Ontology-Mediated SPARQL Query Answering over Knowledge Graphs

Guohui Xiao *, Julien Corman

Faculty of Computer Science, Free University of Bozen-Bolzano, 390100, Bolzano, Italy

A R T I C L E   I N F O

Article history:
Received 16 June 2020
Received in revised form 16 November 2020
Accepted 29 November 2020
Available online 21 December 2020

A B S T R A C T

Ontology-Mediated Query Answering (OMQA) is a well-established framework to answer queries over Knowledge Graphs (KGs), enriched with rdfs or owl ontologies. OMQA was originally designed for Unions of Conjunctive Queries (UCQs), and based on certain answers. More recently, OMQA has been extended to SPARQL queries, but to our knowledge, none of the efforts made in this direction (either in the literature, or the so-called W3C SPARQL entailment regimes) is able to capture both certain answers for UCQs and the standard interpretation of SPARQL over a plain graph. We formalize these as requirements to be met by any semantics that aims at conciliating certain answers and SPARQL answers, and extend these with three additional requirements. Then we define two semantics that satisfy all requirements for SPARQL queries with SELECT, UNION, JOIN, and OPTIONAL. Finally, we investigate the combined complexity of query answering under these semantics over a KG enriched with a DL-Liteq ontology, showing that for several fragments of SPARQL, known upper-bounds for query answering over a plain KG are matched.

© 2020 Elsevier Inc. All rights reserved.

1. Introduction

Knowledge Graphs (KGs) and ontologies play an important role in big data management and processing. A standard task in this setting is querying a KG enriched with a background theory, specified as an ontology. The SPARQL query language, standardized by the W3C, is an expressive SQL-like language designed for RDF KGs [14]. Recently, SPARQL has been extended with so-called entailment regimes, which specify different semantics to query an RDF KG enriched with an rdfs or owl ontology [12]. This allows retrieving answers to a query not only over the facts explicitly stated in the KG, but more generally over what can be inferred from the KG and ontology.

The SPARQL entailment regimes are in turn largely influenced by theoretical work on Ontology-Mediated Query Answering (OMQA), notably in the field of Description Logics (DLs) [3]. However, OMQA was initially developed for unions of conjunctive queries (UCQs), which have a limited expressivity when compared to SPARQL. And as explained in [1], conciliating the standard (compositional) semantics of SPARQL on the one hand, and the semantics used for OMQA on the other hand, called certain answers, is non-trivial. In this work, we build upon [1], and further investigate how SPARQL and certain answers can be combined.

As an illustration, Example 1 provides a simple RDF KG, an owl ontology and a SPARQL query. The KG (a.k.a. ABox) $\mathcal{A}$ states that Alice is a driver, whereas the ontology (a.k.a. TBox) $\mathcal{T}$ states that a driver must have a license (for conciseness, we use DLs as an abstract syntax, rather than some concrete syntax(es) for RDF and owl). Finally, the SPARQL query $q$ retrieves all individuals that have a license.

Example 1. $\mathcal{A} = \{ \text{Driver}(Alice) \}$
$\mathcal{T} = \{ \text{Driver} \sqsubseteq \text{hasLicense} \}$
$q = \text{SELECT } ?x \text{ WHERE } \{ ?x \text{ hasLicense } ?y \}$

One may expect Alice to be retrieved as an answer to $q$. And it would be the case under certain answer semantics, if one considers the natural translation of this query into a UCQ: $q(x) \leftarrow \text{hasLicense}(x, y)$. However, under the standard semantics of SPARQL 1.1 [14], this

* Corresponding author.

E-mail addresses: xiao@inf.unibz.it (G. Xiao), corman@inf.unibz.it (J. Corman).

1 Different conventions are used in the literature for the meanings of "Knowledge Graphs" and "ontology". In this paper, we make a clear distinction between the two (although in practice both can be encoded as sets of RDF triples). So we consider that a KG contains only statements about individuals, whereas an ontology contains only conceptual knowledge about the underlying domain. In Description Logics, KG and ontology in this sense correspond to the so-called ABox and TBox, respectively.
query has no answer. This is expected, because Alice has no driving license declared in the ABox, and because this semantics does not take into account knowledge that may be inferred from the graph and ontology. More surprisingly though, under all SPARQL entailment regimes [12], this query also has no answer.

This mismatch between certain answers and entailment regimes has already been discussed in depth in [1], where the interpretation of the OPTIONAL operator of SPARQL is identified as a key challenge, when trying to define a suitable semantics for SPARQL that complies with certain answers for UCQs. A concrete proposal is also made in [1] in this direction. However, this semantics does not comply with the standard semantics of SPARQL when the TBox is empty. This means that a same query over a plain RDF graph may yield different answers, depending on whether it is evaluated under this semantics, or under the one defined in the SPARQL 1.1 specification [14].

We propose in this article to investigate whether and how this dilemma can be solved. To this end, we first formulate in Section 4 four requirements to be met by any reasonable semantics meant to conciliate certain answers and standard SPARQL answers. Then in Section 5, we use these requirements to review different (existing and new) semantics. We also show that all requirements can be met, for the fragment of SPARQL with SELECT, UNION, JOIN, and OPTIONAL, by proposing two semantics, the first one defined only for DLs with the so-called canonical model property, and the second one for arbitrary DLs. Finally, in Section 6, we provide combined complexity results for query answering under these semantics, over KGs and ontologies in DL-Lite, a popular DL tailored for query answering, which underpins the OWL2 standard. We show in particular that for several (sub-)fragments of SPARQL, upper bounds for these problems match results already known to hold for SPARQL over KGs, which means that under this semantics, and as far as worst-case complexity is concerned, the presence of a TBox does not introduce a computational overhead.

An earlier conference version of this article was presented in JIST 2019, the 9th Joint International Semantic Technology Conference [8]. In the current version, apart from more explanations, examples and proofs, the most significant addition is the second semantics, called epistemic certain answers (defined in Section 5.4). This semantics also satisfies all four requirements, and has a more declarative flavor than the one proposed in the earlier version, called maximal admissible canonical answers (defined in Section 5.3).

Before presenting our technical contributions, Section 2 introduces preliminary notions, and Section 3 reviews existing semantics for SPARQL over a KG and ontology.

2. Preliminaries

We assume infinite and mutually disjoint sets NI, NC, NR, and NV of individuals (constants), concept names (unary predicates), role names (binary predicates), and variables respectively. We also assume an infinite universe U, such that NI ⊆ U. For clarity, we abstract away from concrete domains (as well as RDF term types), since these are irrelevant to the content of this paper. We also assume that NI, NC and NR do not contain any reserved term from the RDF/RDFS/OWL vocabularies (such as rdfs:subClassOf, owl:disjointWith, etc.).

2.1. RDF and SPARQL

An (RDF) triple is an element of ((NI × [rdfs:type] × NC) ∪ (NI × NR × NI)). An RDF KG, or simply graph, is a set of triples, noted A.

For the concrete syntax of SPARQL, we refer to the specification [14]. Following [1], we focus on SPARQL queries whose triple patterns are either in (NV ∪ NI) × [rdfs:type] × NC, or in (NV ∪ NI) × NR × (NV ∪ NI). For readability, we represent triples and triple patterns as atoms in prefix notation, i.e., we use A(e) rather than (e, rdfs:type, A) and for r ∈ NR, we use r(e1, e2) rather than (e1, r, e2). If q is a SPARQL query, we use vars(q) to denote the set of variables projected by q.

We adopt (roughly) the abstract syntax provided in [24] for the fragment of SPARQL with the SELECT, UNION and OPTIONAL operators, using the following grammar, where t is a SPARQL triple pattern, and X ⊆ NV:

\[ q ::= t \mid \text{SELECT}_X q \mid q \text{ UNION } q \mid q \text{ JOIN } q \mid q \text{ OPT } q \]

In addition, if q = \text{SELECT}_X q', then X ⊆ vars(q') must hold. In order to refer to fragments of this language, we use the letters S, U, J and O (in this order), for SELECT, UNION, JOIN, and OPT respectively. E.g., “SUJO” stands for the full language, “UJ” for the fragment with UNION and JOIN only, etc.

If ω is a function, we use dom(ω) (resp. range(ω)) to designate its domain (resp. range). Two functions ω1 and ω2 are compatible, denoted with ω1 ≃ ω2, iff ω1(x) = ω2(x) for each x ∈ dom(ω1) ∩ dom(ω2). If ω1 and ω2 are compatible, then ω1 ∪ ω2 is the only function with domain dom(ω1) ∪ dom(ω2) that is compatible with ω1 and ω2. We say that a function ω2 extends a function ω1, noted ω1 ≤ ω2, iff dom(ω1) ⊆ dom(ω2) and ω1 ≃ ω2. Finally, we use ω|X (resp. ω|X) to designate the restriction of function ω to domain (resp. co-domain) X, i.e. ω|X is the only function compatible with ω that verifies dom(ω|X) = dom(ω) ∩ X, and ω|X is the only function compatible with ω that verifies dom(ω|X) = {v ∈ dom(ω) | ω(v) ∈ X}.

A solution mapping is a function from a finite subset of NV to U. If Ω1 and Ω2 are sets of solutions mappings and X ⊆ NV, then:

\[ Ω_1 \sqcap Ω_2 = \{ ω_1 \cup ω_2 : (ω_1, ω_2) ∈ Ω_1 \times Ω_2 \text{ and } ω_1 \sim ω_2 \} \]
\[ Ω_1 \setminus Ω_2 = \{ ω_1 \in Ω_1 \text{ and } ω_1 \sim ω_2 \text{ for all } ω_2 ∈ Ω_2 \} \]
\[ π_X Ω = \{ ω|X : ω ∈ Ω \} \]

If q is a SPARQL query and ω a solution mapping s.t. vars(q) ⊆ dom(ω), we use ω(q) to designate the query identical to q, but where each occurrence of variable x in a triple pattern is replaced by ω(x).

We now reproduce the inductive definition of answers to a SPARQL query q over a graph A, denoted sparlAns(q, A), provided in [24] for the SUJO fragment (and for set semantics).
Definition 1 (SPARQL answers over a plain graph).

\[
\begin{align*}
\text{sparqlAns}(t, \mathcal{A}) &= \{ \omega \mid \text{dom}(\omega) = \text{vars}(t) \text{ and } \omega(t) \in \mathcal{A} \} \\
\text{sparqlAns}(q_1 \ \text{UNION} \ q_2, \ \mathcal{A}) &= \text{sparqlAns}(q_1, \mathcal{A}) \cup \text{sparqlAns}(q_2, \mathcal{A}) \\
\text{sparqlAns}(q_1 \ \text{JOIN} \ q_2, \ \mathcal{A}) &= \text{sparqlAns}(q_1, \mathcal{A}) \bowtie \text{sparqlAns}(q_2, \mathcal{A}) \\
\text{sparqlAns}(q_1 \ \text{OPT} \ q_2, \ \mathcal{A}) &= (\text{sparqlAns}(q_1, \mathcal{A}) \triangleright \text{sparqlAns}(q_2, \mathcal{A})) \cup \\
&\quad (\text{sparqlAns}(q_1, \mathcal{A}) \setminus \text{sparqlAns}(q_2, \mathcal{A})) \\
\text{sparqlAns}(\text{SELECT}_x q, \mathcal{A}) &= \pi_x \text{sparqlAns}(q, \mathcal{A})
\end{align*}
\]

2.2. Description logic KB, UQCs and certain answers

A Knowledge Base (KB) is a KG enriched with an ontology. As is conventional in the DL literature, we represent a KB \( \mathcal{K} \) as a pair \( \mathcal{K} = (\mathcal{T}, \mathcal{A}) \), where \( \mathcal{A} \) is the KG, i.e. an RDF graph, also called the ABox of \( \mathcal{K} \), and \( \mathcal{T} \) is the ontology, also called the TBox of \( \mathcal{K} \). A TBox is a finite set of logical axioms in some DL. For the syntax of DLs, we refer to [4]. For a KB \( \mathcal{K} = (\mathcal{T}, \mathcal{A}) \), the active domain of \( \mathcal{K} \), denoted with \( \text{aDom}(\mathcal{K}) \), is the set of elements of \( \mathcal{N} \) that appear (syntactically) in \( \mathcal{T} \) or \( \mathcal{A} \).

The semantics of DL KBs is defined in terms of (first-order) interpretations. We adopt in this article the standard name assumption: an interpretation is a structure \( \mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}) \), where the domain \( \Delta^\mathcal{I} \) of \( \mathcal{I} \) is a non-empty subset of \( \mathcal{U} \), and the interpretation function \( \cdot^\mathcal{I} \) of \( \mathcal{I} \) maps each \( c \in \mathcal{N} \) to itself, and each \( A \in \mathcal{N}_2 \) (resp. \( r \in \mathcal{N}_3 \)) to a unary (resp. binary) relation \( A^\mathcal{I} \) (resp. \( r^\mathcal{I} \)) over \( \Delta^\mathcal{I} \). An interpretation \( \mathcal{I} \) is a model of a KB \( \mathcal{K} = (\mathcal{T}, \mathcal{A}) \) if it satisfies every assertion in \( \mathcal{A} \) and axiom in \( \mathcal{T} \). For the formal definition of “satisfies”, we refer to [4].

If \( \mathcal{K} \) is a KB, we use \( \text{mod}(\mathcal{K}) \) to denote the set of models of \( \mathcal{K} \). We focus on satisfiable KBs only, i.e. KBs that admit at least one model, since any formula can be trivially derived from an unsatisfiable KB. We also omit this precision for readability. So “a KB” below is a shortcut for “a satisfiable KB”.

For a DL KB \( \mathcal{K} \), an interpretation \( \mathcal{I}_\mathcal{K} \in \text{mod}(\mathcal{K}) \) is a canonical model of \( \mathcal{K} \) if \( \mathcal{I}_\mathcal{K} \) can be homomorphically mapped to any \( \mathcal{I} \in \text{mod}(\mathcal{K}) \). We say that a DL \( \mathcal{L} \) has the canonical model property if every KB in \( \mathcal{L} \) has a canonical model. This is a key property of DLs tailored for query answering, and many DLs, e.g. DL-Lite\(_R\), DL-Lite\(_\mathcal{Q}\) or Horn-\(SHIQ\), have this property [10].

An interpretation can also be viewed as a (possibly infinite) RDF graph, with triples \( \{ (d, a, r) \mid d \in \Delta, a \in \mathcal{N}_2 \} \cup \{ (d_1, d_2) \mid (d_1, d_2) \in r^\mathcal{I}, r \in \mathcal{N}_3 \} \). This is a slight abuse (the RDF standard does not admit infinite graphs), but we will nonetheless use this convention throughout the article, in order to simplify notation.

A conjunctive query (CQ) \( h \) is an expression of the form:

\[
h(x) \leftarrow p_1(x_1), \ldots, p_m(x_m)
\]

where \( h, p_i \) are predicates and \( x, x_i \) are tuple over \( \mathcal{N}_v \). Abusing notation, we may use \( x \) (resp. \( x_i \)) below to designate the elements of \( x \) (resp. \( x_i \)) viewed as a set. An additional syntactic requirement on a CQ is that \( x \subseteq x_1 \cup \ldots \cup x_m \). The variables in \( x \) are called distinguished, and we use \( \text{vars}(h) \) to designate the distinguished variables of CQ \( h \). We focus in this article on CQs where each \( p_i \) is unary or binary, i.e. \( p_i \in \mathcal{N}_2 \cup \mathcal{N}_3 \). A match for \( h \) in an interpretation \( \mathcal{I} \) is a total function \( \rho \) from \( x_1 \cup \ldots \cup x_m \) to \( \Delta^\mathcal{I} \) such that \( \rho(x_i) \in (p_i)^\mathcal{I} \) for \( i \in \{1, \ldots, m\} \).

A mapping \( \omega \) is an answer to \( h \) over \( \mathcal{I} \) iff there is a match \( \rho \) for \( h \) in \( \mathcal{I} \) s.t. \( \omega = \rho|_{\text{vars}(h)} \).

A union of conjunctive queries (UQC) is a set \( q = \{ h_1, \ldots, h_n \} \) of CQs sharing the same distinguished variables, and \( \omega \) is an answer to \( q \) over \( \mathcal{I} \) iff \( \omega \) is an answer to some \( h_i \) over \( \mathcal{I} \).

Definition 2 (Certain answer). A mapping \( \omega \) is a certain answer to \( q \) over a KB \( \mathcal{K} \) iff \( \omega \) is an answer to \( q \) over each \( \mathcal{I} \in \text{mod}(\mathcal{K}) \). We use \( \text{certAns}(q, \mathcal{K}) \) to designate the set of certain answers to \( q \) over \( \mathcal{K} \).

CQs and UQCs have a straightforward representation as SPARQL queries. The CQ \( h(x) \leftarrow p_1(x_1), \ldots, p_m(x_m) \) in SPARQL syntax is written:

\[
\text{SELECT}_x (p_1(x_1) \mid \ldots \mid p_m(x_m))
\]

And a UQC in SPARQL syntax is of the form:

\[
h_1 \cup \ldots \cup h_n
\]

where each \( h_i \) is a CQ in SPARQL syntax, and \( \text{vars}(h_i) = \text{vars}(h_j) \) for \( i, j \in \{1, \ldots, n\} \).

