Calvanese, D., & Rodriguez-Muro, M. (2008). Explanation in the DL-Lite family of description logics. In *Proceedings of OTM*.

Borgida, A., Franconi, E., & Horrocks, I. (2000). Explaining ALC subsumption. In *Proceedings of ECAI*.

Boros, E., Elbassioni, K. M., Gurvich, V., & Khachiyan, L. (2000). An efficient incremental algorithm for generating all maximal independent sets in hypergraphs of bounded dimension. *Parallel Processing Letters*, *10*(4), 253–266.

Bourgaux, C. (2016). *Inconsistency Handling in Ontology-Mediated Query Answering*. Ph.D. thesis, University of Paris-Saclay, France.

Bursztyn, D., Goasdoué, F., & Manolescu, I. (2015). Efficient query answering in DL-Lite through FOL reformulation (extended abstract). In *Proceedings of DL*.

Bursztyn, D., Goasdoué, F., & Manolescu, I. (2016). Teaching an RDBMS about ontological constraints. *PVLDB*, *9*(12), 1161–1172.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2007). Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning (JAR)*, *39*(3), 385–429.

Calvanese, D., Ortiz, M., Simkus, M., & Stefanoni, G. (2013). Reasoning about explanations for negative query answers in DL-Lite. *Journal of Artificial Intelligence Research (JAIR)*, *48*, 635–669.

Chandra, A. K., & Merlin, P. M. (1977). Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of STOC*.

Cheney, J., Chiticariu, L., & Tan, W. C. (2009). Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, *1*(4), 379–474.

Chomicki, J. (2007). Consistent query answering: Five easy pieces. In *Proceedings of ICDT*.

Chomicki, J., Marcinkowski, J., & Staworko, S. (2004a). Computing consistent query answers using conflict hypergraphs. In *Proceedings of CIKM*.

Chomicki, J., Marcinkowski, J., & Staworko, S. (2004b). Hippo: A system for computing consistent answers to a class of SQL queries. In *Proceedings of EDBT*.

Chortaras, A., Trivela, D., & Stamou, G. (2011). Optimized query rewriting for OWL 2 QL. In *Proceedings of CADE*.

Du, J., Qi, G., & Shen, Y.-D. (2013). Weight-based consistent query answering over inconsistent $\mathcal{SHIQ}$ knowledge bases. *Knowledge and Information Systems*, *34*(2), 335–371.

Du, J., Wang, K., & Shen, Y. (2014). A tractable approach to abox abduction over description logic ontologies. In *Proceedings of AAAI*.

Du, J., Wang, K., & Shen, Y. (2015). Towards tractable and practical ABox abduction over inconsistent description logic ontologies. In *Proceedings of AAAI*.

Eiter, T., Ortiz, M., Simkus, M., Tran, T., & Xiao, G. (2012). Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proceedings of AAAI*.

Fuxman, A., Fazli, E., & Miller, R. J. (2005). Conquer: Efficient management of inconsistent databases. In *Proceedings of SIGMOD*.

Fuxman, A., & Miller, R. J. (2005). First-order query rewriting for inconsistent databases. In *Proceedings of ICDT*.

Guo, Y., Pan, Z., & Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, *3*(2-3), 158–182.

Herschel, M., & Hernández, M. A. (2010). Explaining missing answers to SPJUA queries. *PVLDB*, *3*(1), 185–196.

Horridge, M., Bail, S., Parsia, B., & Sattler, U. (2011). The cognitive complexity of OWL justifications. In *Proceedings of ISWC*.

Horridge, M., Parsia, B., & Sattler, U. (2012). Extracting justifications from bioportal ontologies. In *Proceedings of ISWC*.

Immerman, N. (1987). Expressibility as a complexity measure: results and directions. In *Proceedings of Conference on Structure in Complexity Theory*.

Kalyanpur, A., Parsia, B., Sirin, E., & Hendler, J. A. (2005). Debugging unsatisfiable classes in OWL ontologies. *Journal Web Sem.*, *3*(4), 268–293.

