Big Data Research 23 (2021) 100177

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Ontology-Mediated SPARQL Query Answering over Knowledge Graphs

Guohui Xiao *, Julien Corman

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

ARTICLE INFO

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

ABSTRACT

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 satisfies 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-Lite*$_{\mathcal{R}}$ ontology, showing that for several fragments of SPARQL, known upper-bounds for query answering over a plain KG are matched.

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.[1]

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.
$\mathcal{A} = \{\texttt{Driver(Alice)}\}$
$\mathcal{T} = \{\texttt{Driver} \sqsubseteq \exists \texttt{hasLicense}\}$
q = `SELECT ?x WHERE { ?x 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: $q(x) \leftarrow \texttt{hasLicense}(x, y)$. 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 Graph" 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.

https://doi.org/10.1016/j.bdr.2020.100177
2214-5796/

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*$_{\mathcal{R}}$, a popular DL tailored for query answering, which underpins the OWL 2 QL 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 $\mathsf{N_I}$, $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_V}$ of *individuals* (constants), *concept names* (unary predicates), *role names* (binary predicates), and variables respectively. We also assume an infinite universe U, such that $\mathsf{N_I} \subseteq \mathsf{U}$. 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 $\mathsf{N_I}$, $\mathsf{N_C}$ and $\mathsf{N_R}$ 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 $(\mathsf{N_I} \times \{\texttt{rdf:type}\} \times \mathsf{N_C}) \cup (\mathsf{N_I} \times \mathsf{N_R} \times \mathsf{N_I})$. An RDF KG, or simply *graph*, is a set of triples, noted $\mathcal{A}$.

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 $(\mathsf{N_V} \cup \mathsf{N_I}) \times \{\texttt{rdf:type}\} \times \mathsf{N_C}$, or in $(\mathsf{N_V} \cup \mathsf{N_I}) \times \mathsf{N_R} \times (\mathsf{N_V} \cup \mathsf{N_I})$. For readability, we represent triples and triple patterns as atoms in prefix notation, i.e. we use $A(e)$ rather than $(e, \texttt{rdf:type}, A)$ and for $r \in \mathsf{N_R}$, we use $r(e_1, e_2)$ rather than (e_1, r, e_2). If q is a SPARQL query, we use $\mathsf{vars}(q)$ 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 $X \subseteq \mathsf{N_V}$:

$$q \quad ::= \quad t \mid \text{SELECT}_X\ q \mid q\ \text{UNION}\ q \mid q\ \text{JOIN}\ q \mid q\ \text{OPT}\ q$$

In addition, if $q = \text{SELECT}_X\ q'$, then $X \subseteq \mathsf{vars}(q')$ 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 $\mathsf{dom}(\omega)$ (resp. $\mathsf{range}(\omega)$) to designate its domain (resp. range). Two functions ω_1 and ω_2 are *compatible*, denoted with $\omega_1 \sim \omega_2$, iff $\omega_1(x) = \omega_2(x)$ for each $x \in \mathsf{dom}(\omega_1) \cap \mathsf{dom}(\omega_2)$. If ω_1 and ω_2 are compatible, then $\omega_1 \cup \omega_2$ is the only function with domain $\mathsf{dom}(\omega_1) \cup \mathsf{dom}(\omega_2)$ that is compatible with ω_1 and ω_2. We say that a function ω_2 *extends* a function ω_1, noted $\omega_1 \preceq \omega_2$, iff $\mathsf{dom}(\omega_1) \subseteq \mathsf{dom}(\omega_2)$ and $\omega_1 \sim \omega_2$. Finally, we use $\omega|_X$ (resp. $\omega\|_X$) to designate the restriction of function ω to domain (resp. co-domain) X, i.e. $\omega|_X$ is the only function compatible with ω that verifies $\mathsf{dom}(\omega|_X) = \mathsf{dom}(\omega) \cap X$, and $\omega\|_X$ is the only function compatible with ω that verifies $\mathsf{dom}(\omega\|_X) = \{v \in \mathsf{dom}(\omega) \mid \omega(v) \in X\}$.

A *solution mapping* is a function from a finite subset of $\mathsf{N_V}$ to U. If Ω_1 and Ω_2 are sets of solutions mappings and $X \subseteq \mathsf{N_V}$, then:

$$\begin{array}{lcl} \Omega_1 \bowtie \Omega_2 & = & \{\omega_1 \cup \omega_2 \mid (\omega_1, \omega_2) \in \Omega_1 \times \Omega_2 \text{ and } \omega_1 \sim \omega_2\} \\ \Omega_1 \setminus \Omega_2 & = & \{\omega_1 \mid \omega_1 \in \Omega_1 \text{ and } \omega_1 \nsim \omega_2 \text{ for all } \omega_2 \in \Omega_2\} \\ \pi_X \Omega & = & \{\omega|_X \mid \omega \in \Omega\} \end{array}$$

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

We now reproduce the inductive definition of answers to a SPARQL query q over a graph $\mathcal{A}$, denoted $\mathsf{sparqlAns}(q, \mathcal{A})$, provided in [24] for the SUJO fragment (and for set semantics).

Definition 1 *(SPARQL answers over a plain graph).*

$$
\begin{array}{lcl}
\mathsf{sparqlAns}(t, \mathcal{A}) & = & \{\omega \mid \mathsf{dom}(\omega) = \mathsf{vars}(t) \text{ and } \omega(t) \in \mathcal{A}\} \\
\mathsf{sparqlAns}(q_1 \text{ UNION } q_2, \mathcal{A}) & = & \mathsf{sparqlAns}(q_1, \mathcal{A}) \cup \mathsf{sparqlAns}(q_2, \mathcal{A}) \\
\mathsf{sparqlAns}(q_1 \text{ JOIN } q_2, \mathcal{A}) & = & \mathsf{sparqlAns}(q_1, \mathcal{A}) \bowtie \mathsf{sparqlAns}(q_2, \mathcal{A}) \\
\mathsf{sparqlAns}(q_1 \text{ OPT } q_2, \mathcal{A}) & = & (\mathsf{sparqlAns}(q_1, \mathcal{A}) \bowtie \mathsf{sparqlAns}(q_2, \mathcal{A})) \cup \\
 & & (\mathsf{sparqlAns}(q_1, \mathcal{A}) \setminus \mathsf{sparqlAns}(q_2, \mathcal{A})) \\
\mathsf{sparqlAns}(\text{SELECT}_X\ q, \mathcal{A}) & = & \pi_X\ \mathsf{sparqlAns}(q, \mathcal{A})
\end{array}
$$

2.2. Description logic KB, UCQs 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 $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, 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 $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, the *active domain* of $\mathcal{K}$, denoted with $\mathsf{aDom}(\mathcal{K})$, is the set of elements of $\mathsf{N_I}$ 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 $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where the *domain* $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty subset of U, and the *interpretation function* $\cdot^{I}$ of $\mathcal{I}$ maps each $c \in \mathsf{N_I}$ to itself, and each $A \in \mathsf{N_C}$ (resp. $r \in \mathsf{N_R}$) to a unary (resp, binary) relation A^I (resp. $r^{\mathcal{I}}$) over $\Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ 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 $\mathsf{mod}(\mathcal{K})$ 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 $\mathcal{I}_c \in \mathsf{mod}(\mathcal{K})$ is a *canonical model* of $\mathcal{K}$ if $\mathcal{I}_c$ can be homomorphically mapped to any $\mathcal{I} \in \mathsf{mod}(\mathcal{K})$. 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*$_{\mathcal{R}}$, $\mathcal{EL}$ or Horn-$\mathcal{SHIQ}$, have this property [10].

An interpretation can also be viewed as a (possibly infinite) RDF graph, with triples $\{A(d) \mid d \in A^{\mathcal{I}}, A \in \mathsf{N_C}\} \cup \{r(d_1, d_2) \mid (d_1, d_2) \in r^{\mathcal{I}}, r \in \mathsf{N_R}\}$. 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 a expression of the form:

$$h(\mathbf{x}) \leftarrow p_1(\mathbf{x}_1), \ldots, p_m(\mathbf{x}_m)$$

where h, p_i are predicates and $\mathbf{x}, \mathbf{x}_i$ are tuple over $\mathsf{N_V}$. Abusing notation, we may use $\mathbf{x}$ (resp. $\mathbf{x}_i$) below to designate the elements of $\mathbf{x}$ (resp. $\mathbf{x}_i$) viewed as a set. An additional syntactic requirement on a CQ is that $\mathbf{x} \subseteq \mathbf{x}_1 \cup .. \cup \mathbf{x}_m$. The variables in $\mathbf{x}$ are called *distinguished*, and we use $\mathsf{vars}(h)$ 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. $p_i \in \mathsf{N_C} \cup \mathsf{N_R}$. A *match* for h in an interpretation $\mathcal{I}$ is a total function ρ from $\mathbf{x}_1 \cup \ldots \cup \mathbf{x}_m$ to $\Delta^{\mathcal{I}}$ such that $\rho(\mathbf{x}_i) \in (p_i)^{\mathcal{I}}$ for $i \in \{1..m\}$. A mapping ω is an *answer* to h over $\mathcal{I}$ iff there is a match ρ for h in $\mathcal{I}$ s.t. $\omega = \rho|_{\mathsf{vars}(h)}$.

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

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

CQs and UCQs have a straightforward representation as SPARQL queries. The CQ $h(\mathbf{x}) \leftarrow p_1(\mathbf{x}_1), \ldots, p_m(\mathbf{x}_m)$ in SPARQL syntax is written:

$$\text{SELECT}_{\mathbf{x}}\ (p_1(\mathbf{x}_1) \text{ JOIN } .. \text{ JOIN } p_m(\mathbf{x}_m))$$

And a UCQ in SPARQL syntax is of the form:

$$h_1 \text{ UNION } .. \text{ UNION } h_n$$

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

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 $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ 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 $C(a)$, 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. $\exists \texttt{hasLicense}(x)$.

Consider again Example 1 under the OWL 2 QL entailment regime for instance, which corresponds (roughly) to the DL *DL-Lite*$_\mathcal{R}$. In this example, the query $\exists \texttt{hasLicense}(x)$ has $\{x \mapsto \texttt{Alice}\}$ as unique answer: since the entailed graph contains all ABox assertions entailed by $\mathcal{K}$, it contains the assertion $\exists \texttt{hasLicense}(\texttt{Alice})$ (again, we use the DL syntax rather than OWL, for readability).

So the expressivity of the $\mathcal{L}$-entailment regime is limited by the concepts that can be expressed in $\mathcal{L}$. 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*$_\mathcal{R}$ (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 $\text{SELECT}_{\{x\}}\texttt{hasLicense}(x, y)$ has no answer over the KB of Example 1, because the entailed graph does not contain any assertion of the form $\texttt{hasLicense}(\texttt{Alice}, e)$.

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.
$\mathcal{A} = \{\texttt{Person(Alice)}\}$
$q = \texttt{Person}(x)\ \text{OPT}\ \texttt{hasLicense}(x, y)$

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

Then in [2] and [1] still, the authors remark that in this example, ω can nonetheless be *extended* to an answer in every model of $\langle \emptyset, \mathcal{A}\rangle$. 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 $\mathcal{I}$ an interpretation, let $\mathsf{eAns}(q, \mathcal{I})$ designate all mappings that can be extended to an answer to q in $\mathcal{I}$, i.e.:

$$\mathsf{eAns}(q, \mathcal{I}) = \{\omega \mid \omega \preceq \omega' \text{ for some } \omega' \in \mathsf{sparqlAns}(q, \mathcal{I})\}$$

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

$$\mathsf{eCertAns}(q, \mathcal{K}) = \bigcap_{\mathcal{I} \in \mathsf{mod}(\mathcal{K})} \mathsf{eAns}(q, \mathcal{I})$$

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

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).[2] Given a KB $\mathcal{K}$, the solution consists in retaining, for each maximal SJO subquery q', the *maximal* elements of $\mathsf{eCertAns}(q', \mathcal{K})$ w.r.t. $\preceq$. 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.
$\mathcal{A} = \{\texttt{teachesTo}(\texttt{Alice}, \texttt{Bob}), \texttt{knows}(\texttt{Bob}, \texttt{Carol}), \texttt{teachesTo}(\texttt{Alice}, \texttt{Dan})\}$
$q = \text{SELECT}_{\{x,z\}}(\texttt{teachesTo}(x, y)\ \text{OPT}\ \texttt{knows}(y, z))$

In this example, $\mathsf{sparqlAns}(q, \mathcal{A}) = \{\{x \mapsto \texttt{Alice}, z \mapsto \texttt{Carol}\}, \{x \mapsto \texttt{Alice}\}\}$. Instead, the semantics proposed in [1] yields $\{\{x \mapsto \texttt{Alice}, z \mapsto \texttt{Carol}\}\}$.

Sections 5.3 and 5.4 below define different semantics for evaluating a SPARQL query over a KB, which coincides not only with certain answers for UCQs (as opposed to the SPARQL entailment regimes and [16]), but also with the SPARQL specification in the case where the TBox is empty (as opposed to the proposal made in [1]).