3. Querying a DL KB with SPARQL: existing semantics

In this section, we review existing semantics for SPARQL over a DL KB. We start by briefly recalling some features of the W3C specification for the SPARQL 1.1 entailment regimes [12]. This specification defines different ways to take into account the semantics of RDF, RDFS or Owl, in order to infer additional answers to a SPARQL query. We ignore the aspects pertaining to querying blank nodes and concept/role names, which fall out of the scope of this paper, and focus on the entailment regimes parameterized by an owl profile, i.e. a DL \( \mathcal{L} \). In short, the \( \mathcal{L} \)-entailment regime modifies the evaluation of a SPARQL query \( q \) over an \( \mathcal{L} \)-KB \( \mathcal{K} = (\mathcal{T}, \mathcal{A}) \) as follows:

1. Triple patterns are not evaluated over the ABox \( \mathcal{A} \), but instead over the so-called entailed graph, which consists of all ABox assertions entailed by \( \mathcal{K} \). This includes assertions of the form \( C(o) \), where \( C \) is a complex concept expression allowed in \( \mathcal{L} \). The semantics of other SPARQL operators is preserved.
2. The SPARQL query can use \( \mathcal{L} \)-concepts in triple pattern, e.g. \( \text{hasLicense}(x) \).
Consider again Example 1 under the owl 2 QL entailment regime for instance, which corresponds (roughly) to the DL DL-Lite$_R$. In this example, the query $\text{hasLicense}(x)$ has $\{x \mapsto \text{Alice}\}$ as unique answer: since the entailed graph contains all ABox assertions entailed by $K$, it contains the assertion $\text{hasLicense}(\text{Alice})$ (again, we use the DL syntax rather than owl, for readability).

So the expressivity of the $L$-entailment regime is limited by the concepts that can be expressed in $L$. This is why [16] proposed to extend the semantics of the owl 2 QL profile, retrieving instances of concepts that cannot be expressed in DL-Lite$_R$ (e.g. concepts of the form $\exists R_1 \exists R_2$). Still, under this semantics as well as all entailment regimes defined in the specification, the query $\text{select}_{(q)} \text{hasLicense}(x, y)$ has no answer over the KB of Example 1, because the entailed graph does not contain any assertion of the form $\text{hasLicense}(\text{Alice}, e)$.

This point was discussed in depth in [1], for the SUJO fragment, and based on remarks made earlier in [2]. The current paper essentially builds upon this discussion, which is why we reproduce it below. A first remark made in [2] and [1] is that the OPT operator of SPARQL prevents the usage of certain answers, even when querying a plain graph (or equivalently, a KB with empty TBox). This can be seen with Example 2.

Example 2.

\[ A = \{ \text{Person(Alice)} \} \]

\[ q = \text{Person}(x) \text{ OPT}\text{hasLicense}(x, y) \]

In this example, according to the SPARQL specification, the mapping $\omega = \{ x \mapsto \text{Alice} \}$ is the only answer to $q$ over $A$, i.e. $\text{sparqlAns}(q, A) = \{ \omega \}$. But $\omega$ is not a certain answer to $q$ over the KB $\langle \emptyset, A \rangle$. Consider for instance the interpretation $I$ defined by $I = A \cup \{ \text{hasLicense(Alice, 123)} \}$. Then $\text{sparqlAns}(q, I) = \{ \{ x \mapsto \text{Alice}, y \mapsto 123 \} \}$. So $\omega \notin \text{certAns}(q, \langle \emptyset, A \rangle)$.

Then in [2] and [1] still, the authors remark that in this example, $\omega$ can nonetheless be extended to an answer in every model of $\langle \emptyset, A \rangle$. This is the main intuition used in [1] to adapt the definition of certain answers to SPARQL queries with OPT.

**Definition 3 ([1]).** If $q$ is a query and $I$ an interpretation, let $\text{eAns}(q, I)$ designate all mappings that can be extended to an answer to $q$ in $I$, i.e.:

\[ \text{eAns}(q, I) = \{ \omega \mid \omega \leq \omega' \text{ for some } \omega' \in \text{sparqlAns}(q, I) \} \]

Then if $K$ is a KB, the set $\text{eCertAns}(q, K)$ of mappings that can be extended to an answer in every model of $K$ is defined as:

\[ \text{eCertAns}(q, K) = \bigcap_{I \in \text{mod}(K)} \text{eAns}(q, I) \]

But as pointed out in [1], $\text{eCertAns}(q, I)$ does not comply with SPARQL answers over a plain graph (i.e. when the TBox is empty). Indeed, if some $\omega$ can be extended to an answer in every model of the KB, then this is also the case of any mapping that $\omega$ extends (e.g. trivially the empty mapping). So in Example 2, $\text{eCertAns}(q, \langle \emptyset, A \rangle) = \{ \{ \} \}$, whereas $\text{sparqlAns}(q, A) = \{ \{ x \mapsto \text{Alice} \} \}$.

The semantics proposed in [1] is designed to solve this issue. The precise scope of the proposal is so-called well-designed SUJO queries (see [24] for a definition), in some normal form (no UNION in the scope of SELECT, JOIN or OPT, NO SELECT in the scope of JOIN or OPT, and no OPT in the scope of JOIN). Given a KB $K$, the solution consists in retaining, for each maximal SUJO subquery $q$, the maximal elements of $\text{eCertAns}(q, K)$ w.r.t. $\subseteq$. An additional restriction is put on the domain of such solution mappings, based on the so-called pattern-tree representation (defined in [19]) of well-designed SJO queries. The UNION operator on the other hand is evaluated compositionally, as in Definition 1.

But as illustrated by the authors, this proposal does not comply with the standard semantics for SPARQL over plain graphs. Example 3 below reproduces the one given in [1, Example 4]:

Example 3.

\[ A = \{ \text{teachesTo(Alice, Bob), knows(Bob, Carol), teachesTo(Alice, Dan)} \} \]

\[ q = \text{SELECT}_{(x,y)}(\text{teachesTo}(x, y) \text{ OPT} \text{known}(y, z)) \]

In this example, $\text{sparqlAns}(q, A) = \{ \{ x \mapsto \text{Alice}, \tau \mapsto \text{Carol} \}, \{ x \mapsto \text{Alice} \} \}$. Instead, the semantics proposed in [1] yields $\{ \{ x \mapsto \text{Alice}, \tau \mapsto \text{Carol} \} \}$. But as pointed out elsewhere, additional work is needed to be mentioned, even though they are not immediately related to the problem addressed in this paper. First, a modification of the entailment regimes’ semantics was proposed in [17] for the SJO fragment extended with the SPARQL FILTER operator. For DLs with negation, it consists in ruling out a partial solution mappings if it cannot be extended to an answer in any model of the KB. Finally, another topic of interest when it comes to SPARQL and certain answers, but which falls out of the scope of this paper, is the treatment of blank nodes, discussed in the specification of SPARQL entailment regimes [12], and more recently in [13] and [15].

### 4. Requirements

As seen in the previous section, existing semantics for SPARQL answers over a KB fail to comply either with certain answers (for the fragment of SPARQL that corresponds to UCQs), or with SPARQL answers over a plain graph when the TBox is empty.

---

2. This is without loss of expressivity, but normalization may cause an exponential blowup.
We will show in Section 5 that these two requirements are compatible for some DLs and fragments of SPARQL. But first, in this section, we formalize these two requirements, as properties to be met by any semantics whose purpose is to conciliate certain answers and SPARQL answers. We also define two additional requirements (called \textit{opt extension} and \textit{strong variable binding}, which generalize to KBs some basic properties of SPARQL answers over plain graphs. We note that these requirements apply to arbitrary DLs, whereas Section 5 focuses instead on specific families of DLs.

If \( q \) is a SPARQL query and \( \mathcal{K} \) a KB, we use \( \text{ans}(q, \mathcal{K}) \) below to denote the answers to \( q \) over \( \mathcal{K} \) under some (underspecified) semantics. This allows us to define properties to be met by such a semantics.

Requirement 1 states that \( \text{ans}(q, \mathcal{K}) \) should coincide with certain answers for UCQs.

**Requirement 1** (Certain answer compliance). For any UCQ \( q \) and KB \( \mathcal{K} \),
\[
\text{ans}(q, \mathcal{K}) = \text{certAns}(q, \mathcal{K})
\]

Requirement 2 corresponds to the limitation of [1] identified in Section 3. It requires that \( \text{ans}(q, (\emptyset, \mathcal{A})) \) coincide with answers over \( \mathcal{A} \), as defined in the SPARQL specification.

**Requirement 2** (SPARQL answer compliance). For any query \( q \) and ABox \( \mathcal{A} \),
\[
\text{ans}(q, (\emptyset, \mathcal{A})) = \text{sparqlAns}(q, \mathcal{A})
\]

As will be seen in the next section, it is easy to define semantics that verify Requirements 1 and 2, but fail to comply with basic properties of SPARQL answers over a plain graph. This is why we define additional requirements.

First, as observed in [17] for instance, the \texttt{opt} operator of SPARQL was introduced to “not reject the solutions because some part of the query pattern does not match” [14]. Or in other words, for each answer \( \omega \) to the left operand of an \texttt{opt}, either \( \omega \) or some extension of \( \omega \) is expected be present in the answers to the whole expression. Let \( \Omega_1 \preceq_\omega \Omega_2 \) iff, for each \( \omega_1 \in \Omega_1 \), there is a \( \omega_2 \in \Omega_2 \) s.t. \( \omega_1 \preceq \omega_2 \). Then this property is expressed with Requirement 3.

**Requirement 3** (\texttt{opt} extension). For any queries \( q_1, q_2 \) and KB \( \mathcal{K} \):
\[
\text{ans}(q_1, \mathcal{K}) \preceq_\omega \text{ans}(q_1 \texttt{ opt} q_2, \mathcal{K})
\]

Another important property of SPARQL answers over plain graphs pertains to bound variables. Indeed, a SPARQL query \( q \) (with \texttt{union} and/or \texttt{opt}) may allow partial solution mappings, i.e. whose domain does not cover all variables projected by \( q \). For instance, in Example 2, \( \omega = \{ x \mapsto \texttt{Alice} \} \in \text{sparqlAns}(q, \mathcal{A}) \), even though the variables projected by \( q \) are \( x \) and \( y \). In such a case, we say that variable \( x \) is bound by \( \omega \), whereas variable \( y \) is not. Then a SPARQL query may only admit answers that bind certain sets of variables. For instance the query \( A(x) \texttt{opt} R(x, y) \) admits answers that bind either \( \{ x \} \) or \( \{ x, y, z \} \). But it does not admit answers that bind another set of variables \( \{ y \}, \{ x, y \}, \text{etc.} \). So a natural requirement when generalizing SPARQL answers to KBs is to respect such constraints. We say that a set \( X \) of variables is admissible for a query \( q \) iff there exists a graph \( \mathcal{A} \) and solution mapping \( \omega \) s.t. \( \omega \in \text{sparqlAns}(q, \mathcal{A}) \) and \( \text{dom}(\omega) = X \). Unfortunately, for queries with \texttt{opt}, whether a given set of variables is admissible for a given query is undecidable. So we adopt instead a relaxed notion of admissible bindings. For a SUJO query \( q \), we use \( \text{adm}(q) \) to denote the family of sets of variables defined inductively as follows:

**Definition 4** (Definition of \( \text{adm}(q) \) for the SUJO fragment).
\[
\begin{align*}
\text{adm}(t) &= \{ \text{vars}(t) \} \\
\text{adm}(\texttt{select}_X q) &= \{ X \cap X' \mid X' \in \text{adm}(q) \} \\
\text{adm}(q_1 \texttt{ join} q_2) &= \{ X_1 \cup X_2 \mid (X_1, X_2) \in \text{adm}(q_1) \times \text{adm}(q_2) \} \\
\text{adm}(q_1 \texttt{ opt} q_2) &= \text{adm}(q_1) \cup \text{adm}(q_1 \texttt{ join} q_2) \\
\text{adm}(q_1 \texttt{ union} q_2) &= \text{adm}(q_1) \cup \text{adm}(q_2)
\end{align*}
\]

We can now formulate the corresponding property, as follows:

**Property 1** (Variable binding). For any SUJO query \( q \), KB \( \mathcal{K} \) and \( \omega \in \text{ans}(q, \mathcal{K}) \):
\[
\text{dom}(\omega) \in \text{adm}(q)
\]

This constraint on variable bindings is still arguably weak though, if one consider queries with \texttt{union}. Take for instance the query \( q = A(x) \texttt{union} R(x, y) \). Then \( \text{adm}(q) = \{ \{ x \}, \{ x, y \} \} \). But the semantics of SPARQL over plain graphs puts a stronger requirement on variable bindings. If \( \omega \) is a solution to \( q \), then \( \omega \) may bind \( \{ x \} \) only if \( \omega \) is an answer to the left operand \( A(x) \), and \( \omega \) may bind \( \{ x, y \} \) only if \( \omega \) is an answer to the right operand \( R(x, y) \). It is immediate to see that Property 1 on variable bindings does not enforce this property. So we formulate a stronger requirement:

**Requirement 4** (Strong variable binding). For any SUJO queries \( q_1, q_2 \), KB \( \mathcal{K} \) and solution mapping \( \omega \):
\[
\begin{align*}
\text{if } \omega \in \text{ans}(q_1 \texttt{ union} q_2, \mathcal{K}) \text{ and } \omega \notin \text{ans}(q_2, \mathcal{K}), \text{ then } \text{dom}(\omega) \in \text{adm}(q_1) \\
\text{if } \omega \in \text{ans}(q_1 \texttt{ union} q_2, \mathcal{K}) \text{ and } \omega \notin \text{ans}(q_1, \mathcal{K}), \text{ then } \text{dom}(\omega) \in \text{adm}(q_2)
\end{align*}
\]
Observe that Requirement 4 is stronger than Property 1. To see this, by contraposition, let us assume a semantics that violates Property 1. Then there is a query \( q \), \( KB \), \( K \) and solution mapping \( \omega \) s.t. \( \omega \in \text{ans}(q_1, K) \) and \( \text{dom}(\omega) \notin \text{adm}(q_1) \). Now let \( q_2 \) be a SUJO query whose signature is disjoint with the signature of \( K \) (i.e. none of the predicates in \( q_2 \) appears in \( K \)). Then \( \omega \notin \text{ans}(q_2, K) = \emptyset \). Therefore, \( q_1, q_2, K \) and \( \omega \) violate (the first condition of) Requirement 4.

### 5. Semantics

We now investigate different semantics for answering SPARQL queries over a KB, in view of the requirements expressed in the previous section. We note that each semantics is defined for a specific fragment of SPARQL only, and that this is also the case of Requirements 1 and 4 (the other two requirements are defined for arbitrary SPARQL queries). So when we say below that a semantics defined for fragment \( L_1 \) satisfies a requirement defined for fragment \( L_2 \), this means that the requirement holds for the fragment \( L_1 \cap L_2 \).

Section 5.1 shows that adopting a compositional interpretation of certain answers, analogous to SPARQL entailment regimes (restricted to SUJO queries), is sufficient to satisfy Requirement 2, but fails to satisfy Requirement 1 for the SJ and U fragments already. Then Section 5.2 focuses on DLs with the canonical model property, and investigates semantics with a “procedural” flavor. In particular, we consider generalizing a well-known property of certain answers to UCQs: they are equivalent to answers over the canonical model, but restricted to those that range over the active domain of the KB. We show that this solution satisfies Requirements 1 and 2 for the SUJO fragment, but fails to satisfy Requirement 3 for the O fragment already. Next, Section 5.3 builds upon this last observation, and shows that it is possible to define a semantics, called maximal admissible canonical answers, that satisfies all requirements for the SUJO fragment. Finally, Section 5.4 proposes a more declarative semantics, called epistemic certain answers, considering the (standard) encoding of \( \text{opt} \) with \( \text{union} \) and \( \text{minus} \) operator. It consists in evaluating the right operand of \( \text{minus} \) under certain answer semantics, in an inductive fashion. We show that although this semantics also satisfies all the requirements, it differs from maximal admissible canonical answers.

Table 1 summarizes our observations (for KBs with the canonical model property only), together with observations about the proposal made in [1] (discussed in Section 3).

#### 5.1. SPARQL entailment regimes

Example 2 above showed that certain answer to a query with \( \text{opt} \) may fail to comply with the standard compositional semantics of SPARQL (Definition 1) over a plain graph (i.e. when the TBox is empty). Then a natural attempt to conciliate the two is to proceed “the other way around”: stick to the compositional semantics of SPARQL, and use certain answers for the base case only. This is in essence what the SPARQL entailment regimes propose for queries that correspond to the SUJO fragment (recall the restrictions on reserved \( \text{iri} \)/\( \text{rdfs} \)/\( \text{owl} \) keywords in triple patterns expressed in Section 2).