Kolaitis, P. G., Pema, E., & Tan, W.-C. (2013). Efficient querying of inconsistent databases with binary integer programming. *PVLDB*, *6*(6), 397–408.

Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., & Savo, D. F. (2010). Inconsistency-tolerant semantics for description logics. In *Proceedings of RR*.

Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., & Savo, D. F. (2011). Query rewriting for inconsistent DL-Lite ontologies. In *Proceedings of RR*.

Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., & Savo, D. F. (2015). Inconsistency-tolerant query answering in ontology-based data access. *Journal Web Sem.*, *33*, 3–29.

Liberatore, P. (2005). Redundancy in logic I: Cnf propositional formulae. *Artif. Intell. (AIJ)*, *163*(2), 203–232.

Lukasiewicz, T., Martinez, M. V., & Simari, G. I. (2012). Inconsistency handling in Datalog+/- ontologies. In *Proceedings of ECAI*.

Lukasiewicz, T., Martinez, M. V., & Simari, G. I. (2013). Complexity of inconsistency-tolerant query answering in Datalog+/-. In *Proceedings of OTM*.

Lutz, C., Seylan, I., Toman, D., & Wolter, F. (2013). The combined approach to OBDA: Taming role hierarchies using filters. In *Proceedings of ISWC*.

Marileo, M. C., & Bertossi, L. E. (2010). The consistency extractor system: Answer set programs for consistent query answering in databases. *Data Knowl. Eng.*, *69*(6), 545–572.

McGuinness, D. L., & Borgida, A. (1995). Explaining subsumption in description logics. In *Proceedings of IJCAI*.

Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., & Lutz, C. (2012). OWL 2 Web Ontology Language profiles. W3C Recommendation. Available at http://www.w3.org/TR/owl2-profiles/.

Olteanu, D., & Zavodny, J. (2012). Factorised representations of query results: size bounds and readability. In *Proceedings of ICDT*.

OWL Working Group, W. (2009). *OWL 2 Web Ontology Language: Document overview*. W3C Recommendation. Available at https://www.w3.org/TR/owl2-overview/.

Papadimitriou, C. H. (1995). *Computational complexity*. Addison-Wesley.

Peñaloza, R., & Sertkaya, B. (2010). Complexity of axiom pinpointing in the DL-Lite family of description logics. In *Proceedings of ECAI*.

Pérez-Urbina, H., Horrocks, I., & Motik, B. (2009). Efficient query answering for OWL 2. In *Proceedings of ISWC*.

Rosati, R. (2011). On the complexity of dealing with inconsistency in description logic ontologies. In *Proceedings of IJCAI*.

Rosati, R., & Almatelli, A. (2010). Improving query answering over DL-Lite ontologies. In *Proceedings of KR*.

Rosati, R., Ruzzi, M., Graziosi, M., & Masotti, G. (2012). Evaluation of techniques for inconsistency handling in OWL 2 QL ontologies. In *Proceedings of ISWC*.

Schlobach, S., & Cornet, R. (2003). Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of IJCAI*.

Sebastiani, R., & Vescovi, M. (2009). Axiom pinpointing in lightweight description logics via horn-sat encoding and conflict analysis. In *Proceedings of CADE*.

Selman, B., & Kautz, H. A. (1991). Knowledge compilation using horn approximations. In *Proceedings of AAAI*.

Thomazo, M. (2013). Compact rewritings for existential rules. In *Proceedings of IJCAI*.

Tsalapati, E., Stoilos, G., Stamou, G. B., & Koletsos, G. (2016). Efficient query answering over expressive inconsistent description logics. In *Proceedings of IJCAI*.

Wang, Z., Chitsaz, M., Wang, K., & Du, J. (2015). Towards scalable and complete query explanation with OWL 2 EL ontologies. In *Proceedings of CIKM*.

Zhou, Y., Grau, B. C., Nenov, Y., Kaminski, M., & Horrocks, I. (2015). Pagoda: Pay-as-you-go ontology query answering using a Datalog reasoner. *Journal of Artificial Intelligence Research (JAIR)*, *54*, 309–367.