Before continuing, other works need 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 met by any semantics whose purpose is to conciliate certain answers and SPARQL answers. We also define two additional requirements (called OPT *extension* and *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 $\mathsf{ans}(q, \mathcal{K})$ 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 $\mathsf{ans}(q, \mathcal{K})$ should coincide with certain answers for UCQs.

Requirement 1 *(Certain answer compliance).* For any UCQ q and KB $\mathcal{K}$,

$$\mathsf{ans}(q, \mathcal{K}) = \mathsf{certAns}(q, \mathcal{K})$$

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

Requirement 2 *(SPARQL answer compliance).* For any query q and ABox $\mathcal{A}$,

$$\mathsf{ans}(q, \langle \emptyset, \mathcal{A} \rangle) = \mathsf{sparqlAns}(q, \mathcal{A})$$

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 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 ω to the left operand of an OPT, either ω or some extension of ω is expected be present in the answers to the whole expression. Let $\preceq_g$ be the partial order over sets of solution mappings defined by $\Omega_1 \preceq_g \Omega_2$ 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 *(OPT extension).* For any queries q_1, q_2 and KB $\mathcal{K}$:

$$\mathsf{ans}(q_1, \mathcal{K}) \preceq_g \mathsf{ans}(q_1 \text{ OPT } q_2, \mathcal{K})$$

Another important property of SPARQL answers over plain graphs pertains to bound variables. Indeed, a SPARQL query q (with UNION and/or OPT) may allow *partial* solution mappings, i.e. whose domain does not cover all variables projected by q. For instance, in Example 2, $\omega = \{x \mapsto \texttt{Alice}\} \in \mathsf{sparqlAns}(q, \mathcal{A})$, even though the variables projected by q are x and y. In such a case, we say that variable x is *bound* by ω, whereas variable y is not. Then a SPARQL query may only admit answers that bind certain sets of variables. For instance the query $A(x)$ OPT $(R(x, y)$ JOIN $R(y, z))$ admits answers that bind either $\{x\}$ or $\{x, y, z\}$. But it does not admit answers that bind another set of variables ($\{y\}$, $\{x, y\}$, 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 ω s.t. $\omega \in \mathsf{sparqlAns}(q, \mathcal{A})$ and $\mathsf{dom}(\omega) = X$. Unfortunately, for queries with 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 $\mathsf{adm}(q)$ to denote the family of sets of variables defined inductively as follows:

Definition 4 *(Definition of $\mathsf{adm}(q)$ for the SUJO fragment).*

$$\begin{array}{ll}
\mathsf{adm}(t) & = \{\mathsf{vars}(t)\} \\
\mathsf{adm}(\text{SELECT}_X\ q) & = \{\ X \cap X' \quad \mid X' \in \mathsf{adm}(q)\ \} \\
\mathsf{adm}(q_1 \text{ JOIN } q_2) & = \{\ X_1 \cup X_2 \quad \mid (X_1, X_2) \in \mathsf{adm}(q_1) \times \mathsf{adm}(q_2)\ \} \\
\mathsf{adm}(q_1 \text{ OPT } q_2) & = \mathsf{adm}(q_1) \cup \mathsf{adm}(q_1 \text{ JOIN } q_2) \\
\mathsf{adm}(q_1 \text{ UNION } q_2) & = \mathsf{adm}(q_1) \cup \mathsf{adm}(q_2)
\end{array}$$

We can now formulate the corresponding property, as follows:

Property 1 *(Variable binding). For any SUJO query q, KB $\mathcal{K}$ and $\omega \in \mathsf{ans}(q, \mathcal{K})$:*

$$\mathsf{dom}(\omega) \in \mathsf{adm}(q)$$

This constraint on variable bindings is still arguably weak though, if one consider queries with UNION. Take for instance the query $q = A(x)$ UNION $R(x, y)$. Then $\mathsf{adm}(q) = \{\{x\}, \{x, y\}\}$. But the semantics of SPARQL over plain graphs puts a stronger requirement on variable bindings. If ω is a solution to q, then ω may bind $\{x\}$ only if ω is an answer to the left operand $A(x)$, and ω may bind $\{x, y\}$ only if ω 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 ω:

$$\text{if } \omega \in \mathsf{ans}(q_1 \text{ UNION } q_2, \mathcal{K}) \text{ and } \omega \notin \mathsf{ans}(q_2, \mathcal{K}), \text{ then } \mathsf{dom}(\omega) \in \mathsf{adm}(q_1)$$

$$\text{if } \omega \in \mathsf{ans}(q_1 \text{ UNION } q_2, \mathcal{K}) \text{ and } \omega \notin \mathsf{ans}(q_1, \mathcal{K}), \text{ then } \mathsf{dom}(\omega) \in \mathsf{adm}(q_2)$$

Table 1
Requirements met by alternative semantics for SPARQL over a DL KB (with the canonical model property). "A/B" stands for all fragments between A and B.

Semantics	Fragment	REQ1	REQ2	REQ3	REQ4
Ahmetaj et al. ([1])	pwdPT ($\subseteq$ SJO)	✓	x	?	✓
Entailment regime (Definition 5)	UJO	✓	✓	✓	✓
	SJ / SUJO	x	✓	✓	✓
Canonical (Definition 6)	O / SUJO	✓	✓	x	✓
Restricted (Definition 7)	SUJO	✓	✓	✓	x
Maximal admissible canonical answers (Definition 10) Epistemic certain answers (Definition 12)	SUJO	✓	✓	✓	✓

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_1, KB $\mathcal{K}$ and solution mapping ω s.t. $\omega \in \mathsf{ans}(q_1, \mathcal{K})$ and $\mathsf{dom}(\omega) \notin \mathsf{adm}(q_1)$. Now let q_2 be a SUJO query whose signature is disjoint with the signature of $\mathcal{K}$ (i.e. none of the predicates in q_2 appears in $\mathcal{K}$). Then $\omega \notin \mathsf{ans}(q_2, \mathcal{K}) = \emptyset$. Therefore, q_1, q_2, $\mathcal{K}$ and ω 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 OPT with JOIN, UNION and a MINUS operator. It consists in evaluating the right operand of 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 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 RDF/RDFS/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 $\mathcal{K}$ a KB, we call the resulting set of solution mapping the *entailment regime answers* to q over $\mathcal{K}$, denoted with $\mathsf{eRAns}(q, \mathcal{K})$, defined as follows:

Definition 5 *(Entailment regime answers).*

$$
\begin{array}{lcl}
\mathsf{eRAns}(t, \mathcal{K}) & = & \mathsf{certAns}(t, \mathcal{K}) \\
\mathsf{eRAns}(q_1 \text{ UNION } q_2, \mathcal{K}) & = & \mathsf{eRAns}(q_1, \mathcal{K}) \cup \mathsf{eRAns}(q_2, \mathcal{K}) \\
\mathsf{eRAns}(q_1 \text{ JOIN } q_2, \mathcal{K}) & = & \mathsf{eRAns}(q_1, \mathcal{K}) \bowtie \mathsf{eRAns}(q_2, \mathcal{K}) \\
\mathsf{eRAns}(q_1 \text{ OPT } q_2, \mathcal{K}) & = & (\mathsf{eRAns}(q_1, \mathcal{K}) \bowtie \mathsf{eRAns}(q_2, \mathcal{K})) \cup \\
 & & (\mathsf{eRAns}(q_1, \mathcal{K}) \setminus \mathsf{eRAns}(q_2, \mathcal{K})) \\
\mathsf{eRAns}(\text{SELECT}_X\ q, \mathcal{K}) & = & \pi_X\ \mathsf{eRAns}(q, \mathcal{K})
\end{array}
$$

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 $\mathcal{A}$, $\mathsf{sparqlAns}(q, \mathcal{A}) = \mathsf{certAns}(q, \langle \emptyset, \mathcal{A} \rangle)$. Then the inductive definitions of $\mathsf{sparqlAns}(q, \mathcal{A})$ (Definition 1) and $\mathsf{eRAns}(q, \mathcal{K})$ (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 UNION operator is not compositional for certain answers in some DLs. Consider for instance Example 4 below:

Example 4.
$\mathcal{A} = \{\texttt{Driver(Alice)}\}$
$\mathcal{T} = \{\texttt{Driver} \sqsubseteq \texttt{CarDriver} \sqcup \texttt{TruckDriver}\}$
$q = \texttt{CarDriver}(x)$ UNION $\texttt{TruckDriver}(x)$
Then $\mathsf{certAns}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \{\{x \mapsto \texttt{Alice}\}\}$, but $\mathsf{eRAns}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$.

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.
$\mathcal{A} = \{\texttt{Driver(Alice)}\}$
$\mathcal{T} = \{\texttt{Driver} \sqsubseteq \exists \texttt{hasLicense}\}$
$q = \text{SELECT}_{\{x\}}$ $(\texttt{Driver}(x)$ JOIN $\texttt{hasLicense}(x, y))$
Then $\mathsf{certAns}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \{\{x \mapsto \texttt{Alice}\}\}$, but $\mathsf{eRAns}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$.

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 $\mathcal{L}_{\mathsf{can}}$ with the canonical model property, and use "an $\mathcal{L}_{\mathsf{can}}$ KB" to refer to a KB in such DL. Then if $\mathcal{K}$ is an $\mathcal{L}_{\mathsf{can}}$ KB, we use $\mathsf{can}(\mathcal{K})$ 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 $\mathsf{can}(\mathcal{K})$, restricted to those that range over $\mathsf{aDom}(\mathcal{K})$. We show that extending this definition to queries with OPT is sufficient to satisfy Requirement 2 (in addition to Requirement 1), but fails to satisfy Requirement 3.

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

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

$$\mathsf{canAns}(q, \mathcal{K}) = \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K})) \rhd \mathsf{aDom}(\mathcal{K})$$

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

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

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

Example 6.
$\mathcal{A} = \{\texttt{Driver(Alice)}\}$
$\mathcal{T} = \{\texttt{Driver} \sqsubseteq \exists \texttt{hasLicense}\}$
$q = \texttt{Driver}(x)$ OPT $\texttt{hasLicense}(x, y)$

In this example, Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. Then $\mathsf{canAns}(\texttt{Driver}(x), \mathcal{K}) = \{\{x \mapsto \texttt{Alice}\}\}$. However, $\mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K})) = \{\{x \mapsto \texttt{Alice}, y \mapsto e\}\}$, for some $e \notin \mathsf{aDom}(\mathcal{K})$. Therefore $\mathsf{canAns}(q, \mathcal{K}) = \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K})) \rhd \mathsf{aDom}(\mathcal{K}) = \emptyset$. So $\mathsf{canAns}(\texttt{Driver}(x), \mathcal{K}) \not\preceq_g \mathsf{canAns}(q, \mathcal{K})$, 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 $\mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}))$ to solution mappings that range over $\mathsf{can}(\mathcal{K})$ is too strong. Consider again Example 6, where $\mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K})) = \{\{x \mapsto \texttt{Alice}, y \mapsto e\}\}$. In this example, rather than filtering out this solution mapping (because it does not range over $\mathsf{aDom}(\mathcal{K})$), one would like instead to *restrict* it to the active domain, which yields the desired mapping $\{x \mapsto \texttt{Alice}\}$.

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

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

$$\mathsf{restCanAns}(q, \mathcal{K}) = \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K})) \blacktriangleright \mathsf{aDom}(\mathcal{K})$$

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.
$\mathcal{A} = \{\texttt{Teacher(Alice)}\}$
$\mathcal{T} = \{\texttt{Teacher} \sqsubseteq \exists \texttt{teachesTo}, \texttt{teachesTo} \sqsubseteq \texttt{hasTeacher}^-\}$
$q = \texttt{Teacher}(x)$ OPT $(\texttt{teachesTo}(x, y)$ JOIN $\texttt{hasTeacher}(y, z))$

In this example, $\mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K})) = \{\{x \mapsto \texttt{Alice}, y \mapsto e, z \mapsto \texttt{Alice}\}\}$, for some $e \notin \mathsf{aDom}(\mathcal{K})$. So restricting this solution mapping to $\mathsf{aDom}(\mathcal{K})$ would yield the mapping $\{x \mapsto \texttt{Alice}, z \mapsto \texttt{Alice}\}$. 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 *maximal admissible canonical answers*, noted $\mathsf{mCanAns}(q, \mathcal{K})$.

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

$$\Omega \otimes \mathcal{X} = \{\omega|_X \mid \omega \in \Omega, X \in \max_{\subseteq}(\mathcal{X} \cap 2^{\mathsf{dom}(\omega)})\}$$

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

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

$$\mathsf{mCanAns}(q, \mathcal{K}) = \mathsf{restCanAns}(q, \mathcal{K}) \otimes \mathsf{adm}(q)$$

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.
$\mathcal{A} = \{\texttt{Teacher(Alice)}\}$
$\mathcal{T} = \{\texttt{Teacher} \sqsubseteq \exists \texttt{teachesTo}, \texttt{teachesTo} \sqsubseteq \texttt{hasTeacher}^-\}$
$q = \big(\texttt{Teacher}(x)$ OPT $(\texttt{teachesTo}(x, y)$ JOIN $\texttt{hasTeacher}(y, z))\big)$
UNION $\texttt{hasLicense}(x, z)$
As opposed to Example 7, $\{x, z\}$ is now an admissible set of variables for q (i.e. $\{x, z\} \in \mathsf{adm}(q)$), due to the right operand $\texttt{hasLicense}(x, z)$ of the UNION operator. As a result, the undesired mapping $\{x \mapsto \texttt{Alice}, z \mapsto \texttt{Alice}\}$ is now in $\mathsf{restCanAns}(q, \mathcal{K}) \otimes \mathsf{adm}(q)$, even though this mapping is returned by the left operand of the UNION, for which $\{x, z\}$ is not admissible.