Because the specification of SPARQL entailment regimes [12] is too low-level for the scope of this paper, we provide a more abstract characterization of this approach for the SUJO fragment (using certain answer semantics as the base case). If \( q \) is a query and \( K \) a KB, we call the resulting set of solution mapping the **entailment regime answers** to \( q \) over \( K \), denoted with \( \text{eRAns}(q, K) \), defined as follows:

**Definition 5 (Entailment regime answers).**

\[
\begin{align*}
\text{eRAns}(t, K) &= \text{certAns}(t, K) \\
\text{eRAns}(q_1 \ \text{UNION} \ q_2, K) &= \text{eRAns}(q_1, K) \cup \text{eRAns}(q_2, K) \\
\text{eRAns}(q_1 \ \text{JOIN} \ q_2, K) &= \pi_X \text{eRAns}(q_1, K) \\
\text{eRAns}(q_1 \ \text{OPT} \ q_2, K) &= (\text{eRAns}(q_1, K) \setminus \text{eRAns}(q_2, K)) \cup (\text{eRAns}(q_1, K) \setminus \text{eRAns}(q_2, K)) \\
\text{eRAns}(\text{SELECT}_X q, K) &= \pi_X \text{eRAns}(q, K)
\end{align*}
\]

It is immediate to see that entailment regime answers and SPARQL answers coincide over a plain graph. Indeed, in the base case (i.e. when \( q \) is a triple pattern), for any graph \( A \), \( \text{sparqlAns}(q, A) = \text{certAns}(q, \langle \emptyset, A \rangle) \). Then the inductive definitions of \( \text{sparqlAns}(q, A) \) (Definition 1) and \( \text{eRAns}(q, K) \) (Definition 5) coincide. So entailment regime answers satisfy Requirement 2.

But they fail to comply with certain answers for UCQs (Requirement 1), for two reasons. First, the \( \text{UNION} \) operator is not compositional for certain answers in some DLs. Consider for instance Example 4 below:

---

*Table 1*

<table>
<thead>
<tr>
<th>Semantics</th>
<th>Fragment</th>
<th>REQ1</th>
<th>REQ2</th>
<th>REQ3</th>
<th>REQ4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahmetaj et al. ([11])</td>
<td>pwdPT (⊆ SUJO)</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>✓</td>
</tr>
<tr>
<td>Entailment regime (Definition 5)</td>
<td>UJO</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>SJ / SUJO</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Canonical (Definition 6)</td>
<td>O / SUJO</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Restricted (Definition 7)</td>
<td>SUJO</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Maximal admissible canonical answers (Definition 10)</td>
<td>SUJO</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Epistemic certain answers (Definition 12)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 4.
\[ A = \{\text{Driver}(Alice)\} \]
\[ T = \{\text{Driver} \cup \text{TruckDriver}\} \]
\[ q = \text{CarDriver}(x) \cup \text{TruckDriver}(x) \]
Then \( \text{certAns}(q, (T, A)) = \{\{x \mapsto \text{Alice}\}\} \), but \( \text{eRAns}(q, (T, A)) = \emptyset \).

Second, the select operator is not compositional for certain answers, even for some DLs that have the canonical model property. Consider for instance Example 5 below:

Example 5.
\[ A = \{\text{Driver}(Alice)\} \]
\[ T = \{\text{Driver} \cup \exists \text{hasLicense}\} \]
\[ q = \text{SELECT}(x) \text{(Driver}(x) \cup \text{hasLicense}(x, y)) \]
Then \( \text{certAns}(q, (T, A)) = \{\{x \mapsto \text{Alice}\}\} \), but \( \text{eRAns}(q, (T, A)) = \emptyset \).

So entailment regime answers fail to satisfy Requirement 1 for the U and SJ fragments already.

5.2. Canonical answers

We now focus on DLs with the canonical model property. We assume some underspecified DL, \( \mathcal{L}_{\text{can}} \), with the canonical model property, and use \( \text{an } \mathcal{L}_{\text{can}} \text{ KB} \) to refer to a KB in such DL. Then if \( \mathcal{K} \) is an \( \mathcal{L}_{\text{can}} \text{ KB} \), we use \( \text{can}(\mathcal{K}) \) to designate one of its canonical models.

An equivalent definition of certain answers for DLs with the canonical model property is the following: certain answers to a UCQ \( q \) over a KB \( \mathcal{K} \) coincide with answers to \( q \) over \( \text{can}(\mathcal{K}) \), restricted to those that range over \( a\text{Dom}(\mathcal{K}) \). We show that extending this definition to queries with \( \text{opt} \) is sufficient to satisfy Requirement 2 (in addition to Requirement 1), but fails to satisfy Requirement 3.

If \( \Omega \) is a set of solution mappings and \( B \subseteq N_1 \), let \( \Omega \Rightarrow B = \{\omega \in \Omega \mid \text{range}(\omega) \subseteq B\} \). Then we define the canonical answers to a query \( q \) over an \( \mathcal{L}_{\text{can}} \text{ KB} \( \mathcal{K} \), denoted with \( \text{canAns}(q, \mathcal{K}) \), as follows:

Definition 6 (Canonical answers). For any SUJO query \( q \) and \( \mathcal{L}_{\text{can}} \text{ KB} \( \mathcal{K} \):

\[ \text{canAns}(q, \mathcal{K}) = \text{sparqlAns}(q, \text{can}(\mathcal{K})) \Rightarrow a\text{Dom}(\mathcal{K}) \]

Proposition 1 states that canonical answers coincide with \( \text{sparqlAns} \) answers over a plain graph, i.e. satisfy Requirement 2 (the proof is in Appendix A).

Proposition 1. For any SUJO query \( q \) and \( \mathcal{L}_{\text{can}} \text{ KB} \( \mathcal{K} \), \( \text{canAns}(q, \mathcal{K}) \) satisfies Requirement 2.

From the observation made above, canonical answers also comply with certain answers for UCQs (Requirement 1). But they fail to satisfy \( \text{opt} \) extension (Requirement 3), as illustrated with Example 6.

Example 6.
\[ A = \{\text{Driver}(Alice)\} \]
\[ T = \{\text{Driver} \cup \exists \text{hasLicense}\} \]
\[ q = \text{Driver}(x) \text{ OPT hasLicense}(x, y) \]

In this example, let \( \mathcal{K} = (T, A) \). Then \( \text{canAns}(\text{Driver}(x), \mathcal{K}) = \{\{x \mapsto \text{Alice}\}\} \). However, \( \text{sparqlAns}(q, \text{can}(\mathcal{K})) = \{\{x \mapsto \text{Alice}, y \mapsto e\}\} \), for some \( e \notin a\text{Dom}(\mathcal{K}) \). Therefore \( \text{canAns}(q, \mathcal{K}) = \text{sparqlAns}(q, \text{can}(\mathcal{K})) \Rightarrow a\text{Dom}(\mathcal{K}) = \emptyset \). So \( \text{canAns}(\text{Driver}(x), \mathcal{K}) \not\Rightarrow q \text{ canAns}(q, \mathcal{K}) \), which violates Requirement 3.

5.3. Maximal admissible canonical answers

The canonical answers defined in the previous section fail to satisfy Requirement 3. We show how this definition can be adapted to satisfy all requirements, for the whole SUJO fragment.

Intuitively, in Definition 6, the restriction of \( \text{sparqlAns}(q, \text{can}(\mathcal{K})) \) to solution mappings that range over \( \text{can}(\mathcal{K}) \) is too strong. Consider again Example 6, where \( \text{sparqlAns}(q, \text{can}(\mathcal{K})) = \{\{x \mapsto \text{Alice}, y \mapsto e\}\} \). In this example, rather than filtering out this solution mapping (because it does not range over \( a\text{Dom}(\mathcal{K}) \)), one would like instead to restrict it to the active domain, which yields the desired mapping \( \{x \mapsto \text{Alice}\} \).

To formalize this intuition, if \( \Omega \) is a set of solution mappings and \( B \subseteq N_1 \), let \( \Omega \Rightarrow B = \{\omega \mid \omega \in \Omega \} \). We can now define the restricted canonical answers \( \text{restCanAns}(q, \mathcal{K}) \) to a query \( q \) over an \( \mathcal{L}_{\text{can}} \text{ KB} \( \mathcal{K} \), as follows:

Definition 7 (Restricted canonical answers). For any SUJO query \( q \) and \( \mathcal{L}_{\text{can}} \text{ KB} \( \mathcal{K} \):

\[ \text{restCanAns}(q, \mathcal{K}) = \text{sparqlAns}(q, \text{can}(\mathcal{K})) \Rightarrow a\text{Dom}(\mathcal{K}) \]

However, restricted canonical answers still fail to satisfy the above requirement on admissible variable bindings (Requirement 4), as illustrated with Example 7 below:

Example 7.
**Example 7.**

\[ A = \{ \text{Teacher (Alice)} \} \]
\[ T = \{ \text{Teacher} \models \exists \text{teachesTo, teachesTo} \models \text{hasTeacher}^{-} \} \]
\[ q = \text{Teacher}(X) \ 	ext{OPT} \ (\text{teachesTo}(X,y) \ 	ext{JOIN} \ 	ext{hasTeacher}(y,z)) \]

In this example, \( \text{sparqlAns}(q, \text{can}(K)) = \{ [x \mapsto \text{Alice}, y \mapsto e, z \mapsto \text{Alice}] \} \), for some \( e \in \text{aDom}(K) \). So restricting this solution mapping to \( \text{aDom}(K) \) would yield the mapping \( [x \mapsto \text{Alice}, z \mapsto \text{Alice}] \). However, \( [x, z] \) is not an admissible set of variables for \( q \), because \( q \) requires that whenever variable \( z \) is bound, variable \( y \) must be bound as well.

We now propose to further constrain restricted canonical answers in order to satisfy Requirement 4. We call the resulting solution mappings \emph{maximal admissible} canonical answers, noted \( \text{mCanAns}(q, K) \).

We start with the SJO fragment (i.e. queries without \text{union}) for simplicity, since for this fragment, in order to satisfy Requirement 4, it is sufficient to satisfy Property 1. If \( S \) is a family of sets, let \( \max_{\subseteq}(S) \) designate the set of maximal elements of \( S \) w.r.t. set inclusion. And if \( \Omega \) is a set of solution mappings and \( \chi \) a family of sets of variables, let:

\[ \Omega \otimes \chi = \{ \omega | \chi \mid \omega \in \Omega, \chi \in \max_{\subseteq}(\chi' \cap 2^{\text{dom}(\omega)}) \} \]

We can now define maximal admissible canonical answers for the SJO fragment, as follows:

**Definition 8 (Maximal admissible canonical answers (SJO)).**

\[ \text{mCanAns}(q, K) = \text{restCanAns}(q, K) \otimes \text{adm}(q) \]

This definition, if extended to SUJO queries, is sufficient to satisfy Property 1 (see Lemma 7 in Appendix B), but not Requirement 4. To see this, consider Example 8 below, which is almost identical to Example 7:

**Example 8.**

\[ A = \{ \text{Teacher (Alice)} \} \]
\[ T = \{ \text{Teacher} \models \exists \text{teachesTo, teachesTo} \models \text{hasTeacher}^{-} \} \]
\[ q = \text{Teacher}(X) \ 	ext{OPT} \ (\text{teachesTo}(X,y) \ 	ext{JOIN} \ 	ext{hasTeacher}(y,z)) \]

As opposed to Example 7, \( [x, z] \) is now an admissible set of variables for \( q \) (i.e. \( [x, z] \in \text{adm}(q) \)), due to the right operand \( \text{hasLicense}\) of the \text{union} operator. As a result, the undesired mapping \( [x \mapsto \text{Alice}, z \mapsto \text{Alice}] \) is now in \( \text{restCanAns}(q, K) \otimes \text{adm}(q) \), even though this mapping is returned by the left operand of the \text{union}, for which \( [x, z] \) is not admissible.

So in order to generalize Definition 8 to queries with \text{union} while satisfying Requirement 4, the provenance of each solution mapping needs to be taken into account. To this end, we define the set of \emph{branches} of a SUJO query \( q \), denoted with \( \text{branches}(q) \), as the set of SJO queries that may produce a solution to \( q \), by intuitively “choosing” one operand of each \text{union}. For instance, if \( q = \text{A}(X) \ 	ext{OPT} \ R_1(X, y) \ 	ext{UNION} \ R_2(X, z) \), then \( \text{branches}(q) = \{ \text{A}(X) \ 	ext{OPT} \ R_1(X, y), \text{A}(X) \ 	ext{OPT} \ R_2(X, z) \} \). The function \( \text{branches}(q) \) is defined inductively over \( q \) as expected:

**Definition 9 (Branches of a SUJO query).**

\[
\begin{align*}
\text{branches}(t) & = \{ t \} \\
\text{branches}(\text{SELECT}_X q) & = \{ \text{SELECT}_X q' \mid q' \in \text{branches}(q) \} \\
\text{branches}(q_1 \text{ JOIN } q_2) & = \{ q_1' \text{ JOIN } q_2' \mid (q_1', q_2') \in \text{branches}(q_1) \times \text{branches}(q_2) \} \\
\text{branches}(q_1 \text{ OPT } q_2) & = \{ q_1' \text{ OPT } q_2' \mid (q_1', q_2') \in \text{branches}(q_1) \times \text{branches}(q_2) \} \\
\text{branches}(q_1 \text{ UNION } q_2) & = \text{branches}(q_1) \cup \text{branches}(q_2)
\end{align*}
\]

According to the semantics of \text{sparql} over plain graphs, an answer to a SUJO query \( q \) must be an answer to some branch of \( q \) (the converse does not hold though; see e.g. [25, Example 1]). Or formally, for any SUJO query \( q \) and graph \( A \):

\[
\text{sparqlAns}(q, A) \subseteq \bigcup_{q' \in \text{branches}(q)} \text{sparqlAns}(q', A)
\]

So if \( q' \in \text{branches}(q) \), we use \( \text{sparqlAns}(q, A, q') \) to denote the answers to \( q \) over \( A \) that may be obtained by evaluating branch \( q' \), i.e.:

\[
\text{sparqlAns}(q, A, q') = \text{sparqlAns}(q, A) \cap \text{sparqlAns}(q', A)
\]

Similarly, we adapt Definition 8 to a branch \( q' \) of \( q \):

\[
\text{mCanAns}(q, K, q') = (\text{sparqlAns}(q, \text{can}(K)), q') \otimes \text{adm}(q')
\]

We can now generalize maximal admissible canonical answers to the SUJO fragment:

**Definition 10 (Maximal admissible canonical answers (SUJO)).**

\[
\text{mCanAns}(q, K) = \bigcup_{q' \in \text{branches}(q)} \text{mCanAns}(q, K, q')
\]
Proposition 2 shows that maximal admissible canonical answers satisfy all requirements expressed in Section 4 (the proof is in Appendix B).

**Proposition 2.** For any SUJO query \( q \) and \( \mathcal{L}_{\text{car}}, KB K, m\text{CanAns}(q, K) \) satisfies Requirements 1, 2, 3 and 4.

### 5.4. Epistemic certain answers

In this section, we propose a more declarative semantics, which we call **epistemic certain answers**.

In Definition 1, the evaluation of the \( \text{opt} \) operator over a graph \( \mathcal{A} \) is defined by means of the "\( \setminus \)" operator between sets of solution mappings. So in a sense, the \( \text{opt} \) operator expresses a form of negation. Intuitively, epistemic certain answers consist in treating it like negation as failure.

We make negation explicit in the definition of this semantics by extending the SUJO fragment with a difference operator, denoted \( \setminus \) in what follows. And we will use the letter "\( M \)" (in addition to \( S, U, J, \) and \( O \)) to designate queries with \( \setminus \) operator. We will also use an additional base case: as an alternative to a triple pattern, we allow a (finite) set \( \Omega \) of solution mappings, similar to the \( \text{VALUES} \) operator of SPARQL. Hence we use the letter "\( V \)" to designate the corresponding queries.

So the syntax of SUJOMV queries is:

\[
q := \Omega \mid t \mid \text{SELECT} q \mid q \text{ UNION } q \mid q \text{ JOIN } q \mid q \text{ OPT } q \mid q \setminus \Omega \setminus q
\]

Let us first extend Definition 1 with the evaluation of these two operators:

**Definition 11.** If \( q_1, q_2 \) are SUJO queries, \( \Omega \) a (finite) set of solution mappings and \( \mathcal{A} \) a graph, then:

\[
\text{sparqlAns}(\Omega, \mathcal{A}) = \Omega
\]

\[
\text{sparqlAns}(q_1, \setminus q_2, \setminus \mathcal{A}) = \text{sparqlAns}(q_1, \setminus \mathcal{A}) \cup \text{sparqlAns}(q_2, \setminus \mathcal{A})
\]

Then a SUJO query can be transformed into an equivalent SUJMV query, by replacing inductively each "\( q_1 \text{ OPT } q_2 \)" with "\( (q_1 \text{ JOIN } q_2) \text{ UNION } (q_1 \setminus q_2) \setminus \)." "Equivalent" here means that the answers to both queries over a plain graph coincide.

The semantics that we propose amounts to evaluating the right operand of \( \setminus \) under certain answer semantics: if \( q = q_1 \setminus q_2 \) is a (sub)-query, \( K \) a KB and \( \mathcal{I} \) a model of \( K \), then the answers to \( q \) over \( \mathcal{I} \) under this semantics are the answers to \( q_1 \) over \( \mathcal{I} \), minus the certain answers to \( q_2 \) over \( K \). This intuition of is formalized below:

**Definition 12 (Epistemic certain answers).** If \( q \) is a SUJO query and \( K \) a KB, then

\[
\text{epCertAns}(q, K) = \text{certAns}(\text{partEval}(q, K), K)
\]

where \( \text{partEval}(q, K) \) is the SUJMV query defined inductively as follows:

\[
\begin{align*}
\text{partEval}(t, K) &= t \\
\text{partEval}(\text{SELECT} q, K) &= \text{SELECT} (q) \text{ partEval}(q, K) \\
\text{partEval}(q_1 \text{ JOIN } q_2, K) &= \text{partEval}(q_1, K) \cup \text{partEval}(q_2, K) \\
\text{partEval}(q_1 \text{ UNION } q_2, K) &= \text{partEval}(q_1, K) \cup \text{partEval}(q_2, K) \\
\text{partEval}(q_1 \text{ OPT } q_2, K) &= \text{partEval}(q_1, K) \cup \text{partEval}(q_2, K) \cup \text{partEval}(q_1, K) \setminus \text{partEval}(q_2, K)
\end{align*}
\]

Note that in the definition of \( \text{partEval}(q_1 \text{ OPT } q_2, K) \), \( \text{epCertAns}(q_2, K) \) evaluates to a set of solution mappings, so it corresponds to a subquery of the form \( \Omega \). Note also that the definitions of \( \text{epCertAns}(q, K) \) and \( \text{partEval}(q, K) \) are mutually recursive. For a better understanding of this definition, we illustrate with an example the inductive computation of epistemic certain answers.

**Example 9.**

\[
\begin{align*}
\mathcal{A} &= \{\text{Driver(Alice), Driver(Bob), hasLicense(Bob,b)}\} \\
\mathcal{I} &= \{\text{Driver \in HasLicense}\} \\
q &= \text{Driver(x) OPT hasLicense(x, y)}
\end{align*}
\]

First,

\[
\begin{align*}
\text{epCertAns}(\text{hasLicense}(x, y), K) &= \text{certAns}(\text{partEval}(\text{hasLicense}(x, y), K)) \\
&= \text{certAns}(\text{hasLicense}(x, y), K) = \{(x \mapsto \text{Bob}, y \mapsto b)\}
\end{align*}
\]

Then

\[
\begin{align*}
\text{epCertAns}(q, K) &= \text{certAns}(\text{partEval}(q, K), K) \\
&= \text{certAns}( \\
&\quad \text{partEval}(\text{Driver(x) JOIN hasLicense(x, y), K}) \\
&\quad \text{UNION} \\
&\quad (\text{partEval}(\text{Driver(x), K) MINUS epCertAns(hasLicense(x, y), K)), K)) \\
&= \text{certAns}( \\
&\quad (\text{Driver(x) JOIN hasLicense(x, y)}) \\
&\quad \text{UNION} \\
&\quad (\text{Driver(x) MINUS (\{x \mapsto \text{Bob}, y \mapsto b\}\)), K}) \\
&= \{(x \mapsto \text{Bob}, y \mapsto b), (x \mapsto \text{Alice})\}
\end{align*}
\]
The KB in this example admits a canonical model, so the canonical answers \( \text{canAns}(q, K) \) are defined. But neither \( \text{canAns}(q, K) \) nor \( \text{certAns}(q, K) \) contains the mapping \( \{x \mapsto \text{Alice}\} \). Instead, under \( \text{epCertAns} \), the right operand \( \text{hasLicense}(x, y) \) of the \text{MINUS} operator is first evaluated under certain answer semantics. The result \( \{[x \mapsto \text{Bob}, y \mapsto b]\} \) does not contain any answer that maps \( x \) to \( \text{Alice} \), because variable \( y \) cannot be mapped to the same license for \( \text{Alice} \) in every model of \( K \). This way the mapping \( \{[x \mapsto \text{Alice}]\} \) is not discarded when evaluating the \text{MINUS} operator. Intuitively, this allows distinguishing the “anonymous” license of \( \text{Alice} \) from the “named” license \( b \) of Bob.

Proposition 3 shows that epistemic certain answers satisfy all requirements expressed in Section 4 (the proof is in Appendix C).

**Proposition 3.** For any SUJO query \( q \) and KB \( K \), \( \text{epCertAns}(q, K) \) satisfies Requirements 1, 2, 3 and 4.

**Relationship with existing formalisms.** The treatment of negation in this semantics is often called epistemic, and has been extensively studied in the logic programming literature, among others (see for instance [11,26,9]).

Numerous formal languages and associated semantics have been proposed that can express epistemic negation, usually by means of two different negation operators (“strong” and “weak”), or with an epistemic operator in the language (often noted \( \mathcal{K} \)), thus allowing explicit quantification over models. Interestingly, one of these formalisms, called MKNF [20], has been used in [22] to reason over DLs extended with rules, as a unified way to combine closed and open-world reasoning. In this framework, called MKNF+ , a knowledge base can be viewed as a pair \( (\mathcal{K}, \mathcal{P}) \), where \( \mathcal{K} \) is DL KB and \( \mathcal{P} \) a set of rules, and epistemic certain answers may be captured by means of a \( \mathcal{K} \) operator. For instance, Example 6 can be encoded as a knowledge base \( (\langle T, A \rangle, \mathcal{P}), \) where \( \mathcal{P} \) is the set of rules:

\[
\text{ans}(x, y) \leftarrow -K \ q(x, y) \\
q(x, y) \leftarrow \text{Driver}(x), \text{hasLicense}(x, y) \\
q(x, \text{NULL}) \leftarrow \text{Driver}(x), -K \ \text{hasLicense}(x, y)
\]

This example can be generalized, translating a SUJO query \( q \) into a set of rules (as in [23] for instance), and expressing (possibly nested) certain answers with the \( K \) operator. We did not adopt directly the MKNF+ syntax and semantics though, because such expressivity is not needed in our setting. Instead, we chose to formulate epistemic certain answers in a concise way by means of certain answers. A detailed analysis of the relationship with such formalism is left for future work.

**Proposition 4.** For any \( \mathcal{L}_{\text{can}} \) KB \( K \) and SUJO query \( q \),

\[
\text{epCertAns}(q, K) = \text{sparqlAns} (\text{partEval}(q, K), \text{can}(K)) \triangleright \text{aDom}(K)
\]

This property will be instrumental in comparing \( \text{epCertAns} \) and \( \text{mCanAns} \) (in Section 5.5), but also in investigating the complexity of query answering for \( \text{epCertAns} \) (in Section 6).

5.5. Comparing maximal canonical answers and epistemic certain answers

Both maximal canonical answers (\( \text{mCanAns}, \) defined in Section 5.3) and epistemic certain answers (\( \text{epCertAns}, \) defined in Section 5.4) satisfy the requirements proposed in Section 4. So a natural question is whether (and when) these two semantics coincide.

We first observe that \( \text{mCanAns} \) is only defined for DLs with the canonical model property, whereas \( \text{epCertAns} \) is defined for arbitrary DLs. So we restrict the comparison to DLs where both are defined, i.e. for \( \mathcal{L}_{\text{can}} \) KBs. Proposition 5 shows that the two semantics differ for SUJO queries.

**Proposition 5.** There is a KB \( K \) and SUJO query \( q \) such that \( K \) admits a canonical model, and \( \text{mCanAns}(q, K) \neq \text{epCertAns}(q, K) \).

**Proof.** Consider the KB \( K = (T, A), \) with \( T = \{B \subseteq \exists R\} \) and \( A = \{(a, b), (b, b)\}, \) and the query \( q = (\mathcal{A}(x_1) \ \text{JOIN} \ (B(x_2) \ \text{OPT} \ R(x_2, x_1)\)). \) Then \( \text{mCanAns}(q, K) = \emptyset, \) whereas \( \text{epCertAns}(q, K) = \{[x_1 \mapsto a, x_2 \mapsto b]\}. \)

Next, we show that both semantics coincide for SUJ queries, i.e. without the \text{OPT} operator (the proof is in Appendix E).

**Proposition 6.** For any \( \mathcal{L}_{\text{can}} \) KB \( K \) and SUJ query \( q \),

\[
\text{mCanAns}(q, K) = \text{epCertAns}(q, K)
\]

Interestingly, the \text{sparql} query used in the proof of Proposition 5 is not well-designed. So an interesting open question is whether these two semantics coincide for well-designed SUJO queries. This is left for future work.
6. Complexity

We now provide complexity results for query answering under the semantics defined in Sections 5.3 and 5.4, for different subfragments of the SUJO fragment, and focusing on KBs in DL-Lite\(_R\) [3], a DL tailored for query answering, which corresponds to the owl:2.qi profile. As is conventional, we focus on the decision problem for query answering, i.e. the problems \(\text{EVAL}_{\text{mCanAns}}\) and \(\text{EVAL}_{\text{epCertAns}}\) below.

Table 2 reproduces results for \(\text{EVAL}_{\text{sparqlAns}}\) in several commonly studied fragments that fall within the SUJO fragment. The opt operator has been the focus of a large part of the literature, as \(\text{EVAL}_{\text{sparqlAns}}\) has been shown to be \(\text{PSPACE}\)-complete for the OJ fragment already, in [25]. Particular attention has also been paid to so-called well-designed SJO and JO queries (see [24] for a definition), which have a natural representation as pattern trees [19], with a significant reduction from \(\text{PSPACE}\) to \(\Sigma_2^P\) and \(\text{coNP}\)-completeness respectively. For SJO, we follow [19] and focus on queries where the select operator is terminal, i.e. where it does not appear in the scope of \(\text{JOIN}\) or \(\text{OPT}\). The corresponding fragment is called SJO\(^*\). Finally, another fragment of interest is UJ, for which query answering is already intractable [25], thus contrasting with projection-free UCQs.

So for each fragment, we investigate whether \(\text{EVAL}_{\text{mCanAns}}\) and \(\text{EVAL}_{\text{epCertAns}}\) match the upper bounds for \(\text{EVAL}_{\text{sparqlAns}}\). The results are summarized in Table 2.

### Proposition 7

If \(q\) is a JO query and \(X_1, X_2 \subseteq \text{vars}(q)\), then it can be decided in \(O(|q|^3)\) whether \(X_1 \in \text{max}_{\Sigma_2}(\text{adm}(q) \cap \mathbb{X})\).

**Proof.** (Sketch.) If \(q\) is a JO query, we compute a family base\(\,(q)\) of sets of variables s.t. \(|\text{base}(q)| = O(|q|)\), and s.t. each \(V \in \text{adm}(q)\) is the union of some elements of base\(\,(q)\) and conversely, i.e. \(\text{adm}(q) = \bigcup |S| \mid S \in 2^{\text{base}(q)}\). The family base\(\,(q)\) can be computed inductively as follows:

- If \(q\) is a triple pattern, then base\(\,(q) = \{\text{vars}(q)\}\).
- If \(q = q_1 \text{ JOIN } q_2\), then base\(\,(q) = \{B_1 \cup B_2 \mid B_1 \in \text{min}_{\Sigma_2}(\text{base}(q_1)), B_2 \in \text{base}(q_2)\} \cup \{B_1 \cup B_2 \mid B_1 \in \text{base}(q_1), B_2 \in \text{min}_{\Sigma_2}(\text{base}(q_2))\} \).
- If \(q = q_1 \text{ OPT } q_2\), then base\(\,(q) = \text{base}(q_1) \cup \text{base}(q_1 \text{ JOIN } q_2)\).

### Table 2

Combined complexity of \(\text{EVAL}_{\text{sparqlAns}}, \text{EVAL}_{\text{mCanAns}}\) and \(\text{EVAL}_{\text{epCertAns}}\). “-c” stands for complete, “-h” for hard, and “A/B” for all fragments between A and B.

<table>
<thead>
<tr>
<th>Fragment</th>
<th>(\text{EVAL}_{\text{sparqlAns}})</th>
<th>(\text{EVAL}_{\text{mCanAns}})</th>
<th>(\text{EVAL}_{\text{epCertAns}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>UJ/SUJO</td>
<td>(\text{NP}-\text{c})</td>
<td>(\text{NP}-\text{c})</td>
<td>(\text{NP}-\text{c})</td>
</tr>
<tr>
<td>well-designed JD</td>
<td>(\text{coNP}-\text{c})</td>
<td>(\text{coNP}-\text{c})</td>
<td>(\text{coNP}-\text{h} / \text{in PSPACE})</td>
</tr>
<tr>
<td>well-designed SJO(^*)</td>
<td>(\Sigma_2^P)-c</td>
<td>(\Sigma_2^P)-c</td>
<td>(\Sigma_2^P)-h / \text{in PSPACE})</td>
</tr>
<tr>
<td>OJ/SUJO</td>
<td>(\text{PSPACE-c})</td>
<td>(\text{PSPACE-c})</td>
<td>(\text{PSPACE-c})</td>
</tr>
</tbody>
</table>

We focus on combined complexity, i.e. measured in the size of the whole input (KB and query), and leave data complexity (parameterized either by the size of the query and TBox) as future work. Complexity of \(\text{SPARQL}\) query evaluation over plain graphs has been extensively studied (see [21] for a recent overview). When these results are tight, they provide us immediate lower bounds. Indeed, from Propositions 1 and 3, maximal admissible canonical answers and epistemic certain answers satisfy Requirement 2, so \(\text{EVAL}_{\text{mCanAns}}\) and \(\text{EVAL}_{\text{epCertAns}}\) are at least as hard as the problem \(\text{EVAL}_{\text{sparqlAns}}\) below.

**Input:** graph \(\mathcal{G}\), query \(q\), mapping \(\omega\)**

**Decide:** \(\omega \in \text{mCanAns}(q, \mathcal{G})\)

**Input:** DL-Lite\(_R\) KB \(\mathcal{K}\), query \(q\), mapping \(\omega\)**

**Decide:** \(\omega \in \text{epCertAns}(q, \mathcal{K})\)

**Input:**  graph \(\mathcal{A}\), query \(q\), mapping \(\omega\)**

**Decide:** \(\omega \in \text{sparqlAns}(q, \mathcal{A})\)**
The induction guarantees that $\left| \min_{\omega} (\text{base}(q)) \right| = 1$, so that $|\text{base}(q)| = O(|q|)$ must hold. Then in order to decide $X_1 \in \max_{\omega} (\text{adm}(q) \cap 2^X)$, it is sufficient to: \(i\) check whether $X_1 \in \text{adm}(q)$, i.e. check whether $X_1 \subseteq \bigcup \{B \in \text{base}(q) \mid B \subseteq X_1\}$, and \(ii\) check whether there is an $X' \in \text{adm}(q) \cap 2^X$ s.t. $X \subseteq X'$. This is the case if and only if there is a $B \in \text{base}(q)$ s.t. $X_1 \subseteq X_1 \cup B \subseteq X_2$. □

We note that from the definition of $\text{adm}(q)$, this property is independent from the semantics under investigation, so it holds for $\text{SPARQL}$ over a plain graph. It also follows that deciding whether $X \in \text{adm}(q)$ for an arbitrary $X$ and $\text{JO}$ query $q$ is tractable (consider the case where $X_1 = X_2$). Interestingly, this does not hold for the $\text{UJ}$ fragment already. Indeed, immediately from the reduction used in [25] for hardness of $\text{EVAL}_{\text{SPARQLAns}}$ in this fragment, deciding $X \in \text{adm}(q)$ for any $X$ and $\text{UJ}$ query $q$ is NP-hard.

We now sketch the argument used to derive upper bounds for the $\text{SUJO}$, well-designed $\text{SJO}^*$ and $\text{UJ}$ fragments. For simplicity, we focus on the well-designed $\text{SJO}^*$ fragment. The argument for queries with unions is similar, but with additional technicalities, because the definition of maximal admissible canonical answers in this case is more involved (compare Definitions 8 and 10 above). We also simplify the explanation by assuming that the Galiffin graph of the query is connected. If $\mathcal{G}$ is a graph, we will use $V(\mathcal{G})$ below to designate its vertices.

From the definition of $\text{EVAL}_{\text{CanAns}}$, $(\mathcal{K}, q, \omega)$ is a positive instance iff $\omega \in \text{mCanAns}(\mathcal{K}, q)$, i.e. iff there is an $\omega'$ s.t. (\(i\)) $\omega = \omega'|X$ for some $X \in \text{adm}(\mathcal{q}) \cap 2^{\text{dom}(\omega) \setminus \text{aDom}(X)}$ and (\(ii\)) $\omega' \in \text{sparqlAns}(q, \mathcal{K})$.