So in order to generalize Definition 8 to queries with 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 *branches* of a SUJO query q, denoted with $\mathsf{branches}(q)$, as the set of SJO queries that may produce a solution to q, by intuitively "choosing" one operand of each UNION. For instance, if $q = A(x)$ OPT $(R_1(x, y)$ UNION $R_2(x, z))$, then $\mathsf{branches}(q) = \{A(x)$ OPT $R_1(x, y), A(x)$ OPT $R_2(x, z)\}$. The function $\mathsf{branches}(q)$ is defined inductively over q as expected:

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

$$\begin{array}{ll}
\mathsf{branches}(t) & = \{t\} \\
\mathsf{branches}(\text{SELECT}_X\ q) & = \{\ \text{SELECT}_X\ q' \mid q' \in \mathsf{branches}(q)\ \} \\
\mathsf{branches}(q_1\ \text{JOIN}\ q_2) & = \{\ q_1'\ \text{JOIN}\ q_2' \mid (q_1', q_2') \in \mathsf{branches}(q_1) \times \mathsf{branches}(q_2)\ \} \\
\mathsf{branches}(q_1\ \text{OPT}\ q_2) & = \{\ q_1'\ \text{OPT}\ q_2' \mid (q_1', q_2') \in \mathsf{branches}(q_1) \times \mathsf{branches}(q_2)\ \} \\
\mathsf{branches}(q_1\ \text{UNION}\ q_2) & = \mathsf{branches}(q_1) \cup \mathsf{branches}(q_2)
\end{array}$$

According to the semantics of 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 $\mathcal{A}$:

$$\mathsf{sparqlAns}(q, \mathcal{A}) \subseteq \bigcup_{q' \in \mathsf{branches}(q)} \mathsf{sparqlAns}(q', \mathcal{A})$$

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

$$\mathsf{sparqlAns}(q, \mathcal{A}, q') = \mathsf{sparqlAns}(q, \mathcal{A}) \cap \mathsf{sparqlAns}(q', \mathcal{A})$$

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

$$\mathsf{mCanAns}(q, \mathcal{K}, q') = (\mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}), q') \blacktriangleright \mathsf{aDom}(\mathcal{K})) \otimes \mathsf{adm}(q')$$

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

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

$$\mathsf{mCanAns}(q, \mathcal{K}) = \bigcup_{q' \in \mathsf{branches}(q)} \mathsf{mCanAns}(q, \mathcal{K}, q')$$

It can be easily verified that Definitions 8 and 10 coincide for SJO queries, since in this case $\mathsf{branches}(q) = \{q\}$.

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 $\mathcal{L}_{\mathsf{can}}$ KB $\mathcal{K}$, $\mathsf{mCanAns}(q, \mathcal{K})$ 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 OPT operator over a graph $\mathcal{A}$ is defined by means of the "\" operator between sets of solution mappings. So in a sense, the 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 MINUS in what follows. And we will use the letter "M" (in addition to S, U, J, and O) to designate queries with MINUS. We will also use an additional base case: as an alternative to a triple pattern, we allow a (finite) set Ω of solution mappings, similar to the `VALUES` operator of SPARQL. Hence we use the letter "V" to designate the corresponding queries.

So the syntax of SUJOMV queries is:

$$q \quad ::= \quad \Omega \mid t \mid \text{SELECT}_X\ q \mid q\ \text{UNION}\ q \mid q\ \text{JOIN}\ q \mid q\ \text{OPT}\ q \mid q\ \text{MINUS}\ q$$

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

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

$$\mathsf{sparqlAns}(\Omega, \mathcal{A}) = \Omega$$

$$\mathsf{sparqlAns}(q_1\ \text{MINUS}\ q_2, \mathcal{A}) = \mathsf{sparqlAns}(q_1, \mathcal{A}) \setminus \mathsf{sparqlAns}(q_2, \mathcal{A})$$

Then a SUJO query can be transformed into an equivalent SUJM query, by replacing inductively each "q_1 OPT q_2" with "(q_1 JOIN q_2) UNION (q_1 MINUS q_2)". "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 MINUS under certain answer semantics: if $q = q_1$ MINUS q_2 is a (sub)-query, $\mathcal{K}$ a KB and $\mathcal{I}$ a model of $\mathcal{K}$, 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 $\mathcal{K}$. This intuition of is formalized below:

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

$$\mathsf{epCertAns}(q, \mathcal{K}) = \mathsf{certAns}(\mathsf{partEval}(q, \mathcal{K}), \mathcal{K}),$$

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

$$\begin{array}{lcl}
\mathsf{partEval}(t, \mathcal{K}) & = & t \\
\mathsf{partEval}(\text{SELECT}_X\ q, \mathcal{K}) & = & \text{SELECT}_X\ \mathsf{partEval}(q, \mathcal{K}) \\
\mathsf{partEval}(q_1\ \text{JOIN}\ q_2, \mathcal{K}) & = & \mathsf{partEval}(q_1, \mathcal{K})\ \text{JOIN}\ \mathsf{partEval}(q_2, \mathcal{K}) \\
\mathsf{partEval}(q_1\ \text{UNION}\ q_2, \mathcal{K}) & = & \mathsf{partEval}(q_1, \mathcal{K})\ \text{UNION}\ \mathsf{partEval}(q_2, \mathcal{K}) \\
\mathsf{partEval}(q_1\ \text{OPT}\ q_2, \mathcal{K}) & = & \mathsf{partEval}(q_1\ \text{JOIN}\ q_2, \mathcal{K})\ \text{UNION} \\
 & & (\mathsf{partEval}(q_1, \mathcal{K})\ \text{MINUS}\ \mathsf{epCertAns}(q_2, \mathcal{K}))
\end{array}$$

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

Example 9.

$\mathcal{A} = \{\texttt{Driver(Alice)},\ \texttt{Driver(Bob)},\ \texttt{hasLicense(Bob,b)}\}$

$\mathcal{T} = \{\texttt{Driver} \sqsubseteq \exists \texttt{hasLicense}\}$

$q = \texttt{Driver}(x)\ \text{OPT}\ \texttt{hasLicense}(x, y)$

$$\begin{array}{lll}
\text{First,} & \mathsf{epCertAns}(\texttt{hasLicense}(x, y), \mathcal{K}) = \mathsf{certAns}(\mathsf{partEval}(\texttt{hasLicense}(x, y), \mathcal{K})) \\
= & \mathsf{certAns}(\texttt{hasLicense}(x, y), \mathcal{K}) = \{\{x \mapsto \texttt{Bob}, y \mapsto \texttt{b}\}\}. \\
\text{Then} & \mathsf{epCertAns}(q, \mathcal{K}) = \mathsf{certAns}(\mathsf{partEval}(q, \mathcal{K}), \mathcal{K}) \\
= & \mathsf{certAns}(\\
 & \quad \mathsf{partEval}(\texttt{Driver}(x)\ \text{JOIN}\ \texttt{hasLicense}(x, y), \mathcal{K}) \\
 & \quad \text{UNION} \\
 & \quad (\mathsf{partEval}(\texttt{Driver}(x), \mathcal{K})\ \text{MINUS}\ \mathsf{epCertAns}(\texttt{hasLicense}(x, y), \mathcal{K})), \mathcal{K}) \\
= & \mathsf{certAns}(\\
 & \quad (\texttt{Driver}(x)\ \text{JOIN}\ \texttt{hasLicense}(x, y)) \\
 & \quad \text{UNION} \\
 & \quad (\texttt{Driver}(x)\ \text{MINUS}\ \{\{x \mapsto \texttt{Bob}, y \mapsto \texttt{b}\}\}), \mathcal{K}) \\
= & \{\{x \mapsto \texttt{Bob}, y \mapsto \texttt{b}\}, \{x \mapsto \texttt{Alice}\}\}.
\end{array}$$

The KB in this example admits a canonical model, so the canonical answers $\mathsf{canAns}(q, \mathcal{K})$ are defined. But neither $\mathsf{canAns}(q, \mathcal{K})$ nor $\mathsf{certAns}(q, \mathcal{K})$ contains the mapping $\{x \mapsto \texttt{Alice}\}$. Instead, under epCertAns, the right operand $\texttt{hasLicense}(x, y)$ of the MINUS operator is first evaluated under certain answer semantics. The result $\{\{x \mapsto \texttt{Bob}, y \mapsto \texttt{b}\}\}$ does not contain any answer that maps x to Alice, because variable y cannot be mapped to the same license for Alice in every model of $\mathcal{K}$. This way the mapping $\{\{x \mapsto \texttt{Alice}\}\}$ is not discarded when evaluating the MINUS operator. Intuitively, this allows distinguishing the "anonymous" license of 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 $\mathcal{K}$,* $\mathsf{epCertAns}(q, \mathcal{K})$ *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 **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 $\langle \mathcal{K}, \mathcal{P} \rangle$, where $\mathcal{K}$ is DL KB and $\mathcal{P}$ a set of rules, and epistemic certain answers may be captured by means of a **K** operator. For instance, Example 6 can be encoded as a knowledge base $\langle \langle \mathcal{T}, \mathcal{A} \rangle, \mathcal{P} \rangle$, where $\mathcal{P}$ is the set of rules:

$$\begin{aligned} \mathrm{ans}(x, y) &\leftarrow \mathbf{K}\, q(x, y) \\ q(x, y) &\leftarrow \texttt{Driver}(x), \texttt{hasLicense}(x, y) \\ q(x, \mathrm{NULL}) &\leftarrow \texttt{Driver}(x), \neg \mathbf{K}\, \texttt{hasLicense}(x, y) \end{aligned}$$

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.

epCertAns *over* $\mathcal{L}_{\mathsf{can}}$ *KBs.* Epistemic certain answers are defined for arbitrary KBs. However, for the case where the underlying KB $\mathcal{K}$ admits a canonical model, a key property of this semantics is the following: the epistemic certain answers to a query q coincide with the answers to $\mathsf{partEval}(q, \mathcal{K})$ over the canonical model of $\mathcal{K}$, restricted to solution mappings that range over the active domain. Or formally (the proof is in Appendix D):

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

$$\mathsf{epCertAns}(q, \mathcal{K}) = \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathsf{can}(\mathcal{K})) \triangleright \mathsf{aDom}(\mathcal{K})$$

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

5.5. Comparing maximal canonical answers and epistemic certain answers

Both maximal canonical answers (mCanAns, defined in Section 5.3) and epistemic certain answers (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 mCanAns is only defined for DLs with the canonical model property, whereas epCertAns is defined for arbitrary DLs. So we restrict the comparison to DLs where both are defined, i.e. for $\mathcal{L}_{\mathsf{can}}$ KBs. Proposition 5 shows that the two semantics differ for SUJO queries.

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

Proof. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, with $\mathcal{T} = \{B \sqsubseteq \exists R\}$ and $\mathcal{A} = \{A(a), B(b)\}$, and the query $q = A(x_1)$ JOIN $\big(B(x_2)$ OPT $R(x_2, x_1)\big)$. Then $\mathsf{mCanAns}(q, \mathcal{K}) = \emptyset$, whereas $\mathsf{epCertAns}(q, \mathcal{K}) = \{\{x_1 \mapsto a, x_2 \mapsto b\}\}$. □

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

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

$$\mathsf{mCanAns}(q, \mathcal{K}) = \mathsf{epCertAns}(q, \mathcal{K})$$

Interestingly, the 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.

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.

Fragment	$\text{EVAL}_{\text{sparqlAns}}$	$\text{EVAL}_{\text{mCanAns}}$	$\text{EVAL}_{\text{epCertAns}}$
UJ/SUJ	NP-c	NP-c	NP-c
well-designed JO	coNP-c	coNP-c	coNP-h / in PSPACE
well-designed SJO*	Σ_2^{P}-c	Σ_2^{P}-c	Σ_2^{P}-h / in PSPACE
OJ/SUJO	PSPACE-c	PSPACE-c	PSPACE-c

6. Complexity

We now provide complexity results for query answering under the semantics defined in Sections 5.3 and 5.4, for different sub-fragments of the SUJO fragment, and focusing on KBs in *DL-Lite*$_{\mathcal{R}}$ [3], a DL tailored for query answering, which corresponds to the OWL 2 QL 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.

$\text{EVAL}_{\text{mCanAns}}$
Input: *DL-Lite*$_{\mathcal{R}}$ KB $\mathcal{K}$, query q, mapping ω
Decide: $\omega \in \text{mCanAns}(q, \mathcal{K})$

$\text{EVAL}_{\text{epCertAns}}$
Input: *DL-Lite*$_{\mathcal{R}}$ KB $\mathcal{K}$, query q, mapping ω
Decide: $\omega \in \text{epCertAns}(q, \mathcal{K})$

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 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.

$\text{EVAL}_{\text{sparqlAns}}$
Input: graph $\mathcal{A}$, query q, mapping ω
Decide: $\omega \in \text{sparqlAns}(q, \mathcal{A})$

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 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 PSPACE to Σ_2^{P} and 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 JOIN or 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.

6.1. Complexity of maximal admissible canonical answers

Interestingly, for $\text{EVAL}_{\text{mCanAns}}$, all upper bounds are matched. This means that for these fragments and this semantics, the presence of a TBox does not induce an extra computational cost (as far as worst-case complexity is concerned) when compared to SPARQL answers over a plain graph. This observation is analogous to well-known results for answering UCQs under certain-answer semantics over a *DL-Lite*$_{\mathcal{R}}$ KB [7], which matches the (NP) upper bound for UCQs over a plain graph.

Before explaining these results, we isolate a key observation:

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 $X_1 \in \max_{\subseteq}(\text{adm}(q) \cap 2^{X_2})$*

Proof. (Sketch.) If q is a JO query, we compute a family $\text{base}(q)$ 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 $\text{base}(q)$ and conversely, i.e. $\text{adm}(q) = \{\bigcup \mathcal{B} \mid \mathcal{B} \in 2^{\text{base}(q)}\}$. The family $\text{base}(q)$ can be computed inductively as follows:

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

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

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

We now sketch the argument used to derive upper bounds for the SUJO, well-designed SJO* and UJ fragments. For simplicity, we focus on the well-designed SJO* fragment. The argument for queries with UNION 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 Gaifman 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 $\text{EVAL}_{\mathsf{mCanAns}}$, $\langle \mathcal{K}, q, \omega \rangle$ is a positive instance iff $\omega \in \mathsf{mCanAns}(q, \mathcal{K})$, i.e. iff there is an ω' s.t. *(i)* $\omega = \omega'|_X$ for some $X \in \max_{\subseteq}(\mathsf{adm}(q) \cap 2^{\mathsf{dom}(\omega'\|_{\mathsf{aDom}(\mathcal{K})})})$ and *(ii)* $\omega' \in \mathsf{sparqlAns}(q, \mathcal{K})$.

So a (non-deterministic) procedure to verify $\omega \in \mathsf{mCanAns}(q, \mathcal{K})$ consists in guessing an extension ω' or ω, then verify *(i)*, and then verify *(ii)*. From Proposition 7 above, *(i)* can be verified in $O(|q|^2)$. For *(ii)*, if $\omega' \in \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}))$, from well-known properties of $\mathsf{can}(\mathcal{K})$ for *DL-Lite*$_\mathcal{R}$, it can be shown that:

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

So in order to verify *(ii)*, it is sufficient to guess $\mathcal{G}$, then verify that $\mathcal{G}$ is a subgraph of $\mathsf{can}(\mathcal{K})$, and then verify that $\omega' \in \mathsf{sparqlAns}(q, \mathcal{G})$. Since $\text{EVAL}_{\mathsf{sparqlAns}}$ is in Σ_2^P, $\omega' \in \mathsf{sparqlAns}(q, \mathcal{G})$ can be nondeterministically verified in time in $O(|q| + |\mathcal{G}| + |\omega'|) = O(|q| + |\mathcal{K}| + \omega)$ by some algorithm with an oracle for coNP problems. And a witness for this algorithm can be guessed together with $\mathcal{G}$ and ω' (without gaining a level in the polynomial hierarchy). We note that this last remark does not apply to the well-designed JO fragment: since $\text{EVAL}_{\mathsf{sparqlAns}}$ is coNP-hard, such a procedure would instead imply a quantifier alternation.

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

6.2. Complexity of epistemic certain answers

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

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

The proof of PSPACE-membership for the SUJO fragment can be found in Appendix G. First, since *DL-Lite*$_\mathcal{R}$ has the canonical model property, from Lemma 4, $\omega \in \mathsf{epCertAns}(q, \mathcal{K})$ can be decided by checking whether ω ranges over the active domain on the one hand, and deciding whether ω is an answer to $\mathsf{partEval}(q, \mathcal{K})$ over $\mathsf{can}(\mathcal{K})$ on the other hand. Then we define a top-down procedure that decides $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathsf{can}(\mathcal{K}))$, 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 KB on the one hand, and SPARQL answers over a plain KG 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 SPARQL with SELECT, UNION, JOIN and OPT. More precisely, we defined two semantics that satisfy these requirements, one with a more procedural flavor, defined for DLs with the canonical model property only, and the other with a more declarative flavor, defined for arbitrary DLs. We also provided combined complexity results for query answering over a *DL-Lite*$_\mathcal{R}$ KB under these semantics, for different (sub-)fragments of SPARQL.

As for future work, data complexity may also be investigated, as well as algorithmic aspects, in particular *first-order rewritability*, i.e. the possibility to rewrite a query over a KB into a query over its ABox only, which is a key property for OMQA/OBDA [28,29]. Other fragments of SPARQL and/or DLs may also be considered. E.g., after adding the SPARQL FILTER operator, we can study the satisfiability problem [32]. Finally, on the practical side, we are interested in implementing these semantics in an OBDA system like 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 (2017MMJJRE), by the European Regional Development Fund (ERDF) Investment for Growth and Jobs Programme 2014-2020 through the project IDEE (FESR1133), 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 $\mathcal{L}_{\mathsf{can}}$ KB $\mathcal{K}$,* $\mathsf{canAns}(q, \mathcal{K})$ *satisfies Requirement 2.*

Proof. Lemma 1 below states the proposition. □

Lemma 1. *For any UCQ q and ABox $\mathcal{A}$,*

$$\mathsf{canAns}(q, \langle \emptyset, \mathcal{A} \rangle) = \mathsf{sparqlAns}(q, \mathcal{A})$$

Proof. If $\mathcal{A}$ is an ABox, then $\mathsf{aDom}(\langle \emptyset, \mathcal{A} \rangle)$ is the set of constants appearing in $\mathcal{A}$.
In addition, $\mathsf{can}(\langle \emptyset, \mathcal{A} \rangle) = \mathcal{A}$.
So if q is a query, trivially, $\mathsf{sparqlAns}(q, \langle \emptyset, \mathcal{A} \rangle) \triangleright \mathsf{aDom}(\langle \emptyset, \mathcal{A} \rangle) = \mathsf{sparqlAns}(q, \langle \emptyset, \mathcal{A} \rangle)$.
So from Definition 6, $\mathsf{canAns}(q, \langle \emptyset, \mathcal{A} \rangle) = \mathsf{sparqlAns}(q, \langle \emptyset, \mathcal{A} \rangle)$. □

Appendix B. Proof of Proposition 2

Proposition 2. *For any SUJO query q and $\mathcal{L}_{\mathsf{can}}$ KB $\mathcal{K}$,* $\mathsf{mCanAns}(q, \mathcal{K})$ *satisfies Requirements 1, 2, 3 and 4.*

Proof. The proposition is split into Lemmas 2, 4, 5 and 6 below, one for each requirement. □

Lemma 2. *For any UCQ q and $\mathcal{L}_{\mathsf{can}}$ KB $\mathcal{K}$,* $\mathsf{mCanAns}(q, \mathcal{K}) = \mathsf{certAns}(q, \mathcal{K})$

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

Lemma 3. *For any UCQ q and $\mathcal{L}_{\mathsf{can}}$ KB $\mathcal{K}$,* $\mathsf{mCanAns}(q, \mathcal{K}) = \mathsf{canAns}(q, \mathcal{K})$

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

$$\begin{aligned}\mathsf{mCanAns}(q, \mathcal{K}, q') &= (\mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}), q') \blacktriangleright \mathsf{aDom}(\mathcal{K})) \otimes \mathsf{adm}(q') \\ &= (\mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}), q') \blacktriangleright \mathsf{aDom}(\mathcal{K})) \otimes \{\mathsf{vars}(q)\} \\ &= \{\omega\|_{\mathsf{aDom}(\mathcal{K})} \mid \omega \in \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}), q')\} \otimes \{\mathsf{vars}(q)\}\end{aligned}$$

Then for each $\omega \in \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}), q')$, $\omega \in \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}))$ must hold.
So since $\mathsf{adm}(q) = \{\mathsf{vars}(q)\}$, $\mathsf{dom}(\omega) = \mathsf{vars}(q)$ must hold as well.
Therefore $\mathsf{vars}(q) \subseteq \mathsf{dom}(\omega\|_{\mathsf{aDom}(\mathcal{K})})$ iff $\omega\|_{\mathsf{aDom}(\mathcal{K})} = \omega$, i.e. iff $\mathsf{range}(\omega) \subseteq \mathsf{aDom}(\mathcal{K})$. So from (B.1):

$$\mathsf{mCanAns}(q, \mathcal{K}, q') = \{\omega \in \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}), q') \mid \mathsf{range}(\omega) \subseteq \mathsf{aDom}(\mathcal{K})\} \tag{B.1}$$

$$\mathsf{mCanAns}(q, \mathcal{K}, q') = \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}), q') \triangleright \mathsf{aDom}(\mathcal{K}) \tag{B.2}$$

Finally:

$$\mathsf{mCanAns}(q, \mathcal{K}) = \bigcup_{q' \in \mathsf{branches}(q)} \mathsf{mCanAns}(q, \mathcal{K}, q') \tag{B.3}$$

So from (B.2) and (B.3):

$$\begin{aligned}\mathsf{mCanAns}(q,\mathcal{K}) &= \bigcup_{q'\in\mathsf{branches}(q)} \mathsf{sparqlAns}(q,\mathsf{can}(\mathcal{K}),q') \triangleright \mathsf{aDom}(\mathcal{K})\\ &= \bigcup_{q'\in\mathsf{branches}(q)} (\mathsf{sparqlAns}(q,\mathsf{can}(\mathcal{K})) \cap \mathsf{sparqlAns}(q',\mathsf{can}(\mathcal{K}))) \triangleright \mathsf{aDom}(\mathcal{K})\\ &= \left(\mathsf{sparqlAns}(q,\mathsf{can}(\mathcal{K})) \cap \bigcup_{q'\in\mathsf{branches}(q)} \mathsf{sparqlAns}(q',\mathsf{can}(\mathcal{K}))\right) \triangleright \mathsf{aDom}(\mathcal{K})\end{aligned}$$

And since:

$$\mathsf{sparqlAns}(q,\mathcal{K}) \subseteq \bigcup_{q'\in\mathsf{branches}(q)} \mathsf{sparqlAns}(q',\mathsf{can}(\mathcal{K}))$$

we get:

$$\mathsf{mCanAns}(q,\mathcal{K}) = \mathsf{sparqlAns}(q,\mathsf{can}(\mathcal{K})) \triangleright \mathsf{aDom}(\mathcal{K}) \quad \square$$

Lemma 4. *For any SUJO query q and ABox $\mathcal{A}$,*

$$\mathsf{mCanAns}(q,\langle\emptyset,\mathcal{A}\rangle) = \mathsf{sparqlAns}(q,\mathcal{A})$$

Proof. If $\mathcal{A}$ is an ABox, then $\mathsf{aDom}(\langle\emptyset,\mathcal{A}\rangle)$ is the set of constants appearing in $\mathcal{A}$.
In addition, $\mathsf{can}(\langle\emptyset,\mathcal{A}\rangle) = \mathcal{A}$.
Then from the definition of $\mathsf{mCanAns}(q,\mathcal{K},q')$:

$$\begin{aligned}\mathsf{mCanAns}(q,\langle\emptyset,\mathcal{A}\rangle,q') &= (\mathsf{sparqlAns}(q,\mathcal{A},q') \blacktriangleright \mathsf{aDom}(\langle\emptyset,\mathcal{A}\rangle)) \otimes \mathsf{adm}(q')\\ &= \mathsf{sparqlAns}(q,\mathcal{A},q') \otimes \mathsf{adm}(q') \qquad \text{(B.4)}\end{aligned}$$

Then since $\mathsf{sparqlAns}(q,\mathcal{A},q') \subseteq \mathsf{sparqlAns}(q',\mathcal{A})$, for each $\omega \in \mathsf{sparqlAns}(q,\mathcal{A},q')$, $\omega \in \mathsf{sparqlAns}(q',\mathcal{A})$ must hold.
So $\mathsf{dom}(\omega) \in \mathsf{adm}(q')$ must hold as well.
Therefore $\{\omega|_X \mid X \in \max_{\subseteq}(\mathsf{adm}(q') \cap 2^{\mathsf{dom}(\omega)}\} = \{\mathsf{dom}(\omega)\}$.
So from (B.4):

$$\mathsf{mCanAns}(q,\langle\emptyset,\mathcal{A}\rangle,q') = \mathsf{sparqlAns}(q,\mathcal{A},q') \qquad \text{(B.5)}$$