So a (non-deterministic) procedure to verify $\omega \in \text{mCanAns}(q, \mathcal{K})$ consists in guessing an extension $\omega'$ or $\omega$, then verify (\(i\)), and then verify (\(ii\)). From Proposition 7 above, (\(i\)) can be verified in $O(|q|^5)$. For (\(ii\)), if $\omega' \in \text{sparqlAns}(q, \mathcal{K})$, from well-known properties of $\text{can}(\mathcal{K})$ for $\text{DL-Lite}_\mathcal{C}$, it can be shown that:

- there must exist a subgraph $\mathcal{G}$ of $\text{can}(\mathcal{K})$ s.t. $V(\mathcal{G}) \cap V(A) \neq \emptyset$, and the size of the subgraph of $\mathcal{G}$ induced by $V(\mathcal{G}) \setminus V(A)$ is linearly bounded by $\max(|q|, |T|)$;
- for each maximal connected subgraph $\mathcal{G}'$ of $\mathcal{G}$ s.t. $V(\mathcal{G}') \cap V(A) = \emptyset$, it can be verified in $O((|\mathcal{G}'| + |T|) \cdot |T|)$ whether $\mathcal{G}'$ is a subgraph of $\text{can}(\mathcal{K})$.

So in order to verify (\(ii\)), it is sufficient to guess $\mathcal{G}$, then verify that $\mathcal{G}$ is a subgraph of $\text{can}(\mathcal{K})$, and then verify that $\omega' \in \text{sparqlAns}(q, \mathcal{G})$. Since $\text{EVAL}_{\text{SPARQLAns}}$ is in $\Sigma^P_2$, $\omega' \in \text{sparqlAns}(q, \mathcal{G})$ can be non-deterministically verified in time in $O(|q| + |\mathcal{G}| + |\omega'|) = O(|q| + |\mathcal{K}| + |\omega|)$ by some algorithm with an oracle for $\text{CNP}$ problems. And a witness for this algorithm can be guessed together with $\mathcal{G}$ and $\omega'$ (without gaining a level in the polynomial hierarchy). We note that this last remark does not apply to the well-designed $\text{JO}$ fragment: since $\text{EVAL}_{\text{CanAns}}$ is $\text{CNP}$-hard, such a procedure would instead imply a quantifier alternation.

The proof of $\text{CanAns}$-membership for the well-designed $\text{JO}$ fragment is significantly simpler. First, because the fragment does not allow projection, for any $\text{JO}$ query $q$, $\text{mCanAns}(q, \mathcal{K}) = \text{canAns}(q, \mathcal{K})$ must hold. Then we consider the $\text{ABox}$ $\mathcal{A}'$ that contains all atoms over the active domain that are entailed by $\mathcal{K}$, i.e. $\mathcal{A}' = \{ A(c) \in \text{can}(\mathcal{K}) \mid c \in \text{aDom}(\mathcal{K}) \} \cup \{ \{c_1, c_2\} \in \text{can}(\mathcal{K}) \mid c_1, c_2 \in \text{aDom}(\mathcal{K}) \}$. $\mathcal{A}'$ can be computed in time polynomial in $\mathcal{K}$ and, by immediate induction on $q$, it can be shown that $\text{canAns}(q, \mathcal{K}) = \text{sparqlAns}(q, \mathcal{A}')$. Finally, from [24], $\omega \in \text{sparqlAns}(q, \mathcal{A}')$ is in $\text{CNP}$.

6.2. Complexity of epistemic certain answers

For $\text{EVAL}_{\text{CertAns}}$, we only provide two results, namely $\text{NP}$-membership for the $\text{SUJO}$ fragment, and $\text{PSPACE}$-membership for the $\text{SUJO}$ fragment. These two results match the corresponding known upper bounds for $\text{EVAL}_{\text{SPARQLAns}}$, i.e. for query evaluation over a plain graph. For the other fragments, the question remains open (meaning that the known upper bounds for $\text{EVAL}_{\text{SPARQLAns}}$ may also be matched).

NP-membership for the $\text{SUJO}$ fragment follows immediately from the upper bound for $\text{EVAL}_{\text{CanAns}}$ and from the fact that both semantics coincide over $\mathcal{L}_{\text{can}}$ $\text{KBs}$ for this fragment (Proposition 6).

The proof of $\text{PSPACE}$-membership for the $\text{SUJO}$ fragment can be found in Appendix C. First, since $\text{DL-Lite}_\mathcal{R}$ has the canonical model property, from Lemma 4, $\omega \in \text{epCertAns}(q, \mathcal{K})$ can be decided by checking whether $\omega$ ranges over the active domain on the one hand, and deciding whether $\omega$ is an answer to partEval($q, \omega$) over $\text{canAns}(\mathcal{K})$ on the other hand. Then we define a top-down procedure that decides $\omega \in \text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \text{canAns}(\mathcal{K}))$, and requires an amount of space at most polynomial in $|\mathcal{K}| + |q| + |\omega|$.

7. Conclusion and perspectives

We identified in this article simple properties to be met by a semantics meant to conciliate certain answers to UCQs over a $\text{KB}$ on the one hand, and $\text{SPARQL}$ answers over a plain $\text{KB}$ on the other hand. We formalized these properties as requirements, and evaluated different proposals (some of which were taken from the literature) against these requirements.

We also showed that these requirements can be all satisfied (under set semantics) for the fragment of $\text{SPARQL}$ with $\text{SELECT}$, $\text{UNION}$, $\text{JOIN}$ and $\text{OPT}$. More precisely, we defined two semantics that satisfy these requirements, one with a more procedural flavor, defined for $\text{DLs}$ with the canonical model property only, and the other with a more declarative flavor, defined for arbitrary $\text{DLs}$. We also provided combined complexity results for query answering over a $\text{DL-Lite}_\mathcal{R}$ $\text{KB}$ under these semantics, for different (sub-)fragments of $\text{SPARQL}$.

As for future work, data complexity may also be investigated, as well as algorithmic aspects, in particular first-order rewriteability, i.e. the possibility to rewrite a query over a $\text{KB}$ into a query over its $\text{ABox}$ only, which is a key property for $\text{OMQA/ODBA}$ [28,29]. Other fragments of $\text{SPARQL}$ and/or $\text{DLs}$ may also be considered. E.g., after adding the $\text{SPARQL}$ $\text{FILTER}$ operator, we can study the satisfiability problem [22]. Finally, on the practical side, we are interested in implementing these semantics in an $\text{ODBA}$ system like $\text{Ontop}$ [6,30], studying the optimization techniques in a distributed environment [31,27], and evaluating the performance of query answering [18].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
Acknowledgements

This research has been partially supported by the EU H2020 project INODE (863410), by the Italian PRIN project HOPE (2017MMJRE), by the European Regional Development Fund (ERDF) Investment for Growth and Jobs Programme 2014-2020 through the project IDEE (FESR11133), by the Free University of Bozen-Bolzano through the projects KGID and GeoVKG, and by the National Natural Science Foundation of China (61972455).

Appendix A. Proof of Proposition 1

Proposition 1. For any SUJO query \( q \) and \( L_{\text{can}} KB \, K \), canAns\( (q, K) \) satisfies Requirement 2.

Proof. Lemma 1 below states the proposition. \( \Box \)

Lemma 1. For any UCQ \( q \) and ABox \( A \),

\[
\text{canAns}(q, \langle \emptyset, A \rangle) = \text{sparqAns}(q, A)
\]

Proof. If \( A \) is an ABox, then \( \text{aDom}(\langle \emptyset, A \rangle) \) is the set of constants appearing in \( A \).
In addition, \( \text{can}(\langle \emptyset, A \rangle) = A \).
So if \( q \) is a query, trivially, \( \text{sparqAns}(q, \langle \emptyset, A \rangle) \supseteq \text{aDom}(\langle \emptyset, A \rangle) = \text{sparqAns}(q, \langle \emptyset, A \rangle) \).
So from Definition 6, \( \text{canAns}(q, \langle \emptyset, A \rangle) = \text{sparqAns}(q, \langle \emptyset, A \rangle) \). \( \Box \)

Appendix B. Proof of Proposition 2

Proposition 2. For any SUJO query \( q \) and \( L_{\text{can}} KB \, K \), mCanAns\( (q, K) \) satisfies Requirements 1, 2, 3 and 4.

Proof. The proposition is split into Lemmas 2, 4, 5 and 6 below, one for each requirement. \( \Box \)

Lemma 2. For any UCQ \( q \) and \( L_{\text{can}} KB \, K \), mCanAns\( (q, K) \) = certAns\( (q, K) \)

Proof. Let \( q \) be a UCQ and \( K \) an \( L_{\text{can}} KB \).
Lemma 3 below states that mCanAns\( (q, K) \) = canAns\( (q, K) \).
Then the claim follows immediately from the observation (made in Section 5.2) that canAns\( (q, K) \) satisfies Requirement 1. \( \Box \)

Lemma 3. For any UCQ \( q \) and \( L_{\text{can}} KB \, K \), mCanAns\( (q, K) \) = canAns\( (q, K) \)

Proof. Let \( q \) be a UCQ.
Then \( q \) is of the form \( h_1 \cup \ldots \cup h_n \), where each \( h_i \) can only contains SELECT or JOIN operators, and \( \text{vars}(h_i) = \text{vars}(h_j) \) for all \( i, j \in \{1..n\} \).
So immediately from Definition 1, for each \( q' \in \text{branches}(q) \), \( \text{adm}(q') = \{\text{vars}(q)\} \).
Therefore for any \( q' \in \text{branches}(q) \), \( \text{adm}(q') = \{\text{vars}(q)\} \)

Then from the definition of \( \text{mCanAns}(q, K, q') \):

\[
\text{mCanAns}(q, K, q') = (\text{sparqAns}(q, \text{can}(K), q') \supset \text{aDom}(K)) \otimes \text{adm}(q')
\]

\[
= (\text{sparqAns}(q, \text{can}(K), q') \supset \text{aDom}(K)) \otimes \{\text{vars}(q)\}
\]

\[
= \{\omega \mid \text{aDom}(K) \} \cap \{\omega \in \text{sparqAns}(q, \text{can}(K), q') \} \cap \{\text{vars}(q)\}
\]

Then for each \( \omega \in \text{sparqAns}(q, \text{can}(K), q') \), \( \omega \in \text{sparqAns}(q, \text{can}(K)) \) must hold.
So since \( \text{adm}(q) = \{\text{vars}(q)\} \), \( \text{dom}(\omega) = \text{vars}(q) \) must hold as well.
Therefore \( \text{vars}(q) \subseteq \text{dom}(\omega) \subseteq \text{aDom}(K) \), i.e. \( \text{if} \ \text{range}(\omega) \subseteq \text{aDom}(K) \).
So from (B.1):

\[
\text{mCanAns}(q, K, q') = \{\omega \in \text{sparqAns}(q, \text{can}(K), q') \mid \text{range}(\omega) \subseteq \text{aDom}(K)\} \quad (B.1)
\]

\[
\text{mCanAns}(q, K, q') = \text{sparqAns}(q, \text{can}(K), q') \supset \text{aDom}(K) \quad (B.2)
\]

Finally:

\[
\text{mCanAns}(q, K) = \bigcup_{q' \in \text{branches}(q)} \text{mCanAns}(q, K, q') \quad (B.3)
\]

So from (B.2) and (B.3):
\[
m\text{CanAns}(q, \mathcal{K}) = \bigcup_{q' \in \text{branches}(q)} \text{sparqlAns}(q, \text{can}(\mathcal{K}), q') \triangleright \text{aDom}(\mathcal{K})
\]
\[
= \bigcup_{q' \in \text{branches}(q)} (\text{sparqlAns}(q, \text{can}(\mathcal{K})) \cap \text{sparqlAns}(q', \text{can}(\mathcal{K}))) \triangleright \text{aDom}(\mathcal{K})
\]
\[
= \left( \text{sparqlAns}(q, \text{can}(\mathcal{K})) \cap \bigcup_{q' \in \text{branches}(q)} \text{sparqlAns}(q', \text{can}(\mathcal{K})) \right) \triangleright \text{aDom}(\mathcal{K})
\]

And since:
\[
\text{sparqlAns}(q, \mathcal{K}) \subseteq \bigcup_{q' \in \text{branches}(q)} \text{sparqlAns}(q', \text{can}(\mathcal{K}))
\]
we get:
\[
m\text{CanAns}(q, \mathcal{K}) = \text{sparqlAns}(q, \text{can}(\mathcal{K})) \triangleright \text{aDom}(\mathcal{K}) \quad \square
\]

**Lemma 4.** For any SUJO query \(q\) and ABox \(\mathcal{A}\),
\[
m\text{CanAns}(q, \langle \emptyset, \mathcal{A}\rangle) = \text{sparqlAns}(q, \mathcal{A})
\]

**Proof.** If \(\mathcal{A}\) is an ABox, then \(\text{aDom}(\langle \emptyset, \mathcal{A}\rangle)\) is the set of constants appearing in \(\mathcal{A}\).
In addition, \(\text{can}((\emptyset, \mathcal{A})) = \mathcal{A}\).
Then from the definition of \(m\text{CanAns}(q, \mathcal{K}, q'):\)
\[
m\text{CanAns}(q, \langle \emptyset, \mathcal{A}\rangle, q') = (\text{sparqlAns}(q, \mathcal{A}, q') \triangleright \text{aDom}(\langle \emptyset, \mathcal{A}\rangle)) \otimes \text{adm}(q')
\]
\[
= \text{sparqlAns}(q, \mathcal{A}, q') \otimes \text{adm}(q') \quad (B.4)
\]
Then since \(\text{sparqlAns}(q, \mathcal{A}, q') \subseteq \text{sparqlAns}(q', \mathcal{A})\), for each \(\omega \in \text{sparqlAns}(q, \mathcal{A}, q')\), \(\omega \in \text{sparqlAns}(q', \mathcal{A})\) must hold.
So \(\text{dom}(\omega) \in \text{adm}(q')\) must hold as well.
Therefore \(\{\omega|_{X} \mid X \in \text{max}_{\leq} (\text{adm}(q') \cap 2^{\text{dom}(\omega)})\} = \{\text{dom}(\omega)\}\).
So from \((B.4):\)
\[
m\text{CanAns}(q, \langle \emptyset, \mathcal{A}\rangle, q') = \text{sparqlAns}(q, \mathcal{A}, q')
\]
(B.5)
Finally:
\[
m\text{CanAns}(q, \langle \emptyset, \mathcal{A}\rangle) = \bigcup_{q' \in \text{branches}(q)} m\text{CanAns}(q, \langle \emptyset, \mathcal{A}\rangle, q')
\]
(B.6)
So from \((B.5)\) and \((B.6):\)
\[
m\text{CanAns}(q, \langle \emptyset, \mathcal{A}\rangle) = \bigcup_{q' \in \text{branches}(q)} \text{sparqlAns}(q, \mathcal{A}, q')
\]
\[
= \bigcup_{q' \in \text{branches}(q)} (\text{sparqlAns}(q, \mathcal{A}) \cap \text{sparqlAns}(q', \mathcal{A}))
\]
\[
= \text{sparqlAns}(q, \mathcal{A}) \cap \bigcup_{q' \in \text{branches}(q)} \text{sparqlAns}(q', \mathcal{A})
\]
And since:
\[
\text{sparqlAns}(q, \langle \emptyset, \mathcal{A}\rangle) \subseteq \bigcup_{q' \in \text{branches}(q)} \text{sparqlAns}(q', \mathcal{A})
\]
we get:
\[
m\text{CanAns}(q, \langle \emptyset, \mathcal{A}\rangle) = \text{sparqlAns}(q, \mathcal{A}) \quad \square
\]

**Lemma 5.** For any SUJO queries \(q_1, q_2\) and \(\mathcal{L}_{\text{can}}\) KB \(\mathcal{K}\):
\[
m\text{CanAns}(q_1, \mathcal{K}) \preceq_{g} m\text{CanAns}(q_{1 \text{ opt}} q_2, \mathcal{K})
\]
Proof. Let \( q_1, q_2 \) be SUJO queries, let \( K \) be an \( \mathcal{L}_{\text{can}} \) KB, and let \( \omega_1 \in m\text{CanAns}(q_1, K) \).

We need to show that there is an \( \omega_2 \in m\text{CanAns}(q_2, K) \) s.t. \( \omega_1 \leq \omega_2 \).

Since \( \omega_1 \in m\text{CanAns}(q_1, K) \), there must be an SJO query \( q' \in \text{branches}(q_1) \) s.t. \( \omega_1 \in m\text{CanAns}(q_1, K, q') \).

So there is a \( \rho_1 \in \text{sparqlAns}(q_1, \text{can}(K)) \cap \text{sparqlAns}(q', \text{can}(K)) \) and an \( X \in \max_{<}(\text{adm}(q) \cap 2^{\text{dom}(\rho_1)}) \) s.t. \( \omega_1 = \rho_1|_X \).

Since \( \rho_1 \in \text{sparqlAns}(q', \text{can}(K)) \), from Definition 1, there must be a \( \rho_2 \in \text{sparqlAns}(q', \text{opt}(q_2, \text{can}(K))) \) s.t. \( \rho_1 \preceq \rho_2 \).

We first show that \( \rho_2 \in \text{sparqlAns}(q_1, \text{opt}(q_2, \text{can}(K))) \) must hold.

For this, we distinguish two cases:

- \( \rho_1 = \rho_2 \).
  - From Definition 1, for each \( \rho_3 \in \text{sparqlAns}(q_2, \text{can}(K)) \), \( \rho_1 \sim \rho_3 \) must hold.
    - Then because \( \rho_1 \in \text{sparqlAns}(q_1, \text{can}(K)) \), from Definition 1 still, \( \rho_1 = \rho_2 \in \text{sparqlAns}(q_1, \text{opt}(q_2, \text{can}(K))) \) must hold.

- \( \rho_1 \neq \rho_2 \).
  - Because \( \rho_1 \in \text{sparqlAns}(q', \text{can}(K)) \), \( \rho_2 \in \text{sparqlAns}(q', \text{opt}(q_2, \text{can}(K))) \) and \( \rho_1 \preceq \rho_2 \), from Definition 1, there must be a \( \rho_3 \in \text{sparqlAns}(q_2, \text{can}(K)) \) s.t. \( \rho_1 \sim \rho_3 \).
    - So \( \rho_1 \sim \rho_3 \) holds.
    - Then because \( \rho_1 \in \text{sparqlAns}(q_1, \text{can}(K)) \), \( \rho_3 \in \text{sparqlAns}(q_2, \text{can}(K)) \) and \( \rho_1 \sim \rho_3 \), from Definition 1 still, \( \rho_1 \cup \rho_3 = \rho_2 \in \text{sparqlAns}(q_1, \text{opt}(q_2, \text{can}(K))) \) must hold.

Now because \( \rho_1 \preceq \rho_2 \), \( \text{dom}(\rho_1) = X \subseteq \text{dom}(\rho_2) \).

And since \( X \in \text{adm}(q') \cap 2^{\text{dom}(\rho_1)} \) holds, so there must be an \( X' \) s.t. \( X \subseteq X' \) and \( X' \in \max_{<}(\text{adm}(q') \cap 2^{\text{dom}(\rho_2)}) \).

Finally, because \( q' \in \text{branches}(q_1) \), from Definition 9, \( q' \in \text{branches}(q_2) \).

So from Definition 8, \( \rho_2|_X \in m\text{CanAns}(q_1, \text{opt}(q_2, K), K) \).

Now let \( \omega_2 = \rho_2|_X \).

To complete the proof, we only need to show that \( \omega_1 \leq \omega_2 \).

First, since \( \omega_1 = \rho_1|_X \), \( \omega_1 \preceq \rho_1 \) must hold.

Then from the definition of \( \rho_2 \), \( \rho_1 \preceq \rho_2 \).

So from the transitivity of \( \preceq \), \( \omega_1 \leq \rho_2 \).

Finally, since \( X \subseteq X' \), \( \omega_1|_X \leq \omega_2|_X \) must hold, i.e. \( \omega_1 \leq \omega_2 \).

\( \square \)

Lemma 6. For any queries \( q_1, q_2 \), \( \mathcal{L}_{\text{can}} \) KB \( K \) and solution mapping \( \omega \):

\[
\text{if } \omega \in m\text{CanAns}(q_1 \cup q_2) \text{ and } \omega \notin m\text{CanAns}(q_2), \text{ then } \text{dom}(\omega) \in \text{adm}(q_1)
\]

Proof. Let \( \omega \in m\text{CanAns}(q_1 \cup q_2, K) \) s.t. \( \omega \notin m\text{CanAns}(q_2, K) \).

Then from Definition 10, because \( \omega \in m\text{CanAns}(q_1 \cup q_2, K) \):

\[
\omega \in \bigcup_{q' \in \text{branches}(q_1 \cup q_2)} m\text{CanAns}(q_1 \cup q_2, K, q')
\]

And from Definition 9:

\[
\text{branches}(q_1 \cup q_2) = \text{branches}(q_1) \cup \text{branches}(q_2)
\]

So:

\[
\omega \in \bigcup_{q' \in \text{branches}(q_1 \cup q_2) \setminus \text{branches}(q_2)} m\text{CanAns}(q_1 \cup q_2, K, q')
\]

So there is an SJO query \( q' \in \text{branches}(q_1) \cup \text{branches}(q_2) \) s.t. \( \omega \in m\text{CanAns}(q_1 \cup q_2, K, q') \).

So there is an \( \omega' \in \text{sparqlAns}(q_1 \cup q_2, \text{can}(K)) \cap \text{sparqlAns}(q', \text{can}(K)) \) s.t. \( \omega = \omega'|_X \) for some \( X \in \max_{<}(\text{adm}(q') \cap 2^{\text{dom}(\omega')}) \).

Then we can distinguish three cases:

- \( q' \in \text{branches}(q_1) \setminus \text{branches}(q_2) \).
  - Since \( \omega' \in \text{sparqlAns}(q_1 \cup q_2, \text{can}(K)) \) and \( q' \notin \text{branches}(q_2) \), \( \omega' \notin \text{sparqlAns}(q_2, \text{can}(K)) \) must hold.
    - Then because \( \omega' \in \text{sparqlAns}(q_1 \cup q_2, \text{can}(K)) \), from Definition 1, \( \omega' \in \text{sparqlAns}(q_1, \text{can}(K)) \), must hold.
      - So from Definition 8, \( \omega \in m\text{CanAns}(q_1, K, q') \).
      - And since \( q' \in \text{branches}(q_1) \), from Definition 10, \( \omega \in m\text{CanAns}(q_1, K) \).
      - So from Lemma 7 below, \( \text{dom}(\omega) \in \text{adm}(q_1) \).

- \( q' \in \text{branches}(q_2) \setminus \text{branches}(q_1) \).
  - Since \( \omega' \in \text{sparqlAns}(q_1 \cup q_2, \text{can}(K)) \) and \( q' \notin \text{branches}(q_1) \), \( \omega' \notin \text{sparqlAns}(q_1, \text{can}(K)) \) must hold.
    - Then because \( \omega' \in \text{sparqlAns}(q_1 \cup q_2, \text{can}(K)) \), from Definition 1, \( \omega' \in \text{sparqlAns}(q_2, \text{can}(K)) \), must hold.
      - So from Definition 8, \( \omega \in m\text{CanAns}(q_2, K, q') \).
      - And since \( q' \in \text{branches}(q_2) \), from Definition 10, \( \omega \in m\text{CanAns}(q_2, K) \), which would contradict the hypothesis.
Lemma 7. For any SUJO query $q$, $L_{can}$ KB $K$ and $\omega \in mCanAns(q, K)$:
\[
\text{dom}(\omega) \subseteq \text{adm}(q).
\]

Proof. Let $q$ be a SUJO query and $K$ an $L_{can}$ KB.
Then $mCanAns(q, K) = \bigcup_{q' \in \text{branches}(q)} (\text{sparqlAns}(q, A, q') \cap \text{adm}(K)) \cap \text{adm}(q')$.

So for each $\omega \in mCanAns(q, K)$, there is a $q' \in \text{branches}(q)$ and solution mapping $\omega'$ s.t. $\omega = \omega'|X$ for some $X \in \max_{\subseteq} (\text{adm}(q') \cap 2^{\text{dom}(\omega')})$.

So $\text{dom}(\omega) \subseteq \text{adm}(q)$.

Then Lemma 8 below shows that for any $q' \in \text{branches}(q)$, $\text{adm}(q') \subseteq \text{adm}(q)$.

So $\text{dom}(\omega) \subseteq \text{adm}(q')$. \hfill \square

Lemma 8. For any SUJO query $q$ and SJO $q' \in \text{branches}(q)$:
\[
\text{adm}(q') \subseteq \text{adm}(q).
\]

Proof. Let $q$ be a SUJO query, $q' \in \text{branches}(q)$ and $X \in \text{adm}(q')$.
We need to show that $X \in \text{adm}(q)$.

By induction on $q$:

- $q$ is a triple pattern.
  Then $\text{branches}(q) = \{q\}$, so the property trivially holds.

- $q = \text{SELECTY } q_2$.
  From Definition 9, $q' = q_1 \join q_2$ for some $q'_1 \in \text{branches}(q_1)$.
  So from Definition 4, $X = X_1 \cap X_2$ for some $X_1 \in \text{adm}(q'_1)$.
  Then by IH, $X' \cap X_2 \in \text{adm}(q_2)$.
  And again from Definition 4, $X \in \text{adm}(\text{SELECTY } q_2) = \text{adm}(q)$.

- $q = q_1 \union q_2$.
  From Definition 9, $q' = q'_1 \join q'_2$ for some $(q'_1, q'_2) \in \text{branches}(q_1) \times \text{branches}(q_2)$.
  So from Definition 4, $X = X_1 \cup X_2$ for some $(X_1, X_2) \in \text{adm}(q'_1) \times \text{adm}(q'_2)$.
  Then by IH, $X_1 \in \text{adm}(q_1)$ and $X_2 \in \text{adm}(q_2)$.
  And again from Definition 4, $X \in \text{adm}(q_1 \union q_2) = \text{adm}(q)$.

- If $q' = q_1 \opt q_2$, then $X \in \text{branches}(q_1 \join q_2)$ or $X \in \text{branches}(q_1)$ must hold.
  - If $q' \in \text{branches}(q_1 \join q_2)$, then we showed above that $X \in \text{adm}(q_1 \join q_2)$ must hold.
    And from Definition 4, $\text{adm}(q_1 \join q_2) \subseteq \text{adm}(q)$.
    Then $X \in \text{adm}(q)$.
  - If $q' \in \text{branches}(q_1)$, then by IH, $X \in \text{adm}(q_1)$.
    And from Definition 4, $\text{adm}(q_1) \subseteq \text{adm}(q)$.
    So $X \in \text{adm}(q)$. \hfill \square

Appendix C. Proof of Proposition 3

Proposition 3. For any SUJO query $q$ and KB $K$, $\text{epCertAns}(q, K)$ satisfies Requirements 1, 2, 3 and 4.

Proof. The proposition is split into Lemmas 9, 10, 11 and 12 below, one for each requirement. \hfill \square

Lemma 9. For any UCQ $q$ and KB $K$, $\text{epCertAns}(q, K) = \text{certAns}(q, K)$

Proof. Let $q$ be a UCQ and $K$ a KB.
Since $q$ contains no \opt operator, from Definition 12, $\text{partEval}(q, K) = q$. 

"
So (i) $\text{certAns}(\text{partEval}(q, K), K) = \text{certAns}(q, K)$.

And from Definition 12 still, (ii) $\text{certAns}(\text{partEval}(q, K), K) = \text{epCertAns}(q, K)$.

So from (i) and (ii), $\text{epCertAns}(q, K) = \text{certAns}(q, K)$.  

\textbf{Lemma 10.} For any SUJO query $q$ and ABox $A$,

$$\text{epCertAns}(q, (\emptyset, A)) = \text{sparqlAns}(q, A)$$

\textbf{Proof.} Let $q$ be a SUJO query, let $A$ be an ABox, and let $K = (\emptyset, A)$.

Then $A = \text{can}(K)$.

Now by induction on $q$:

- $q$ is a triple pattern.
  
  Then $\text{partEval}(q, K) = q$.
  
  - ($\Rightarrow$).
    
    Let $\omega \in \text{epCertAns}(q, K)$.
    
    Then from Definition 12, $\omega \in \text{certAns}(\text{partEval}(q, K), K)$.
    
    And since $\text{partEval}(q, K) = q$, $\omega \in \text{certAns}(q, K)$.
    
    So for every model $I$ of $K$, $\omega \in \text{sparqlAns}(q, I)$.
    
    In particular, $\omega \in \text{sparqlAns}(q, A)$.

  
  - ($\Leftarrow$).
    
    Let $\omega \in \text{sparqlAns}(q, A)$.
    
    Then range($\omega$) $\subseteq aDom(K)$.
    
    And since $A = \text{can}(K)$, for any model $I$ of $K$, there is a homomorphism $h$ from $A$ to $I$.
    
    Then because range($\omega$) $\subseteq aDom(K)$, $h \circ \omega = \omega$ must hold.
    
    So for any model $I$ of $K$, $\omega \in \text{sparqlAns}(q, I)$.
    
    And since $\text{partEval}(q, K) = q$, for any model $I$ of $K$,
    
    $\omega \in \text{sparqlAns}(\text{partEval}(q, K), I)$.
    
    Therefore from Definition 12, $\omega \in \text{epCertAns}(q, K)$.

- $q = \text{SELECT} X q'$.
  
  - ($\Rightarrow$).
    
    Let $\omega \in \text{epCertAns}(q, K)$.
    
    Then from Definition 12, $\omega \in \text{certAns}(\text{partEval}(q, K), K)$.
    
    So for every model $I$ of $K$, $\omega \in \text{sparqlAns}(\text{partEval}(q, K), I)$.
    
    In particular, $\omega \in \text{sparqlAns}(\text{partEval}(q, K), A)$.
    
    So from Definitions 1 and 12, $\omega = \omega'|X$ for some $\omega' \in \text{sparqlAns}(\text{partEval}(q', K), A)$.
    
    So range($\omega'$) $\subseteq aDom(K)$.
    
    Then since $A = \text{can}(K)$, for any model $I$ of $K$, there is a homomorphism $h$ from $A$ to $I$.
    
    And because range($\omega'$) $\subseteq aDom(K)$, $h \circ \omega' = \omega'$.
    
    Therefore for any model $I$ of $K$, $\omega' \in \text{sparqlAns}(\text{partEval}(q', K), I)$.
    
    So from Definition 12, $\omega' \in \text{epCertAns}(q', K)$.
    
    Then by IH, $\omega' \in \text{sparqlAns}(q', A)$.
    
    And since $\omega = \omega'|X$, from Definition 1,
    
    $\omega \in \text{sparqlAns}(\text{SELECT} X q', A) = \text{sparqlAns}(q, A)$.

  
  - ($\Leftarrow$).
    
    Let $\omega \in \text{sparqlAns}(q, A)$.
    
    Then from Definition 1, $\omega = \omega'|X$ for some $\omega' \in \text{sparqlAns}(q', A)$.
    
    And by IH, $\omega' \in \text{epCertAns}(q', A)$.
    
    So from Definition 12, for any model $I$ of $K$,
    
    $\omega' \in \text{sparqlAns}(\text{partEval}(q', K), I)$.
    
    And since $\omega = \omega'|X$, from Definition 1, for any model $I$ of $K$,
    
    $\omega \in \text{sparqlAns}(\text{SELECT} X \text{partEval}(q', K), I) = \text{sparqlAns}(\text{partEval}(q, K), I)$.

Therefore from Definition 12, $\omega \in \text{epCertAns}(q, K)$.

- $q = q_1 \text{ JOIN } q_2$.
  
  - ($\Rightarrow$).
    
    Let $\omega \in \text{epCertAns}(q, K)$.
    
    Then from Definition 12, $\omega \in \text{certAns}(\text{partEval}(q, K), K)$.
    
    So for every model $I$ of $K$, $\omega \in \text{sparqlAns}(\text{partEval}(q, K), I)$.
    
    In particular, $\omega \in \text{sparqlAns}(\text{partEval}(q, K), A)$.
    
    So from Definitions 1 and 12, $\omega = \omega_1 \cup \omega_2$ for some $(\omega_1, \omega_2) \in \text{sparqlAns}(\text{partEval}(q_1, K), A) \times \text{sparqlAns}(\text{partEval}(q_2, K), A)$.
    
    So for $i \in \{1, 2\}$, range($\omega_i$) $\subseteq aDom(K)$.
    
    Then since $A = \text{can}(K)$, for any model $I$ of $K$, there is a homomorphism $h$ from $A$ to $I$.
And because range(\(\omega_1\)) \(\subseteq\) aDom(\(K\)), \(h \circ \omega_1 = \omega_1\).
Therefore for any model \(I\) of \(K\), \(\omega_1 \in\) sparqlAns(partEval(q_1, K), I).
So from Definition 12, \(\omega_1 \in\) epCertAns(q_1, K).
Then by IH, \(\omega_1 \in\) sparqlAns(q_1, A).
So from Definition 1, \(\omega = \omega_1 \cup \omega_2 \in\) sparqlAns(q_1 \ join \ q_2, A) = sparqlAns(q, A).

(\(\Rightarrow\))
Let \(\omega \in\) sparqlAns(q, A).
Then from Definition 1, \(\omega = \omega_1 \cup \omega_2\) for some \((\omega_1, \omega_2) \in\) sparqlAns(q_1, A) \times sparqlAns(q_2, A).
So by IH, for \(i \in \{1, 2\}\), \(\omega_i \in\) epCertAns(q_i, K).
So from Definition 12, for any model \(I\) of \(K\), \(\omega \in\) sparqlAns(partEval(q_i, K), I).
Therefore from Definition 1, for any model \(I\) of \(K\), \(\omega = \omega_1 \cup \omega_2 \in\) sparqlAns(partEval(q_1, K) \ join \ partEval(q_2, K), I).
And since partEval(q_1, K) \ join \ partEval(q_2, K) = partEval(q, K), for any model \(I\) of \(K\), \(\omega \in\) sparqlAns(partEval(q, K), I).
So from Definition 12, \(\omega \in\) epCertAns(q, K).