Finally:

$$\mathsf{mCanAns}(q,\langle\emptyset,\mathcal{A}\rangle) = \bigcup_{q'\in\mathsf{branches}(q)} \mathsf{mCanAns}(q,\langle\emptyset,\mathcal{A}\rangle,q') \qquad \text{(B.6)}$$

So from (B.5) and (B.6):

$$\begin{aligned}\mathsf{mCanAns}(q,\langle\emptyset,\mathcal{A}\rangle) &= \bigcup_{q'\in\mathsf{branches}(q)} \mathsf{sparqlAns}(q,\mathcal{A},q')\\ &= \bigcup_{q'\in\mathsf{branches}(q)} (\mathsf{sparqlAns}(q,\mathcal{A}) \cap \mathsf{sparqlAns}(q',\mathcal{A}))\\ &= \mathsf{sparqlAns}(q,\mathcal{A}) \cap \bigcup_{q'\in\mathsf{branches}(q)} \mathsf{sparqlAns}(q',\mathcal{A})\end{aligned}$$

And since:

$$\mathsf{sparqlAns}(q,\langle\emptyset,\mathcal{A}\rangle) \subseteq \bigcup_{q'\in\mathsf{branches}(q)} \mathsf{sparqlAns}(q',\mathcal{A})$$

we get:

$$\mathsf{mCanAns}(q,\langle\emptyset,\mathcal{A}\rangle) = \mathsf{sparqlAns}(q,\mathcal{A}) \quad \square$$

Lemma 5. *For any SUJO queries q_1, q_2 and $\mathcal{L}_{\mathsf{can}}$ KB $\mathcal{K}$:*

$$\mathsf{mCanAns}(q_1,\mathcal{K}) \preceq_g \mathsf{mCanAns}(q_1 \text{ OPT } q_2,\mathcal{K})$$

Proof. Let q_1, q_2 be SUJO queries, let $\mathcal{K}$ be an $\mathcal{L}_{\text{can}}$ KB, and let $\omega_1 \in \text{mCanAns}(q_1, \mathcal{K})$.
We need to show that there is an $\omega_2 \in \text{mCanAns}(q_1 \text{ OPT } q_2, \mathcal{K})$ s.t. $\omega_1 \preceq \omega_2$.
Since $\omega_1 \in \text{mCanAns}(q_1, \mathcal{K})$, there must be an SJO query $q' \in \text{branches}(q_1)$ s.t. $\omega_1 \in \text{mCanAns}(q_1, \mathcal{K}, q')$.
So there is a $\rho_1 \in \text{sparqlAns}(q_1, \text{can}(\mathcal{K})) \cap \text{sparqlAns}(q', \text{can}(\mathcal{K}))$ and an $X \in \max_{\subseteq}(\text{adm}(q') \cap 2^{\text{dom}(\rho_1)})$ s.t. $\omega_1 = \rho_1|_X$.
Since $\rho_1 \in \text{sparqlAns}(q', \text{can}(\mathcal{K}))$, from Definition 1, there must be a $\rho_2 \in \text{sparqlAns}(q' \text{ OPT } q_2, \text{can}(\mathcal{K}))$ s.t. $\rho_1 \preceq \rho_2$.
We first show that $\rho_2 \in \text{sparqlAns}(q_1 \text{ OPT } q_2, \text{can}(\mathcal{K}))$ must hold.
For this, we distinguish two cases:

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

Now because $\rho_1 \preceq \rho_2$, $\text{dom}(\rho_1) = X \subseteq \text{dom}(\rho_2)$.
And since $X \in \text{adm}(q')$, $X \in \text{adm}(q') \cap 2^{\text{dom}(\rho_2)}$ holds.
So there must be an X' s.t. $X \subseteq X'$ and $X' \in \max_{\subseteq}(\text{adm}(q') \cap 2^{\text{dom}(\rho_2)})$.
Finally, because $q' \in \text{branches}(q_1)$, from Definition 9, $q' \in \text{branches} q_1 \text{ OPT } q_2$.
So from Definition 8, $\rho_2|_{X'} \in \text{mCanAns}(q_1 \text{ OPT } q_2, \mathcal{K})$.
Now let $\omega_2 = \rho_2|_{X'}$.
To complete the proof, we only need to show that $\omega_1 \preceq \omega_2$.
First, since $\omega_1 = \rho_1|_X$, $\omega_1 \preceq \rho_1$ must hold.
Then from the definition of ρ_2, $\rho_1 \preceq \rho_2$.
So from the transitivity of $\preceq$, $\omega_1 \preceq \rho_2$.
Finally, since $X \subseteq X'$, $\omega_1|_X \preceq \rho_2|_{X'}$ must hold, i.e. $\omega_1 \preceq \omega_2$. □

Lemma 6. *For any queries q_1, q_2, $\mathcal{L}_{\text{can}}$ KB $\mathcal{K}$ and solution mapping ω:*

$$\text{if } \omega \in \text{mCanAns}(q_1 \text{ UNION } q_2) \text{ and } \omega \notin \text{mCanAns}(q_2), \text{ then } \text{dom}(\omega) \in \text{adm}(q_1)$$

Proof. Let $\omega \in \text{mCanAns}(q_1 \text{ UNION } q_2, \mathcal{K})$ s.t. $\omega \notin \text{mCanAns}(q_2, \mathcal{K})$.
Then from Definition 10, because $\omega \in \text{mCanAns}(q_1 \text{ UNION } q_2, \mathcal{K})$:

$$\omega \in \bigcup_{q' \in \text{branches}(q_1 \text{ UNION } q_2)} \text{mCanAns}(q_1 \text{ UNION } q_2, \mathcal{K}, q')$$

And from Definition 9:

$$\text{branches}(q_1 \text{ UNION } q_2) = \text{branches}(q_1) \cup \text{branches}(q_2)$$

So:

$$\omega \in \bigcup_{q' \in \text{branches}(q_1) \cup \text{branches}(q_2)} \text{mCanAns}(q_1 \text{ UNION } q_2, \mathcal{K}, q')$$

So there is an SJO query $q' \in \text{branches}(q_1) \cup \text{branches}(q_2)$ s.t. $\omega \in \text{mCanAns}(q_1 \text{ UNION } q_2, \mathcal{K}, q')$
So there is an $\omega' \in \text{sparqlAns}(q_1 \text{ UNION } q_2, \text{can}(\mathcal{K})) \cap \text{sparqlAns}(q', \text{can}(\mathcal{K}))$ s.t.
$\omega = \omega'|_X$ for some $X \in \max_{\subseteq}(\text{adm}(q') \cap 2^{\text{dom}(\omega')})$.
Then we can distinguish three cases:

- $q' \in \text{branches}(q_1) \setminus \text{branches}(q_2)$.
  Since $\omega' \in \text{sparqlAns}(q', \text{can}(\mathcal{K}))$ and $q' \notin \text{branches}(q_2)$, $\omega' \notin \text{sparqlAns}(q_2, \text{can}(\mathcal{K}))$ must hold.
  Then because $\omega' \in \text{sparqlAns}(q_1 \text{ UNION } q_2, \text{can}(\mathcal{K}))$, from Definition 1, $\omega' \in \text{sparqlAns}(q_1)$, must hold.
  So from Definition 8, $\omega \in \text{mCanAns}(q_1, \mathcal{K}, q')$.
  And since $q' \in \text{branches}(q_1)$, from Definition 10, $\omega \in \text{mCanAns}(q_1, \mathcal{K})$.
  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 $\omega' \in \text{sparqlAns}(q', \text{can}(\mathcal{K}))$ and $q' \notin \text{branches}(q_1)$, $\omega' \notin \text{sparqlAns}(q_1, \text{can}(\mathcal{K}))$ must hold.
  Then because $\omega' \in \text{sparqlAns}(q_1 \text{ UNION } q_2, \text{can}(\mathcal{K}))$, from Definition 1, $\omega' \in \text{sparqlAns}(q_2)$, must hold.
  So from Definition 8, $\omega \in \text{mCanAns}(q_2, \mathcal{K}, q')$.
  And since $q' \in \text{branches}(q_2)$, from Definition 10, $\omega \in \text{mCanAns}(q_2, \mathcal{K})$, which would contradict the hypothesis.

- $q' \in \mathsf{branches}(q_1) \cap \mathsf{branches}(q_2)$.
  Since $\omega' \in \mathsf{sparqlAns}(q_1 \text{ UNION } q_2, \mathsf{can}(\mathcal{K}))$, from Definition 1,
  $\omega' \in \mathsf{sparqlAns}(q_1, \mathsf{can}(\mathcal{K}))$ or $\omega' \in \mathsf{sparqlAns}(q_2, \mathsf{can}(\mathcal{K}))$ must hold.
  If $\omega' \in \mathsf{sparqlAns}(q_1, \mathsf{can}(\mathcal{K}))$, then from Definition 8, $\omega \in \mathsf{mCanAns}(q_1, \mathcal{K}, q')$.
  And since $q' \in \mathsf{branches}(q_1)$, from Definition 10, $\omega \in \mathsf{mCanAns}(q_1, \mathcal{K})$.
  So from Lemma 7 below, $\mathsf{dom}(\omega) \in \mathsf{adm}(q_1)$
  If $\omega' \in \mathsf{sparqlAns}(q_2, \mathsf{can}(\mathcal{K}))$ instead, then from Definition 8, $\omega \in \mathsf{mCanAns}(q_2, \mathcal{K}, q')$.
  And since $q' \in \mathsf{branches}(q_2)$, from Definition 10, $\omega \in \mathsf{mCanAns}(q_2, \mathcal{K})$, which would contradict the hypothesis. □

Lemma 7. *For any SUJO query q, $\mathcal{L}_{\mathsf{can}}$ KB $\mathcal{K}$ and $\omega \in \mathsf{mCanAns}(q, \mathcal{K})$:*

$$\mathsf{dom}(\omega) \in \mathsf{adm}(q)$$

Proof. Let q be a SUJO query and $\mathcal{K}$ an $\mathcal{L}_{\mathsf{can}}$ KB.
Then $\mathsf{mCanAns}(q, \mathcal{K}) = \bigcup_{q' \in \mathsf{branches}(q)} (\mathsf{sparqlAns}(q, \mathcal{A}, q') \blacktriangleright \mathsf{aDom}(\mathcal{K})) \otimes \mathsf{adm}(q')$.
So for each $\omega \in \mathsf{mCanAns}(q, \mathcal{K})$, there is a $q' \in \mathsf{branches}(q)$ and solution mapping ω' s.t. $\omega = \omega'|_X$ for some $X \in \max_{\subseteq}(\mathsf{adm}(q') \cap 2^{\mathsf{dom}(\omega')})$.
So $\mathsf{dom}(\omega) \in \mathsf{adm}(q')$.
Then Lemma 8 below shows that for any $q' \in \mathsf{branches}(q)$, $\mathsf{adm}(q') \subseteq \mathsf{adm}(q)$.
So $\mathsf{dom}(\omega) \in \mathsf{adm}(q')$. □

Lemma 8. *For any SUJO query q and SJO $q' \in \mathsf{branches}(q)$:*

$$\mathsf{adm}(q') \subseteq \mathsf{adm}(q)$$

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

- q is a triple pattern.
  Then $\mathsf{branches}(q) = \{q\}$, so the property trivially holds.
- $q = \text{SELECT}_Y\ q_2$.
  From Definition 9, $q' = \text{SELECT}_Y\ q_2'$ for some $q_2' \in \mathsf{branches}(q_2)$.
  So from Definition 4, $X = Y \cap Y'$ for some $Y' \in \mathsf{adm}(q_2')$.
  Then by IH, $Y' \in \mathsf{adm}(q_2)$.
  So $X = Y \cap Y'$ for some $Y' \in \mathsf{adm}(q_2)$.
  And again from Definition 4, $X \in \mathsf{adm}(\text{SELECT}_Y\ q_2) = \mathsf{adm}(q)$.
- $q = q_1$ JOIN q_2.
  From Definition 9, $q' = q_1'$ JOIN q_2' for some $(q_1', q_2') \in \mathsf{branches}(q_1) \times \mathsf{branches}(q_2)$.
  So from Definition 4, $X = X_1 \cup X_2$ for some $(X_1, X_2) \in \mathsf{adm}(q_1') \times \mathsf{adm}(q_2')$.
  Then by IH, $X_1 \in \mathsf{adm}(q_1)$ and $X_2 \in \mathsf{adm}(q_2)$.
  So $X = X_1 \cup X_2$ for some $(X_1, X_2) \in \mathsf{adm}(q_1) \times \mathsf{adm}(q_2)$.
  And again from Definition 4, $X \in \mathsf{adm}(q_1 \text{ JOIN } q_2) = \mathsf{adm}(q)$.
- $q = q_1$ UNION q_2.
  From Definition 9, $q' \in \mathsf{branches}(q_i)$ for some $i \in \{1, 2\}$.
  So from Definition 4, $X \in \mathsf{adm}(q_i)$ for some $i \in \{1, 2\}$.
  Then again from Definition 4, $X \in \mathsf{adm}(q_1 \text{ UNION } q_2) = \mathsf{adm}(q)$.
- If $q = q_1$ OPT q_2, then $q' \in \mathsf{branches}(q_1 \text{ JOIN } q_2)$ or $q' \in \mathsf{branches}(q_1)$ must hold.
  - If $q' \in \mathsf{branches}(q_1 \text{ JOIN } q_2)$, then we showed above that $X \in \mathsf{adm}(q_1 \text{ JOIN } q_2)$ must hold.
    And from Definition 4, $\mathsf{adm}(q_1 \text{ JOIN } q_2) \subseteq \mathsf{adm}(q)$.
    So $X \in \mathsf{adm}(q)$.
  - If $q' \in \mathsf{branches}(q_1)$, then by IH, $X \in \mathsf{adm}(q_1)$.
    And from Definition 4, $\mathsf{adm}(q_1) \subseteq \mathsf{adm}(q)$.
    So $X \in \mathsf{adm}(q)$. □

Appendix C. Proof of Proposition 3

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

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

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

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

So *(i)* certAns(partEval($q, \mathcal{K}$), $\mathcal{K}$) = certAns($q, \mathcal{K}$).
And from Definition 12 still, *(ii)* certAns(partEval($q, \mathcal{K}$), $\mathcal{K}$) = epCertAns($q, \mathcal{K}$).
So from *(i)* and *(ii)*, epCertAns($q, \mathcal{K}$) = certAns($q, \mathcal{K}$). □

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

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

Proof. Let q be a SUJO query, let $\mathcal{A}$ be an ABox, and let $\mathcal{K} = \langle \emptyset, \mathcal{A} \rangle$.
Then $\mathcal{A} = \text{can}(\mathcal{K})$.
Now by induction on q:

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

    $$\omega \in \text{sparqlAns}(\text{SELECT}_X\ \text{partEval}(q', \mathcal{K}), \mathcal{I}) =$$
    $$\text{sparqlAns}(\text{partEval}(q, \mathcal{K}), \mathcal{I}).$$

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

And because $\mathsf{range}(\omega_i) \subseteq \mathsf{aDom}(\mathcal{K})$, $h \circ \omega_i = \omega_i$.
Therefore for any model $\mathcal{I}$ of $\mathcal{K}$, $\omega_i \in \mathsf{sparqlAns}(\mathsf{partEval}(q_i, \mathcal{K}), \mathcal{I})$.
So from Definition 12, $\omega_i \in \mathsf{epCertAns}(q_i, \mathcal{K})$.
Then by IH, $\omega_i \in \mathsf{sparqlAns}(q_i, \mathcal{A})$.
So from Definition 1, $\omega = \omega_1 \cup \omega_2 \in \mathsf{sparqlAns}(q_1 \text{ JOIN } q_2, \mathcal{A}) = \mathsf{sparqlAns}(q, \mathcal{A})$.

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

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

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

∗ If $\omega \in \mathsf{sparqlAns}(q_1 \text{ JOIN } q_2, \mathcal{A})$, as shown for the case $q = q_1 \text{ JOIN } q_2$, $\omega \in \mathsf{epCertAns}(q_1 \text{ JOIN } q_2, \mathcal{K})$ must hold.
So from Definition 12, for every model $\mathcal{I}$ of $\mathcal{K}$,

$$\begin{aligned}
&\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1 \text{ JOIN } q_2, \mathcal{K}), \mathcal{I})\\
&= \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K}), \mathcal{I})\\
&\subseteq \mathsf{sparqlAns}((\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K}))\\
&\qquad \text{UNION } (\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{epCertAns}(q_2, \mathcal{K})), \mathcal{I})\\
&= \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathcal{I}).
\end{aligned}$$

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

∗ If $\omega \in \mathsf{sparqlAns}(q_1 \text{ MINUS } q_2, \mathcal{A})$, then
(i) $\omega \in \mathsf{sparqlAns}(q_1, \mathcal{A})$ and
(ii) for all $\omega' \in \mathsf{sparqlAns}(q_2, \mathcal{A})$, $\omega \nsim \omega'$.
From *(i)*, by IH, $\omega \in \mathsf{epCertAns}(q_1, \mathcal{K})$.
So from Definition 12,
(iii) for any model $\mathcal{I}$ of $\mathcal{K}$, $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}), \mathcal{I})$.
Then again by IH, $\mathsf{sparqlAns}(q_2, \mathcal{A}) = \mathsf{epCertAns}(q_2, \mathcal{K})$.
So from *(ii)*, *(iv)* for all $\omega' \in \mathsf{epCertAns}(q_2, \mathcal{K})$, $\omega \nsim \omega'$.
Therefore from *(iii)*, *(iv)* and Definitions 1 and 12, for any model $\mathcal{I}$ of $\mathcal{K}$,

$$\begin{aligned}
&\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{epCertAns}(q_2, \mathcal{K}), \mathcal{I})\\
&\subseteq \mathsf{sparqlAns}((\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{epCertAns}(q_2, \mathcal{K}))\\
&\qquad \text{UNION } (\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K})), \mathcal{I})\\
&= \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathcal{I}).
\end{aligned}$$

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

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

$$\mathsf{epCertAns}(q_1, \mathcal{K}) \preceq_g \mathsf{epCertAns}(q_1 \text{ OPT } q_2, \mathcal{K})$$

Proof. Let q_1, q_2 be SUJO queries, let $\mathcal{K}$ be a KB, and let $\omega_1 \in \mathsf{epCertAns}(q_1, \mathcal{K})$.
We need to show that there is an $\omega' \in \mathsf{epCertAns}(q_1 \text{ OPT } q_2, \mathcal{K})$ s.t. $\omega_1 \preceq \omega'$.
Since $\omega_1 \in \mathsf{epCertAns}(q_1, \mathcal{K})$, from Definition 12, for every model $\mathcal{I}$ of $\mathcal{K}$, $\omega_1 \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}), \mathcal{I})$.
Then we have two cases:

- there is an $\omega_2 \in \mathsf{epCertAns}(q_2, \mathcal{K})$ s.t. $\omega_1 \sim \omega_2$.
  Then for every model $\mathcal{I}$ of $\mathcal{K}$, $\omega_2 \in \mathsf{sparqlAns}(\mathsf{partEval}(q_2, \mathcal{K}), \mathcal{I})$.
  So from Definitions 1 and 12, for every model $\mathcal{I}$ of $\mathcal{K}$:

$$\begin{aligned}
&\omega_1 \cup \omega_2 \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K}), \mathcal{I})\\
&\qquad \subseteq \mathsf{sparqlAns}((\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K}))\\
&\qquad\qquad \text{UNION } (\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{certAns}(q_2, \mathcal{K})), \mathcal{I})\\
&\qquad = \mathsf{sparqlAns}(\mathsf{partEval}(q_1 \text{ OPT } q_2, \mathcal{K}), \mathcal{I})
\end{aligned}$$

  So from Definition 12, $\omega_1 \cup \omega_2 \in \mathsf{epCertAns}(q_1 \text{ OPT } q_2, \mathcal{K})$.
  And $\omega_1 \preceq \omega_1 \cup \omega_2$.
- there is no $\omega_2 \in \mathsf{epCertAns}(q_2, \mathcal{K})$ s.t. $\omega_2 \sim \omega$.
  Then from Definitions 1 and 12, for every model $\mathcal{I}$ of $\mathcal{K}$:

$$\begin{aligned}
&\omega_1 \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{epCertAns}(q_2, \mathcal{K}), \mathcal{I})\\
&\quad \subseteq \mathsf{sparqlAns}((\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K}))\\
&\qquad\quad \text{UNION } (\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{certAns}(q_2, \mathcal{K})), \mathcal{I})\\
&\quad = \mathsf{sparqlAns}(\mathsf{partEval}(q_1 \text{ OPT } q_2, \mathcal{K}), \mathcal{I})
\end{aligned}$$

  So from Definition 12, $\omega_1 \in \mathsf{epCertAns}(q_1 \text{ OPT } q_2, \mathcal{K})$.
  And $\omega_1 \preceq \omega_1$. □

Lemma 12. *For any SUJO queries q, q', KB $\mathcal{K}$ and solution mapping ω,*

$$\text{if } \omega \in \mathsf{epCertAns}(q \text{ UNION } q', \mathcal{K}) \text{ and } \omega \notin \mathsf{epCertAns}(q', \mathcal{K}), \text{ then } \mathsf{dom}(\omega) \in \mathsf{adm}(q)$$

Proof. Let q, q' be two SUJO queries let $\mathcal{K}$ be a KB,
and let $\omega \in \mathsf{epCertAns}(q \text{ UNION } q', \mathcal{K})$ s.t. $\omega \notin \mathsf{epCertAns}(q', \mathcal{K})$.
From Definition 12,
$\mathsf{epCertAns}(q \text{ UNION } q', \mathcal{K}) = \mathsf{certAns}(\mathsf{partEval}(q, \mathcal{K}) \text{ UNION } \mathsf{partEval}(q', \mathcal{K}), \mathcal{K})$
and $\mathsf{epCertAns}(q', \mathcal{K}) = \mathsf{certAns}(\mathsf{partEval}(q', \mathcal{K}), \mathcal{K})$.
So there is a model $\mathcal{I}$ of $\mathcal{K}$ s.t.
$\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}) \text{ UNION } \mathsf{partEval}(q', \mathcal{K}), \mathcal{I})$ and
$\omega \notin \mathsf{sparqlAns}(\mathsf{partEval}(q', \mathcal{K}), \mathcal{I})$.
Therefore from Definition 1, $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathcal{I})$ must hold.
Then from Lemma 13 below, it follows that $\mathsf{dom}(\omega) \in \mathsf{adm}(q)$. □

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

$$\mathsf{dom}(\omega) \in \mathsf{adm}(q)$$

Proof. Let q be a SUJO query, let $\mathcal{I}$ be a graph, and let $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathcal{I})$.
By induction on q:

- q is a triple pattern.
  Then $\mathsf{partEval}(q, \mathcal{K}) = q$.
  And from Definitions 1 and 4,
  $\mathsf{dom}(\omega) = \mathsf{vars}(\mathsf{partEval}(q, \mathcal{K})) = \mathsf{vars}(q) \in \mathsf{adm}(q)$.
- $q = \text{SELECT}_X\ q'$.
  Then $\mathsf{partEval}(q, \mathcal{K}) = \text{SELECT}_X\ \mathsf{partEval}(q', \mathcal{K})$.
  So from Definition 1, there is a $\omega' \in \mathsf{sparqlAns}(\mathsf{partEval}(q', \mathcal{K}), \mathcal{I})$
  s.t. $\omega = \omega'|_X$.
  And by IH, $\mathsf{dom}(\omega') \in \mathsf{adm}(q')$.
  So from Definition 4, $\mathsf{dom}(\omega') \cap X = \mathsf{dom}(\omega) \in \mathsf{adm}(q)$.
- $q = q_1 \text{ JOIN } q_2$.
  Then $\mathsf{partEval}(q, \mathcal{K}) = \mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K})$.
  So from Definition 1, $\omega = \omega_1 \cup \omega_2$ for some $(\omega_1, \omega_2) \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}), \mathcal{I}) \times \mathsf{sparqlAns}(\mathsf{partEval}(q_2, \mathcal{K}), \mathcal{I})$.
  And by IH, for $i \in \{1, 2\}$, $\mathsf{dom}(\omega_i) \in \mathsf{adm}(q_i)$.
  So from Definition 4, $\mathsf{dom}(\omega) = \mathsf{dom}(\omega_1) \cup \mathsf{dom}(\omega_2) \in \mathsf{adm}(q)$.
- $q = q_1 \text{ UNION } q_2$.
  Then $\mathsf{partEval}(q, \mathcal{K}) = \mathsf{partEval}(q_1, \mathcal{K}) \text{ UNION } \mathsf{partEval}(q_2, \mathcal{K})$.
  So from Definition 1, $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}), \mathcal{I})$ or
  $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_2, \mathcal{K}), \mathcal{I})$ must hold.
  So by IH, for some $i \in \{1, 2\}$, $\mathsf{dom}(\omega) \in \mathsf{adm}(q_i)$.
  And from Definition 4, $\mathsf{adm}(q_i) \subseteq \mathsf{adm}(q)$.
  So $\mathsf{dom}(\omega) \in \mathsf{adm}(q)$.
- $q = q_1 \text{ OPT } q_2$.
  Then $\mathsf{partEval}(q, \mathcal{K}) = (\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K})) \text{ UNION }$
  $(\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{epCertAns}(q_2, \mathcal{K}))$.
  So from Definition 1,
  $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K}), \mathcal{I})$ or
  $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{epCertAns}(q_2, \mathcal{K}), \mathcal{I})$ must hold.
  - If $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}) \text{ JOIN } \mathsf{partEval}(q_2, \mathcal{K}), \mathcal{I})$,
    from the proof above for the case $q = q_1 \text{ JOIN } q_2$,
    $\mathsf{dom}(\omega) \in \mathsf{adm}(q_1 \text{ JOIN } q_2)$ must hold.
    And from Definition 4, $\mathsf{adm}(q_1 \text{ JOIN } q_2) \subseteq \mathsf{adm}(q)$.
    So $\mathsf{dom}(\omega) \in \mathsf{adm}(q)$.
  - If $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}) \text{ MINUS } \mathsf{epCertAns}(q_2, \mathcal{K}), \mathcal{I})$,
    then $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}))$.
    So by IH, $\mathsf{dom}(\omega) \in \mathsf{adm}(q_1)$.
    And from Definition 4, $\mathsf{adm}(q_1) \subseteq \mathsf{adm}(q)$.
    So $\mathsf{dom}(\omega) \in \mathsf{adm}(q)$. □