\(\bullet\) \(q = q_1\) \ UNION \ \(q_2\).

(\(\Rightarrow\))
Let \(\omega \in\) epCertAns(q, K).
Then from Definition 12, \(\omega \in\) certAns(partEval(q, K), K).
So for every model \(I\) of \(K\), \(\omega \in\) sparqlAns(partEval(q, K), I).
In particular, \(\omega \in\) sparqlAns(partEval(q, K), A).
So range(\(\omega\)) \(\subseteq\) aDom(K).
In addition, from Definitions 1 and 12, \(\omega \in\) sparqlAns(partEval(q_i, K), A) for some \(i \in \{1, 2\}\).
Then since \(A =\) can(K), for any model \(I\) of \(K\), there is a homomorphism \(h\) from \(A\) to \(I\).
And because range(\(\omega\)) \(\subseteq\) aDom(\(K\)), \(h \circ \omega = \omega\).
So for any model \(I\) of \(K\), \(\omega \in\) sparqlAns(partEval(q_i, K), I).
So from Definition 12, \(\omega \in\) epCertAns(q_i, K).
Then by IH, \(\omega \in\) sparqlAns(q_i, A) must hold.
So from Definition 1, \(\omega \in\) sparqlAns(q_1 \ UNION \ q_2, A) = sparqlAns(q, A).

(\(\Leftarrow\))
Let \(\omega \in\) sparqlAns(q, A).
Then from Definition 12, \(\omega \in\) certAns(partEval(q, K), K).
So for every model \(I\) of \(K\), \(\omega \in\) sparqlAns(partEval(q, K), I).
In particular, \(\omega \in\) sparqlAns(partEval(q, K), A).
So from Definitions 1 and 12, \(\omega \in\) sparqlAns(partEval(q_i, K), A) or \(\omega \in\) sparqlAns(partEval(q_i, K) \ MINUS \ epCertAns(q_2, K), A) must hold.
* If \(\omega \in\) sparqlAns(partEval(q_1, K) \ join \ partEval(q_2, K), A),
  as shown in the proof for the case \(q = q_1\) \ join \ \(q_2\),
  \(\omega \in\) sparqlAns(q_1 \ join \ q_2, A) must hold.
  So from Definition 1, \(\omega \in\) sparqlAns(q_1 \ OPT \ q_2, A) = sparqlAns(q, A).
* If \(\omega \in\) sparqlAns(partEval(q_1, K) \ MINUS \ epCertAns(q_2, K), A),
  then (i) \(\omega \in\) sparqlAns(partEval(q_1, K), A) and
  (ii) for all \(\omega' \in\) epCertAns(q_2, K), \(\omega \sim \omega'\).
  From (i), range(\(\omega\)) \(\subseteq\) aDom(K) must hold.
  And since \(A =\) can(K), for any model \(I\) of \(K\), there is a homomorphism \(h\) from \(A\) to \(I\).
  Then because range(\(\omega\)) \(\subseteq\) aDom(\(K\)), \(h \circ \omega = \omega\).
  So for any model \(I\) of \(K\), \(\omega \in\) sparqlAns(partEval(q_1, K), I).
  So from Definition 12, \(\omega \in\) epCertAns(q_1, K).
  So by IH, (iii) \(\omega \in\) sparqlAns(q_1, A).
  And by IH still, (iv) epCertAns(q_2, K) = sparqlAns(q_2, A).
  So from (ii) and (iv), (v) for all \(\omega' \in\) epCertAns(q_2, A), \(\omega \sim \omega'\).
  So from (iii), (v) and Definition 1, \(\omega \in\) sparqlAns(q_1 \ OPT \ q_2, A) = sparqlAns(q, A).

(\(\Leftarrow\))
Let \(\omega \in\) sparqlAns(q, A).
Then from Definition 1, \(\omega \in\) sparqlAns(q_1 \ join \ q_2, A) or \(\omega \in\) sparqlAns(q_1 \ MINUS \ q_2, A) must hold.
* If $\omega \in \text{sparqlAns}(q_1 \text{ join } q_2, A)$, as shown for the case $q = q_1 \text{ join } q_2$, $\omega \in \text{epCertAns}(q_1 \text{ join } q_2, K)$ must hold. So from Definition 12, for every model $I$ of $K$,

$$\omega \in \text{sparqlAns}(\text{partEval}(q_1 \text{ join } q_2, K), I)$$

$$= \text{sparqlAns}(\text{partEval}(q_1, K) \text{ join partEval}(q_2, K), I)$$

$$\subseteq \text{sparqlAns}((\text{partEval}(q_1, K) \text{ join partEval}(q_2, K)))$$

$$\text{union} (\text{partEval}(q_1, K) \text{ minus epCertAns}(q_2, K)), I)$$

$$= \text{sparqlAns}(\text{partEval}(q, K), I).$$

So from Definition 1, $\omega \in \text{epCertAns}(q, K)$.

* If $\omega \in \text{sparqlAns}(q_1 \text{ minus } q_2, A)$, then (i) $\omega \in \text{sparqlAns}(q_1, A)$ and (ii) for all $\omega' \in \text{sparqlAns}(q_2, A)$, $\omega \sim \omega'$. From (i), by IH, $\omega \in \text{epCertAns}(q_1, K)$.

So from Definition 12, (iii) for any model $I$ of $K$, $\omega \in \text{sparqlAns}(\text{partEval}(q_1, K), I)$.

Then again by IH, $\text{sparqlAns}(q_2, A) = \text{epCertAns}(q_2, K)$.

So from (ii), (iv) for all $\omega' \in \text{epCertAns}(q_2, K)$, $\omega \sim \omega'$.

Therefore from (iii), (iv) and Definitions 1 and 12, for any model $I$ of $K$,

$$\omega \in \text{sparqlAns}(\text{partEval}(q_1, K) \text{ minus epCertAns}(q_2, K), I)$$

$$\subseteq \text{sparqlAns}((\text{partEval}(q_1, K) \text{ minus epCertAns}(q_2, K)))$$

$$\text{union} (\text{partEval}(q_1, K) \text{ join partEval}(q_2, K)), I)$$

$$= \text{sparqlAns}(\text{partEval}(q, K), I).$$

So from Definition 12, $\omega \in \text{epCertAns}(q, K)$. □

**Lemma 11.** For any SUJO queries $q_1, q_2$ and KB $K$:

$$\text{epCertAns}(q_1, K) \preceq_j \text{epCertAns}(q_2, K)$$

**Proof.** Let $q_1, q_2$ be SUJO queries, let $K$ be a KB, and let $\omega_1 \in \text{epCertAns}(q_1, K)$. We need to show that there is an $\omega' \in \text{epCertAns}(q_1 \text{ opt } q_2, K)$ s.t. $\omega_1 \preceq \omega'$.

Since $\omega_1 \in \text{epCertAns}(q_1, K)$, from Definition 12, for every model $I$ of $K$, $\omega_1 \in \text{sparqlAns}(\text{partEval}(q_1, K), I)$.

Then we have two cases:

- there is an $\omega_2 \in \text{epCertAns}(q_2, K)$ s.t. $\omega_1 \sim \omega_2$.

  Then for every model $I$ of $K$, $\omega_2 \in \text{sparqlAns}(\text{partEval}(q_2, K), I)$.

  So from Definitions 1 and 12, for every model $I$ of $K$:

  $$\omega_1 \cup \omega_2 \in \text{sparqlAns}(\text{partEval}(q_1, K) \cup \text{partEval}(q_2, K), I)$$

  $$\subseteq \text{sparqlAns}((\text{partEval}(q_1, K) \cup \text{partEval}(q_2, K)))$$

  $$\text{union} (\text{partEval}(q_1, K) \text{ minus certAns}(q_2, K)), I)$$

  $$= \text{sparqlAns}(\text{partEval}(q_1 \text{ opt } q_2, K), I)$$

  So from Definition 12, $\omega_1 \in \text{epCertAns}(q_1 \text{ opt } q_2, K)$.

And $\omega_1 \preceq \omega_2$.

- there is no $\omega_2 \in \text{epCertAns}(q_2, K)$ s.t. $\omega_2 \sim \omega$.

  Then from Definitions 1 and 12, for every model $I$ of $K$:

  $$\omega_1 \in \text{sparqlAns}(\text{partEval}(q_1, K) \text{ minus epCertAns}(q_2, K), I)$$

  $$\subseteq \text{sparqlAns}((\text{partEval}(q_1, K) \text{ join partEval}(q_2, K)))$$

  $$\text{union} (\text{partEval}(q_1, K) \text{ minus certAns}(q_2, K)), I)$$

  $$= \text{sparqlAns}(\text{partEval}(q_1 \text{ opt } q_2, K), I)$$

  So from Definition 12, $\omega_1 \in \text{epCertAns}(q_1 \text{ opt } q_2, K)$.

And $\omega_1 \preceq \omega_2$. □

**Lemma 12.** For any SUJO queries $q, q'$, KB $K$ and solution mapping $\omega$,

if $\omega \in \text{epCertAns}(q \text{ union } q', K)$ and $\omega \notin \text{epCertAns}(q', K)$, then $\text{dom}(\omega) \in \text{adm}(q)$
Proof. Let \( q, q' \) be two SUJO queries let \( \mathcal{K} \) be a KB, and let \( \omega \in \text{epCertAns}(q \cup q', \mathcal{K}) \) s.t. \( \omega \notin \text{epCertAns}(q', \mathcal{K}) \).

From Definition 12,

\[
\text{epCertAns}(q \cup q', \mathcal{K}) = \text{certAns}(\text{partEval}(q, \mathcal{K}) \cup \text{partEval}(q', \mathcal{K})),
\]

and \( \text{epCertAns}(q', \mathcal{K}) = \text{certAns}(\text{partEval}(q', \mathcal{K}), \mathcal{K}) \).

So there is a model \( \mathcal{I} \) of \( \mathcal{K} \) s.t.

\( \omega \in \text{sparqlAns}(\text{partEval}(q, \mathcal{K}) \cup \text{partEval}(q', \mathcal{K}), \mathcal{I}) \) and

\( \omega \notin \text{sparqlAns}(\text{partEval}(q', \mathcal{K}), \mathcal{I}) \).

Therefore from Definition 1, \( \omega \in \text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \mathcal{I}) \) must hold.

Then from Lemma 13 below, it follows that \( \text{dom}(\omega) \in \text{adm}(q) \). \( \square \)

Lemma 13. For any SUJO query \( q \) and graph \( \mathcal{I} \), for any \( \omega \in \text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \mathcal{I}) \).

\[
\text{dom}(\omega) \in \text{adm}(q). 
\]

Proof. Let \( q \) be a SUJO query, let \( \mathcal{I} \) be a graph, and let \( \omega \in \text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \mathcal{I}) \).

By induction on \( q \):

- \( q \) is a triple pattern.
  Then \( \text{partEval}(q, \mathcal{K}) = q \).
  And from Definitions 1 and 4,
  \[
  \text{dom}(\omega) = \text{vars}(\text{partEval}(q, \mathcal{K})) = \text{vars}(q) \in \text{adm}(q).
  \]

- \( q = \text{SELECT}_X q' \).
  Then \( \text{partEval}(q, \mathcal{K}) = \text{SELECT}_X \text{partEval}(q', \mathcal{K}) \).
  So from Definition 1, there is a \( \omega' \in \text{sparqlAns}(\text{partEval}(q', \mathcal{K}), \mathcal{I}) \)
  s.t. \( \omega = \omega'|X \).
  And by IH, \( \text{dom}(\omega') \in \text{adm}(q) \).
  So from Definition 4, \( \text{dom}(\omega') \cap X = \text{dom}(\omega) \in \text{adm}(q) \).

- \( q = q_1 \cup q_2 \).
  Then \( \text{partEval}(q, \mathcal{K}) = \text{partEval}(q_1, \mathcal{K}) \cup \text{partEval}(q_2, \mathcal{K}) \).
  So from Definition 1, \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}), \mathcal{I}) \) or
  \( \omega \in \text{sparqlAns}(\text{partEval}(q_2, \mathcal{K}), \mathcal{I}) \) must hold.
  So by IH, for some \( i \in \{1, 2\} \), \( \text{dom}(\omega) \in \text{adm}(q_i) \).
  And from Definition 4, \( \text{adm}(q_i) \subseteq \text{adm}(q) \).
  So \( \text{dom}(\omega) \in \text{adm}(q) \).

- \( q = q_1 \text{ OPT} q_2 \).
  Then \( \text{partEval}(q, \mathcal{K}) = (\text{partEval}(q_1, \mathcal{K}) \text{ OPT} \text{partEval}(q_2, \mathcal{K})) \cup \text{epCertAns}(q_2, \mathcal{K}). \)
  So from Definition 1,
  \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}) \cup \text{partEval}(q_2, \mathcal{K}), \mathcal{I}) \) or
  \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}) \cup \text{epCertAns}(q_2, \mathcal{K}), \mathcal{I}) \) must hold.
  If \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}) \cup \text{partEval}(q_2, \mathcal{K}), \mathcal{I}) \),
  from the proof above for the case \( q = q_1 \text{ JOIN} q_2 \),
  \( \text{dom}(\omega) = \text{adm}(q_1) \cup \text{adm}(q_2) \) must hold.
  And from Definition 4, \( \text{adm}(q_1) \cup \text{adm}(q_2) \subseteq \text{adm}(q) \).
  So \( \text{dom}(\omega) \in \text{adm}(q) \).
  If \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}) \cup \text{epCertAns}(q_2, \mathcal{K}), \mathcal{I}) \),
  then \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K})) \).
  So by IH, \( \text{dom}(\omega) \in \text{adm}(q_1) \).
  And from Definition 4, \( \text{adm}(q_1) \subseteq \text{adm}(q) \).
  So \( \text{dom}(\omega) \in \text{adm}(q) \). \( \square \)

Appendix D. Proof of Proposition 4

Proposition 4. For any \( \mathcal{L}_{can} \) KB \( \mathcal{K} \) and SUJO query \( q \),

\[
\text{epCertAns}(q, \mathcal{K}) = \text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \text{can}(\mathcal{K})) \Rightarrow \text{aDom}(\mathcal{K})
\]

Proof. Let \( q \) be a SUJO query and \( \mathcal{K} \) an \( \mathcal{L}_{can} \) KB.

20
• (⇒).
  Let \( \omega \in \text{epCertAns}(q, \mathcal{K}) \).
  Then from Definition 12, \( \omega \in \text{certAns}(\text{partEval}(q, \mathcal{K}), \mathcal{K}) \).
  So \( \text{range}(\omega) \subseteq \text{aDom}(\mathcal{K}) \) and \( \omega \in \text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \text{can}(\mathcal{K})) \) both hold.

• (⇐).
  By induction on \( q \).
  For the cases:
  - \( q \) is a triple pattern,
  - \( q = \text{SELECTX } q' \),
  - \( q = q_1 \text{ JOIN } q_2 \),
  - \( q = q_1 \text{ UNION } q_2 \).
  the proof is identical to the one of Lemma 10 (left direction).
  For the case \( q = q_1 \text{ OPT } q_2 \), let \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}), \text{can}(\mathcal{K})) \Rightarrow \text{aDom}(\mathcal{K}) \).
  Then either:
  - \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}) \text{ JOIN } \text{partEval}(q_2, \mathcal{K}), \text{can}(\mathcal{K})) \).
  - (i) \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}), \text{can}(\mathcal{K})) \), and
  - (ii) for all \( \omega' \in \text{epCertAns}(q_2, \mathcal{K}) \), \( \omega' \equiv \omega' \).
  From (i), by IH, \( \omega \in \text{epCertAns}(q_1, \mathcal{K}) \).
  So from Definition 12, (iii) for any model \( \mathcal{I} \) of \( \mathcal{K} \),
  \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}), \mathcal{I}) \).
  So from (ii) and (iii), for any model \( \mathcal{I} \) of \( \mathcal{K} \),
  \( \omega \in \text{sparqlAns}(\text{partEval}(q_1, \mathcal{K}) \text{ MINUS epCertAns}(q_2, \mathcal{K}), \mathcal{I}) \).
  Therefore from Definition 12, \( \omega \in \text{epCertAns}(q, \mathcal{K}) \).