Appendix D. Proof of Proposition 4

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

$$\mathsf{epCertAns}(q, \mathcal{K}) = \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathsf{can}(\mathcal{K})) \triangleright \mathsf{aDom}(\mathcal{K})$$

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

- ($\Rightarrow$).
  Let $\omega \in \mathsf{epCertAns}(q, \mathcal{K})$.
  Then from Definition 12, $\omega \in \mathsf{certAns}(\mathsf{partEval}(q, \mathcal{K}), \mathcal{K})$.
  So $\mathsf{range}(\omega) \subseteq \mathsf{aDom}(\mathcal{K})$ and $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathsf{can}(\mathcal{K}))$ both hold.
- ($\Leftarrow$).
  By induction on q.
  For the cases:
  – q is a triple pattern,
  – $q = \text{SELECT}_X\ q'$,
  – $q = q_1$ JOIN q_2,
  – $q = q_1$ UNION q_2,
  the proof is identical to the one of Lemma 10 (left direction).
  For the case $q = q_1$ OPT q_2, let
  $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q, \mathcal{K}), \mathsf{can}(\mathcal{K})) \triangleright \mathsf{aDom}(\mathcal{K})$.
  Then either:
  – $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K})$ JOIN $\mathsf{partEval}(q_2, \mathcal{K}), \mathsf{can}(\mathcal{K}))$.
    Then the proof is identical to the case $q = q_1$ JOIN q_2.
  – *(i)* $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}), \mathsf{can}(\mathcal{K}))$, and
    (ii) for all $\omega' \in \mathsf{epCertAns}(q_2, \mathcal{K})$, $\omega \nsim \omega'$.
    From *(i)*, by IH, $\omega \in \mathsf{epCertAns}(q_1, \mathcal{K})$.
    So from Definition 12, *(iii)* for any model $\mathcal{I}$ of $\mathcal{K}$,
    $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K}), \mathcal{I})$.
    So from *(ii)* and *(iii)*, for any model $\mathcal{I}$ of $\mathcal{K}$,
    $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_1, \mathcal{K})$ MINUS $\mathsf{epCertAns}(q_2, \mathcal{K}), \mathcal{I})$.
    Therefore from Definition 12, $\omega \in \mathsf{epCertAns}(q, \mathcal{K})$. □

Appendix E. Proof of Proposition 6

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

$$\mathsf{mCanAns}(q, \mathcal{K}) = \mathsf{epCertAns}(q, \mathcal{K})$$

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

Appendix F. Complexity of maximal admissible certain answers

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

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

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

- If q is a triple pattern, then $\mathsf{base}(q) = \{\mathsf{vars}(q)\}$.
- If $q = q_1$ JOIN q_2, then $\mathsf{base}(q) = \{B_1 \cup B_2 \mid B_1 \in \min_{\subseteq}(\mathsf{base}(q_1)), B_2 \in \mathsf{base}(q_2)\} \cup$
  $\{B_1 \cup B_2 \mid B_1 \in \mathsf{base}(q_1), B_2 \in \min_{\subseteq}(\mathsf{base}(q_2))\}$.
- If $q = q_1$ OPT q_2, then $\mathsf{base}(q) = \mathsf{base}(q_1) \cup \mathsf{base}(q_1$ 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 $\mathsf{base}(q)$ w.r.t. set-inclusion is guaranteed to be unique. This is shown with Lemma 14 below.
- $\mathsf{adm}(q) = \{\bigcup \mathcal{B} \mid \mathcal{B} \in 2^{\mathsf{base}(q)}\}$. This is shown with Lemma 15 below.
- $|\mathsf{base}(q)| = O(|q|)$. This is shown with Lemma 16 below. □

Lemma 14. *For any JO query q, $|\min_{\subseteq}(\mathsf{base}(q))| = 1$.*

Proof. By induction on the structure of q.

- If q is a triple pattern, then $|\mathsf{base}(q)| = 1$, so $|\min_{\subseteq}(\mathsf{base}(q))| = 1$.
- If $q = q_1$ JOIN q_2, let $\mathcal{B}_1 = \{B_1 \cup B_2 \mid B_1 \in \min_{\subseteq}(\mathsf{base}(q_1)), B_2 \in \mathsf{base}(q_2)\}$, and $\mathcal{B}_2 = \{B_1 \cup B_2 \mid B_1 \in \mathsf{base}(q_1), B_2 \in \min_{\subseteq}(\mathsf{base}(q_2))\}$.
  By IH, for $i \in \{1,2\}$, $|\min_{\subseteq}(\mathsf{base}(q_i))| = \{M_i\}$ for some $M_i \subseteq \mathsf{vars}(q_i)$.
  Then from the definition of $\mathcal{B}_1$, $M_1 \cup M_2 \in \mathcal{B}_1$.
  And for each $B_2 \in \mathsf{base}(q_2)$, $M_2 \subseteq B_2$.
  So for each $M_1 \cup B_2 \in \mathcal{B}_1$, $M_1 \cup M_2 \subseteq M_1 \cup B_2$.
  So $\min_{\subseteq}(\mathcal{B}_1) = \{M_1 \cup M_2\}$.
  And similarly, $\min_{\subseteq}(\mathcal{B}_2) = \{M_1 \cup M_2\}$.
  Then because $\mathsf{base}(q) = \mathcal{B}_1 \cup \mathcal{B}_2$, $\min_{\subseteq}(\mathsf{base}(q)) = \{M_1 \cup M_2\}$.
- If $q = q_1$ OPT q_2, by IH, $\min_{\subseteq}(\mathsf{base}(q_1)) = \{M\}$ for some $M \subseteq \mathsf{vars}(q_1)$.
  So $M \subseteq B$ for each $B \in \mathsf{base}(q_1)$.
  And we showed above that $M \subseteq B$ for each $B \in \mathsf{base}(q_1$ JOIN $q_2)$.
  Then from Definition 14, $\mathsf{base}(q) = \mathsf{base}(q_1) \cup \mathsf{base}(q_1$ JOIN $q_2)$.
  So $M \in \mathsf{base}(q_1) \subseteq \mathsf{base}(q)$, and $M \subseteq B$ for each $B \in \mathsf{base}(q_1) \cup \mathsf{base}(q_1$ JOIN $q_2) = \mathsf{base}(q)$.
  Therefore $\min_{\subseteq}(\mathsf{base}(q)) = \{M\}$. □

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

Proof. By induction on the structure of q.

- If q is a triple pattern, then $\mathsf{base}(q) = \mathsf{adm}(q) = \{\mathsf{vars}(q)\}$.
- If $q = q_1$ JOIN q_2:
  - ($\Rightarrow$).
    Let $X \in \mathsf{adm}(q)$.
    From Definition 4, $X = X_1 \cup X_2$ for some $(X_1, X_2) \in \mathsf{adm}(q_1) \times \mathsf{adm}(q_2)$.
    And by IH, for $i \in \{1,2\}$, $X_i = \bigcup \mathcal{B}_i$ for some $\mathcal{B}_i \in 2^{\mathsf{base}(q_1)}$.
    Then from Lemma 14, $|\min_{\subseteq}(\mathsf{base}(q_i))| = \{M_i\}$ for some $M_i \subseteq \mathsf{vars}(q_i)$.
    So for each $B_i \in \mathcal{B}_i$, $M_i \subseteq B_i$.
    Therefore $\bigcup \mathcal{B}_i = \{M_i\} \cup \bigcup \mathcal{B}_i$.
    And since $X = X_1 \cup X_2$, we have:
    $$X = \bigcup \mathcal{B}_1 \cup \bigcup \mathcal{B}_2$$
    $$X = \{M_1\} \cup \bigcup \mathcal{B}_1 \cup \{M_2\} \cup \bigcup \mathcal{B}_2$$
    $$X = \{M_2 \cup B_1 \mid B_1 \in \mathcal{B}_1\} \cup \{M_1 \cup B_2 \mid B_2 \in \mathcal{B}_2\}$$
    Then from Definition 13, for each $B_1 \in \mathcal{B}_1$, $M_2 \cup B_1 \in \mathsf{base}(q)$.
    Similarly, for each $B_2 \in \mathcal{B}_2$, $M_1 \cup B_2 \in \mathsf{base}(q)$.
    So $X = \bigcup \mathcal{B}$ for some $\mathcal{B} \in 2^{\mathsf{base}(q)}$.
  - ($\Leftarrow$).
    Let $X = \bigcup \mathcal{B}$ for some $\mathcal{B} \in 2^{\mathsf{base}(q)}$.
    From Definition 13, for each $B \in \mathcal{B}$, there are $(B_1, B_2) \in \mathsf{base}(q_1) \times \mathsf{base}(q_2)$ s.t. $B = B_1 \cup B_2$.
    For $i \in \{1,2\}$, let $\mathcal{B}_i = \{B_i \mid B_i \cup B' \in \mathcal{B}, B_i \in \mathsf{base}(q_i)\}$.
    Then for $i \in \{1,2\}$, $\mathcal{B}_i \neq \emptyset$.
    And $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$.
    So $X = \bigcup \mathcal{B} = \bigcup \mathcal{B}_1 \cup \bigcup \mathcal{B}_2$.
    And by IH, for $i \in \{1,2\}$, $\bigcup \mathcal{B}_i \in \mathsf{adm}(q_i)$.
    Therefore $X = X_1 \cup X_2$ for some $(X_1, X_2) \in \mathsf{adm}(q_1) \times \mathsf{adm}(q_2)$.
    So From Definition 4, $X \in \mathsf{adm}(q)$.
- If $q = q_1$ OPT q_2:
  - ($\Rightarrow$).
    Let $X \in \mathsf{adm}(q)$.
    From Definition 4, $X \in \mathsf{adm}(q_1)$ or $X \in \mathsf{adm}(q_1$ JOIN $q_2)$ must hold.
    If $X \in \mathsf{adm}(q_1)$, then by IH, $X = \bigcup \mathcal{B}$ for some $\mathcal{B} \in 2^{\mathsf{base}(q_1)}$.

And from Definition 13, $\mathsf{base}(q_1) \subseteq \mathsf{base}(q)$.
If $X \in \mathsf{adm}(q_1 \text{ JOIN } q_2)$, then we showed above that $X = \bigcup \mathcal{B}$ for some $\mathcal{B} \in 2^{\mathsf{base}(q_1 \text{ JOIN } q_2)}$.
And from Definition 13, $\mathsf{base}(q_1 \text{ JOIN } q_2) \subseteq \mathsf{base}(q)$.
So in both cases, $X = \bigcup \mathcal{B}$ for some $\mathcal{B} \in 2^{\mathsf{base}(q)}$.

- ($\Leftarrow$).
Let $X = \bigcup \mathcal{B}$ for some $\mathcal{B} \in 2^{\mathsf{base}(q)}$.
From Definition 13, for each $B \in \mathcal{B}$, $B \in \mathsf{base}(q_1)$ or there are $(B_1, B_2) \in \mathsf{base}(q_1) \times \mathsf{base}(q_2)$ s.t. $B = B_1 \cup B_2$.
For $i \in \{1, 2\}$, let $\mathcal{B}_i = \{B_i \mid B_i \cup B' \in \mathcal{B}, B_i \in \mathsf{base}(q_i)\}$.
Then $\mathcal{B}_1 \neq \emptyset$.
And $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$.
If $\mathcal{B}_2 = \emptyset$, then $X = \bigcup \mathcal{B} = \bigcup \mathcal{B}_1$.
And by IH, $\bigcup \mathcal{B}_1 \in \mathsf{adm}(q_1)$.
So From Definition 4, $X \in \mathsf{adm}(q)$.
If $\mathcal{B}_2 \neq \emptyset$, then $X = \bigcup \mathcal{B} = \bigcup \mathcal{B}_1 \cup \bigcup \mathcal{B}_2$.
And by IH, for $i \in \{1, 2\}$, $\bigcup \mathcal{B}_i \in \mathsf{adm}(q_i)$.
Therefore $X = X_1 \cup X_2$ for some $(X_1, X_2) \in \mathsf{adm}(q_1) \times \mathsf{adm}(q_2)$.
So From Definition 4, $X \in \mathsf{adm}(q)$. □

Lemma 16. *For any JO query q, $|\mathsf{base}(q)| = O(|q|)$*

Proof. By induction on the structure of q.

- If q is a triple pattern, then $|\mathsf{base}(q)| = 1$.
- If $q = q_1$ JOIN q_2, then immediately from the definition of $\mathsf{base}(q)$,
$|\mathsf{base}(q)| = O(|\min_{\subseteq}(\mathsf{base}(q_1))|) \cdot |\mathsf{base}(q_2)| + |\min_{\subseteq}(\mathsf{base}(q_2))|) \cdot |\mathsf{base}(q_1)|)$.
So from Lemma 14, $|\mathsf{base}(q)| = O(|\mathsf{base}(q_2)| + |\mathsf{base}(q_1)|)$.
And by IH, $|\mathsf{base}(q_i)| = O(|q_i|)$ for $i \in \{1, 2\}$.
So $|\mathsf{base}(q)| = O(|q_1|) + O(|q_2|) = O(|q|)$.
- If $q = q_1$ OPT q_2, the argument is similar to the case $q = q_1$ JOIN q_2. □

Appendix G. Complexity of epistemic certain answers

Proposition 8. $\text{EVAL}_{\mathsf{epCertAns}}$ *is in* PSPACE *for SUJO queries*

Proof. Let q_0 be a SUJO query, let $\mathcal{K}$ be an $\mathcal{L}_{\mathsf{can}}$ KB, and let ω be a solution mapping.
Algorithm 1 below describes a procedure that decides
$\omega \in \mathsf{sparqlAnspartEval}(q_0, \mathcal{K}), \mathsf{can}(\mathcal{K})$.
From Proposition 4, in order to decide $\omega \in \mathsf{epCertAns}(q_0, \mathcal{K})$, it is sufficient to call this procedure and decide $\mathsf{range}(\omega) \subseteq \mathsf{aDom}(\mathcal{K})$.

In order to explain the procedure, we recall some well-known properties of canonical models for *DL-Lite*$_{\mathcal{R}}$ KBs. Specifically, $\mathsf{can}(\mathcal{K})$ below refers to the canonical model of a *DL-Lite*$_{\mathcal{R}}$ KB $\mathcal{K}$ defined in [5].

(I) Let $\mathsf{roles}(\mathcal{T})$ be the set of binary predicates (i.e. elements of $\mathsf{N_R}$) that appears in $\mathcal{T}$ and their inverse.
Then each element of the domain $\Delta^{\mathsf{can}(\mathcal{K})}$ of $\mathsf{can}(\mathcal{K})$ can be identified by a word of the form $w = aR_1..R_n$, with $n \geq 0$, where $a \in \mathsf{aDom}(\mathcal{K})$ and $R_i \in \mathsf{roles}(\mathcal{T})$ for $i \in [1..n]$.
If w is a word over $\mathsf{aDom}(\mathcal{K}) \cup \mathsf{roles}(\mathcal{T})$, we will use $l(w)$ to denote its length.
Note that w may not identify any element of $\Delta^{\mathsf{can}(\mathcal{K})}$.
If it does, we will use $w \in \Delta^{\mathsf{can}(\mathcal{K})}$ as a shortcut.

(II) Let $w = aR_1..R_n$ be a word over $\mathsf{aDom}(\mathcal{K}) \cup \mathsf{roles}(\mathcal{T})$.
Then $w \in \Delta^{\mathsf{can}(\mathcal{K})}$ can be decided as follows:
- if $n = 0$, then decide $a \in \mathsf{aDom}(\mathcal{K})$,
- otherwise iterate (at most $O(|\mathcal{T}|)$ times) over $\mathcal{T}$ to decide $\mathcal{T} \models \exists R_{n-1}^- \sqsubseteq \exists R_n$, then again to decide $\mathcal{T} \models \exists R_{n-2}^- \sqsubseteq \exists R_{n-1}$, etc., and finally decide $\mathcal{K} \models \exists R_1(a)$.

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

(III) If $A(w)$ is a triple with $A \in \mathsf{N_C}$ and $w = aR_1..R_n \in \Delta^{\mathsf{can}(\mathcal{K})}$,
then $A(w) \in \mathsf{can}(\mathcal{K})$ can be decided as follows:
- if $n = 0$, then decide $\mathcal{K} \models A(a)$,
- otherwise decide $\mathcal{T} \models \exists R_n^- \sqsubseteq A$.

Both can be decided by iterating over $\mathcal{T}$ and/or $\mathcal{A}$, using space polynomial in $|\mathcal{K}|$.
And similarly for the case $R(w1, w2) \in \mathsf{can}(\mathcal{K})$, where $R \in \mathsf{N_R}$ (with an additional verification: one of $w1, w2$ 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 $\{\} \in \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}))$ iff there is a match ω for q in $\mathsf{can}(\mathcal{K})$ such that, for each $w \in \mathsf{range}(\omega)$, $l(w) \leq \max(|\mathcal{T}|, m)$.

Algorithm 1 simulates a top-down evaluation of an input boolean query q_0 over $\mathsf{can}(\mathcal{K})$, where q_0 contains m variables.
Correctness follows immediately from Definitions 1 and 12, and Point (IV) above.

Algorithm 1 DECIDE $\omega \in \mathsf{sparqlAns}(\mathsf{partEval}(q_0, \mathcal{K}), \mathsf{can}(\mathcal{K}))$.

Input: boolean SUJO query q_0 with m variables, *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

```
1:
2: return isMatch(ω, q0, K)
3:
4: function isMatch(ω, q, K)
5:
6:     if q = t
7:         return ω(t) ∈ can(K)
8:
9:     if q = SELECT_X q'
10:        V = vars(q') \ X
11:        for all ω' : V → {w ∈ Δ^can(K) | l(w) ≤ max(|T|, m)}
12:            if isMatch(ω ∪ ω', q', K)
13:                return true
14:        end for
15:        return false
16:
17:    if q = q1 UNION q2
18:        if isMatch(ω, q1, K)
19:            return true
20:        return isMatch(ω, q2, K)
21:
22:    if q = q1 JOIN q2
23:        for all (V1, V2) | V1 ⊆ vars(q1), V2 ⊆ vars(q2)
24:            if isMatch(ω|V1, q1, K) and isMatch(ω|V2, q2, K)
25:                return true
26:        end for
27:        return false
28:
29:    if q = q1 OPT q2
30:        if isMatch(ω, q1 JOIN q2, K)
31:            return true
32:        if isMatch(ω, q1, K)
33:            for all ω' : vars(q2) → aDom(K)
34:                if ω ∼ ω' and isMatch(ω', q, K)
35:                    return false
36:            end for
37:            return true
38:        return false
39: end function
```

To see why this procedure can be executed using space polynomial in $|\mathcal{K}| + |q_0|$, observe first that each call to the function isMatch provides a solution mapping ω such that each $w \in \mathsf{range}(\omega)$ verifies $w \in \Delta^{\mathsf{can}(\mathcal{K})}$ and $l(w) \leq \max(|\mathcal{T}|, m)$.
Then:

- In the case $q = t$ (line 7), because $\mathsf{range}(\omega) \subseteq \Delta^{\mathsf{can}(\mathcal{K})}$, from Point (III) above, $\omega(t) \in \mathsf{can}(\mathcal{K})$ 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 $\Delta^{\mathsf{can}(\mathcal{K})}$ with length $\leq \max(|\mathcal{T}|, m)$ are enumerated. This can be achieved by enumerating all words w over $\mathsf{aDom}(\mathcal{K}) \cup \mathsf{roles}(\mathcal{T})$ of length $\leq \max(|\mathcal{T}|, m)$, and, for each of these, decide $w \in \Delta^{\mathsf{can}(\mathcal{K})}$ using polynomial space, as explained in Point (II) above.
- For the case $q = q_1$ JOIN q_2, from Definition 1, $\omega \in \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K}))$ iff there are two sets V_1, V_2 of variables s.t., for $i \in \{1, 2\}$, $V_i \subseteq \mathsf{vars}(q_i)$ and $\omega_i|_{V_i} \in \mathsf{sparqlAns}(q_i, \mathsf{can}(\mathcal{K}))$.
  Moreover, the cartesian product $2^{\mathsf{vars}(q_1)} \times 2^{\mathsf{vars}(q_2)}$ can be enumerated (line 23) using space polynomial in $|\mathsf{vars}(q)|$. □

References

[1] S. Ahmetaj, W. Fischl, R. Pichler, M. Šimkus, S. Skritek, Towards reconciling SPARQL and certain answers, in: Proceedings of the 24th International Conference on World Wide Web, ACM, 2015, pp. 23–33.
[2] M. Arenas, J. Pérez, Querying semantic web data with SPARQL, in: Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, 2011, pp. 305–316.
[3] A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev, The DL-Lite family and relations, J. Artif. Intell. Res. 36 (2009) 1–69.
[4] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.
[5] M. Bienvenu, M. Ortiz, M. Simkus, G. Xiao, Tractable queries for lightweight description logics, in: IJCAI, 2013.
[6] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: answering SPARQL queries over relational databases, Semant. Web J. 8 (3) (2017) 471–487.
[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: the DL-Lite family, J. Automat. Reason. 39 (3) (2007) 385–429.
[8] J. Corman, G. Xiao, Certain answers to a SPARQL query over a knowledge base, in: The 9th Joint International Semantic Technology Conference (JIST), 2019.
[9] S. Costantini, About epistemic negation and world views in epistemic logic programs, Theory Pract. Log. Program. 19 (5–6) (2019) 790–807.
[10] T. Eiter, M. Ortiz, M. Simkus, T.K. Tran, G. Xiao, Query rewriting for Horn-SHIQ plus rules, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012), AAAI, Toronto, Ontario, Canada, July 22–26, 2012, AAAI Press, 2012.
[11] M. Gelfond, Strong introspection, in: AAAI, AAAI Press/The MIT Press, 1991, pp. 386–391.
[12] B. Glimm, C. Ogbuji, SPARQL 1.1 entailment regimes, W3C recommendation, W3C, March 2013.
[13] C. Gutierrez, D. Hernández, A. Hogan, A. Polleres, Certain answers for SPARQL?, in: AMW, 2016.
[14] S. Harris, A. Seaborne, E. Prud'hommeaux, SPARQL 1.1 query language, W3C recommendation, W3C, 2013.

[15] D. Hernández, C. Gutierrez, A. Hogan, Certain answers for SPARQL with blank nodes, in: International Semantic Web Conference, Springer, 2018, pp. 337–353.
[16] R. Kontchakov, M. Rezk, M. Rodríguez-Muro, G. Xiao, M. Zakharyaschev, Answering SPARQL queries over databases under OWL 2 QL entailment regime, in: International Semantic Web Conference, Springer, 2014, pp. 552–567.
[17] E.V. Kostylev, B.C. Grau, On the semantics of SPARQL queries with optional matching under entailment regimes, in: International Semantic Web Conference, Springer, 2014, pp. 374–389.
[18] D. Lanti, G. Xiao, D. Calvanese, VIG: data scaling for OBDA benchmarks, Semant. Web 10 (2) (2019) 413–433.
[19] A. Letelier, J. Pérez, R. Pichler, S. Skritek, Static analysis and optimization of semantic web queries, ACM Trans. Database Syst. 38 (4) (2013) 25.
[20] V. Lifschitz, Nonmonotonic databases and epistemic queries, in: IJCAI, vol. 91, 1991, pp. 381–386.
[21] S. Mengel, S. Skritek, On tractable query evaluation for SPARQL, arXiv preprint arXiv:1712.08939, 2017.
[22] B. Motik, R. Rosati, Reconciling description logics and rules, J. ACM 57 (5) (2010).
[23] A. Polleres, J.P. Wallner, On the relation between SPARQL 1.1 and answer set programming, J. Appl. Non-Class. Log. 23 (1–2) (2013) 159–212.
[24] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of SPARQL, ACM Trans. Database Syst. 34 (3) (2009) 16.
[25] M. Schmidt, M. Meier, G. Lausen, Foundations of SPARQL query optimization, in: Proceedings of the 13th International Conference on Database Theory, ACM, 2010, pp. 4–33.
[26] Y. Shen, T. Eiter, Evaluating epistemic negation in answer set programming, Artif. Intell. 237 (2016) 115–135.
[27] X. Wang, S. Wang, Y. Xin, Y. Yang, J. Li, X. Wang, Distributed Pregel-based provenance-aware regular path query processing on RDF knowledge graphs, World Wide Web 23 (3) (2020) 1465–1496.
[28] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: a survey, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-18, International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 5511–5519.
[29] G. Xiao, L. Ding, B. Cogrel, D. Calvanese, Virtual knowledge graphs: an overview of systems and use cases, Data Intell. 1 (2019) 201–223.
[30] G. Xiao, D. Lanti, R. Kontchakov, S. Komla-Ebri, E. Güzel-Kalayci, L. Ding, J. Corman, B. Cogrel, D. Calvanese, E. Botoeva, The virtual knowledge graph system ontop, in: ISWC, 2020.
[31] Q. Xu, X. Wang, J. Li, Q. Zhang, L. Chai, Distributed subgraph matching on big knowledge graphs using Pregel, IEEE Access 7 (2019) 116453–116464.
[32] X. Zhang, J.V. den Bussche, F. Picalausa, On the satisfiability problem for SPARQL patterns, J. Artif. Intell. Res. 56 (2016) 403–428.