### Appendix E. Proof of Proposition 6

**Proposition 6.** For any \( \mathcal{L}_{can} \) KB \( \mathcal{K} \) and SUJ query \( q \).

\[
\text{mCanAns}(q, \mathcal{K}) = \text{epCertAns}(q, \mathcal{K})
\]

**Proof.** Let \( q \) be a SUJ query, and let \( \mathcal{K} \) be \( \mathcal{L}_{can} \) KB.
From Proposition 4,
\[
\text{epCertAns}(q, \mathcal{K}) = \text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \text{can}(\mathcal{K})) \Rightarrow \text{aDom}(\mathcal{K}).
\]
So in order to prove the result, it is sufficient to show that
\[
\text{mCanAns}(q, \mathcal{K}) = \text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \text{can}(\mathcal{K})) \Rightarrow \text{aDom}(\mathcal{K}).
\]
Then from Definition 12, because \( q \) is a SUJ query, \( \text{partEval}(q, \mathcal{K}) = q \).
So it is sufficient to show that
\[
\text{mCanAns}(q, \mathcal{K}) = \text{sparqlAns}(q, \text{can}(\mathcal{K})) \Rightarrow \text{aDom}(\mathcal{K}).
\]
The left direction is immediate from Definition 10, and the fact that for any graph \( \mathcal{G} \) and SUJO query \( q \), if \( \omega \in \text{sparqlAns}(q, \mathcal{G}) \), then there must be a \( b \in \text{branches}(q) \) s.t. \( \omega \in \text{sparqlAns}(b, \mathcal{G}) \) and \( \text{dom}(\omega) \in \text{adm}(b) \).
For the right direction, let \( \omega \in \text{mCanAns}(q, \mathcal{K}) \).
Then from Definition 10, there must be a \( b \in \text{branches}(q) \) s.t.
\[
(i) \ \omega \in \text{mCanAns}(q, \mathcal{K}, b).
\]
So from (i), (ii) \( \text{dom}(\omega) \in \text{adm}(b) \).
Now from Definition 9, because \( q \) is a SUJ query, \( b \) must be an SJ query.
So from Definition 4, (iii) \( \text{adm}(b) = \{\text{vars}(b)\} \).
So from (ii) and (iii), (iv) \( \text{dom}(\omega) = \text{vars}(b) \).
So from (i) and (iv), \( \omega \in \text{sparqlAns}(q, \text{can}(\mathcal{K})) \) and \( \text{range}(\omega) \subseteq \text{aDom}(\mathcal{K}) \).

### Appendix F. Complexity of maximal admissible certain answers

**Proposition 7.** If \( q \) is a JO query and \( X_1, X_2 \subseteq \text{vars}(q) \), then it can be decided in \( O(|q|^2) \) whether \( X_1 \in \text{max}_{\subseteq}(\text{adm}(q) \cap 2^{X_2}) \)

**Proof.** Let \( q \) be a JO query and \( X_1, X_2 \subseteq \text{vars}(q) \).
We reproduce here the inductive definition of base(q), for readability.

**Definition 13 (Base of a JO query).**

- If \( q \) is a triple pattern, then base(q) = \{\text{vars}(q)\}.
- If \( q = q_1 \text{ JOIN } q_2 \), then base(q) = \( \{B_1 \cup B_2 \mid B_1 \in \text{min}_{\subseteq} \text{base}(q_1) \}, B_2 \in \text{base}(q_2)\} \cup \{B_1 \cup B_2 \mid B_1 \in \text{base}(q_1), B_2 \in \text{min}_{\subseteq} \text{base}(q_2)\} \).
- If \( q = q_1 \text{ OPT } q_2 \), then base(q) = base(q_1) \cup \text{base}(q_1 \text{ JOIN } q_2) \).
In order to complete the proof sketched in Section 6, it is sufficient to show that:

- For any JO query \( q \), the minimal element of base(\( q \)) w.r.t. set-inclusion is guaranteed to be unique. This is shown with Lemma 14 below.
- \( \text{adm}(q) = \{ \bigcup B \mid B \in 2^{\text{base}(q)} \} \). This is shown with Lemma 15 below.
- \( |\text{base}(q)| = 0(|q|) \). This is shown with Lemma 16 below. □

**Lemma 14.** For any JO query \( q \), \( |\text{min}_{\subseteq}(\text{base}(q))| = 1 \).

**Proof.** By induction on the structure of \( q \).

- If \( q \) is a triple pattern, then \( |\text{base}(q)| = 1 \), so \( |\text{min}_{\subseteq}(\text{base}(q))| = 1 \).
- If \( q = q_1 \ JOIN q_2 \), let \( B_1 = \{ B_1 \cup B_2 \mid B_1 \in \text{min}_{\subseteq}(\text{base}(q_1)) \}, B_2 \in \text{base}(q_2) \}, \) and \( B_2 = \{ B_1 \cup B_2 \mid B_1 \in \text{base}(q_1), B_2 \in \text{min}_{\subseteq}(\text{base}(q_2)) \} \).

  By IH, for \( i \in [1, 2] \), \( |\text{min}_{\subseteq}(\text{base}(q_i))| = |M_i| \) for some \( M_i \subseteq \text{vars}(q_i) \).

  Then from the definition of \( B_1 \), \( M_1 \cup M_2 \in B_1 \).

  And for each \( B_2 \in \text{base}(q_2) \), \( M_2 \subseteq B_2 \).

  So for each \( M_1 \cup M_2 \in B_1 \), \( M_1 \cup M_2 \subseteq M_1 \cup B_2 \).

  So \( \text{min}_{\subseteq}(B_1) = \{ M_1 \cup M_2 \} \).

  And similarly, \( \text{min}_{\subseteq}(B_2) = \{ M_1 \cup M_2 \} \).

  Then because base(\( q \)) = \( B_1 \cup B_2 \), \( \text{min}_{\subseteq}(\text{base}(q)) = \{ M_1 \cup M_2 \} \).

- If \( q = q_1 \ OPT q_2 \), by IH, \( \text{min}_{\subseteq}(\text{base}(q_1)) = \{ M \} \) for some \( M \subseteq \text{vars}(q_1) \).

  So \( M \subseteq B \) for each \( B \in \text{base}(q_1) \).

  And we showed above that \( M \subseteq B \) for each \( B \in \text{base}(q_1 \ JOIN q_2) \).

  Then from Definition 14, base(\( q \)) = base(\( q_1 \)) \cup base(\( q_1 \ JOIN q_2) \).

  So \( M \in \text{base}(q_1) \subseteq \text{base}(q) \), and \( M \subseteq B \) for each \( B \in \text{base}(q_1) \cup \text{base}(q_1 \ JOIN q_2) = \text{base}(q) \).

  Therefore \( \text{min}_{\subseteq}(\text{base}(q)) = \{ M \} \). □

**Lemma 15.** For any JO query \( q \), \( \text{adm}(q) = \{ \bigcup B \mid B \in 2^{\text{base}(q)} \} \)

**Proof.** By induction on the structure of \( q \).

- If \( q \) is a triple pattern, then base(\( q \)) = adm(\( q \)) = \{ \text{vars}(q) \} \).

- If \( q = q_1 \ JOIN q_2 \):

  - \( \Rightarrow \).

    Let \( X \in \text{adm}(q) \).

    From Definition 4, \( X = X_1 \cup X_2 \) for some \( (X_1, X_2) \in \text{adm}(q_1) \times \text{adm}(q_2) \).

    And by IH, for \( i \in [1, 2] \), \( X_i = \bigcup B_i \) for some \( B_i \in 2^{\text{base}(q_i)} \).

    Then from Lemma 14, \( |\text{min}_{\subseteq}(\text{base}(q_i))| = |M_i| \) for some \( M_i \subseteq \text{vars}(q_i) \).

    So for each \( B_i \in B_i \), \( M_i \subseteq B_i \).

    Therefore \( \bigcup B_i = \{ M_i \} \cup B_i \).

    And since \( X = X_1 \cup X_2 \), we have:

    \[
    X = \bigcup B_1 \cup \bigcup B_2 \\
    X = \{ M_1 \cup \bigcup B_1 \cup \{ M_2 \} \cup \bigcup B_2 \\
    X = \{ M_2 \cup B_1 \mid B_1 \in B_1 \} \cup \{ M_1 \cup B_2 \mid B_2 \in B_2 \}
    \]

    Then from Definition 13, for each \( B_1 \in B_1 \), \( M_2 \cup B_1 \in \text{base}(q) \).

    Similarly, for each \( B_2 \in B_2 \), \( M_1 \cup B_2 \in \text{base}(q) \).

    So \( X = \bigcup B \) for some \( B \in 2^{\text{base}(q)} \).

  - \( \Leftarrow \).

    Let \( X = \bigcup B \) for some \( B \in 2^{\text{base}(q)} \).

    From Definition 13, for each \( B \in \text{adm}(q) \), there are \( (B_1, B_2) \in \text{base}(q_1) \times \text{base}(q_2) \) s.t. \( B = B_1 \cup B_2 \).

    For \( i \in [1, 2] \), let \( B_i = \{ B_i \mid B_i \in B, B_i \in \text{adm}(q_i) \} \).

    Then for \( i \in [1, 2] \), \( B_i \neq \emptyset \).

    And \( B = B_1 \cup B_2 \).

    So \( X = \bigcup B = \bigcup B_1 \cup \bigcup B_2 \).

    And by IH, for \( i \in [1, 2] \), \( \bigcup B_i \in \text{adm}(q_i) \).

    Therefore \( X = X_1 \cup X_2 \) for some \( (X_1, X_2) \in \text{adm}(q_1) \times \text{adm}(q_2) \).

    So from Definition 4, \( X \in \text{adm}(q) \).

- If \( q = q_1 \ OPT q_2 \):

  - \( \Rightarrow \).

    Let \( X \in \text{adm}(q) \).

    From Definition 4, \( X \in \text{adm}(q_1) \) or \( X \in \text{adm}(q_1 \ JOIN q_2) \) must hold.

    If \( X \in \text{adm}(q_1) \), then by IH, \( X = \bigcup B \) for some \( B \in 2^{\text{base}(q_1)} \).
And from Definition 13, \( \text{base}(q_1) \subseteq \text{base}(q) \).

If \( X \in \text{adm}(q_1 \cup q_2) \), then we showed above that \( X = \bigcup B \) for some \( B \in 2^{\text{base}(q_1 \cup q_2)} \).

And from Definition 13, \( \text{base}(q_1 \cup q_2) \subseteq \text{base}(q) \).

So in both cases, \( X = \bigcup B \) for some \( B \in 2^{\text{base}(q)} \).

= \{ \}

Let \( X = \bigcup B \) for some \( B \in 2^{\text{base}(q)} \).

From Definition 13, for each \( B \in B, B \in \text{base}(q_1) \) or there are \( (B_1, B_2) \in \text{base}(q_1) \times \text{base}(q_2) \) s.t. \( B = B_1 \cup B_2 \).

For \( i \in \{1, 2\} \), let \( B_i = \{ B_1 | B_1 \cup B' \in B, B_i \in \text{base}(q_i) \} \).

Then \( B_1 \neq \emptyset \).

And \( B = B_1 \cup B_2 \).

If \( B_2 = \emptyset \), then \( X = \bigcup B = \bigcup B_1 \).

And by IH, \( \bigcup B_1 \in \text{adm}(q_1) \).

So From Definition 4, \( X \in \text{adm}(q) \).

If \( B_2 \neq \emptyset \), then \( X = \bigcup B = \bigcup B_1 \cup B_2 \).

And by IH, for \( i \in \{1, 2\} \), \( \bigcup B_i \in \text{adm}(q_i) \).

Therefore \( X = X_1 \cup X_2 \) for some \( (X_1, X_2) \in \text{adm}(q_1) \times \text{adm}(q_2) \).

So From Definition 4, \( X \in \text{adm}(q) \).

\[ \square \]

Lemma 16. For any FO query \( q \), \( \text{|base}(q)| = O(|q|) \)

Proof. By induction on the structure of \( q \).

- If \( q \) is a triple pattern, then \( \text{|base}(q)| = 1 \).
- If \( q = q_1 \cup q_2 \), then immediately from the definition of \( \text{base}(q) \).
  \[ \text{|base}(q) = O(|q_1|) + \text{|base}(q_2)| \]

So from Lemma 14, \( \text{|base}(q)| = O(|\text{base}(q_1)| + |\text{base}(q_2)|) \).

And by IH, \( \text{|base}(q_i)| = O(|q_i|) \) for \( i \in \{1, 2\} \).

So \( \text{base}(q) = O(|q_1|) + O(|q_2|) = O(|q|) \).

If \( q = q_1 \cup q_2 \), the argument is similar to the case \( q = q_1 \cup q_2 \).

\[ \square \]

Appendix G. Complexity of epistemic certain answers

Proposition 8. \text{EVAL}_{\text{epCertAns}} \text{ is in } \text{PSpace} \text{ for } \text{SUJO} \text{ queries}.

Proof. Let \( q_0 \) be a SUJO query, let \( K \) be an \( L_{\text{can}} \) KB, and let \( \omega \) be a solution mapping.

Algorithm 1 below describes a procedure that decides \( \omega \in \text{sparqlAnspartEval}(q_0, K) \), \( \text{can}(K) \).

From Proposition 4, in order to decide \( \omega \in \text{epCertAns}(q_0, K) \), it is sufficient to call this procedure and decide \( \text{range}(\omega) \subseteq \text{aDom}(K) \).

In order to explain the procedure, we recall some well-known properties of canonical models for \( DL_{\text{Lite}_K} \) KBs. Specifically, \( \text{can}(K) \) below refers to the canonical model of a \( DL_{\text{Lite}_K} \) KB \( K \) defined in [5].

\( (I) \) Let \( \text{roles}(T) \) be the set of binary predicates (i.e. elements of \( N_0 \)) that appears in \( T \) and their inverse.

Then each element of the domain \( \Delta_{\text{can}(K)} \) of \( \text{can}(K) \) can be identified by a word \( w = aR_1..R_n \), with \( n \geq 0 \), where \( a \in \text{aDom}(K) \) and \( R_i \in \text{roles}(T) \) for \( i \in [1..n] \).

If \( w \) is a word over \( \text{aDom}(K) \cup \text{roles}(T) \), we will use \( l(w) \) to denote its length.

Note that \( w \) may not identify any element of \( \Delta_{\text{can}(K)} \).

If it does, we will use \( w \in \Delta_{\text{can}(K)} \) as a shortcut.

\( (II) \) Let \( w = aR_1..R_n \) be a word over \( \text{aDom}(K) \cup \text{roles}(T) \).

Then \( w \in \Delta_{\text{can}(K)} \) can be decided as follows:

- if \( n = 0 \), then decide \( a \in \text{aDom}(K) \),
- otherwise iterate (at most \( O(|T|) \) times) over \( T \) to decide \( T \models \exists R_{n-1} \subseteq \exists R_n \), then again to decide \( T \models \exists R_{n-2} \subseteq \exists R_{n-1} \), etc., and
- finally decide \( K \models \exists R_1(a) \).

The amount of space required in both cases is polynomial in \( |K| + l(w) \).

\( (III) \) If \( A(w) \) is a tuple with \( A \in N_0 \) and \( w = aR_1..R_n \in \Delta_{\text{can}(K)} \),

then \( A(w) \in \text{can}(K) \) can be decided as follows:

- if \( n = 0 \), then decide \( K \models A(a) \),
- otherwise decide \( T \models \exists R_1(a) \).

Both can be decided by iterating over \( T \) and/or \( A \), using space polynomial in \( |K| \).

And similarly for the case \( R(w_1, w_2) \in \text{can}(K) \), where \( R \in N_0 \) (with an additional verification: one of \( w_1, w_2 \) must be an immediate prefix of the other).

\( (IV) \) If \( q \) is a boolean CQ, and if \( m \) is the number of variables that appear in \( q \), then \( \{ \} \in \text{sparqlAnspart}(q, \text{can}(K)) \) iff there is a match \( \omega \) for \( q \) in \( \text{can}(K) \) such that, for each \( w \in \text{range}(\omega) \), \( l(w) \leq \max(|T|, m) \).

Algorithm 1 simulates a top-down evaluation of an input boolean query \( q_0 \) over \( \text{can}(K) \), where \( q_0 \) contains \( m \) variables. Correctness follows immediately from Definitions 1 and 12, and Point \( (IV) \) above.
To see why this procedure can be executed using space polynomial in $|K| + |q|$, observe first that each call to the function isMatch provides a solution mapping $\omega$ such that each $w \in \text{range}(\omega)$ verifies $w \in \Delta^{\text{can}}(K)$ and $l(w) \leq \max(|T|, m)$.

Then:

- In the case $q = t$ (line 7), because $\text{range}(\omega) \subseteq \Delta^{\text{can}}(K)$, from Point (III) above, $\omega(t) \in \text{can}(K)$ can be decided in polynomial space.
- In the case $q = \text{select}\_X q'$ (line 11), all possible mappings from the set $V$ of distinguished variables in $q$ to words in $\Delta^{\text{can}}(K)$ with length $\leq \max(|T|, m)$ are enumerated. This can be achieved by enumerating all words $w$ over adom($K$) $\cup \text{roles}(T)$ of length $\leq \max(|T|, m)$, and, for each of these, decide $w \in \Delta^{\text{can}}(K)$ using polynomial space, as explained in Point (II) above.
- For the case $q = q_1 \cup q_2$, from Definition 1, $\omega \in \text{sparqlAns}(q, \text{can}(K))$ iff there are two sets $V_1, V_2$ of variables s.t., for $i \in \{1, 2\}$, $V_i \subseteq \text{vars}(q_i)$ and $\omega|_{V_i} \in \text{sparqlAns}(q_i, \text{can}(K))$. Moreover, the cartesian product $2^{|\text{vars}(q_1)|} \times 2^{|\text{vars}(q_2)|}$ can be enumerated (line 23) using space polynomial in $|\text{vars}(q)|$.

References


