Certain Answers to a SPARQL Query over a Knowledge Base

Julien Corman and Guohui Xiao(✉)

Free University of Bozen-Bolzano, Bolzano, Italy
xiao@inf.unibz.it

Abstract. Ontology-Mediated Query Answering (OMQA) is a well-established framework to answer queries over an RDFS or OWL Knowledge Base (KB). 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 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 aiming at conciliating certain answers and SPARQL answers, and define three additional requirements, which generalize to KBs some basic properties of SPARQL answers. Then we show that a semantics can be defined that satisfies all requirements for SPARQL queries with `SELECT`, `UNION`, and `OPTIONAL`, and for DLs with the canonical model property. We also investigate combined complexity for query answering under such a semantics over *DL-Lite*$_{\mathcal{R}}$ KBs. In particular, we show for different fragments of SPARQL that known upper-bounds for query answering over a plain graph are matched.

1 Introduction

SPARQL is an expressive SQL-like query language designed for Semantic Web data, exposed as RDF graphs. Recently, SPARQL has been extended with so-called *entailment regimes*, which specify different semantics to query an RDFS or OWL *Knowledge Base* (KB), i.e. data enriched with a background theory. This allows retrieving answers to a query not only over the facts explicitly stated in the KB, but more generally over what can be inferred from the KB.

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). However, OMQA was initially developed for *unions of conjunctive queries* (UCQs), which have a limited expressivity when compared to SPARQL. It turns out that 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.

As an illustration, Example 1 provides a simple KB and SPARQL query. The dataset (a.k.a *ABox*) $\mathcal{A}$ states that `Alice` is a driver, whereas the background theory

© Springer Nature Switzerland AG 2020
X. Wang et al. (Eds.): JIST 2019, LNCS 12032, pp. 320–335, 2020.
https://doi.org/10.1007/978-3-030-41407-8_21

(a.k.a. *TBox*) $\mathcal{T}$ states that a driver must have a license (for conciseness, we use DLs for the TBox, rather than some concrete syntax of OWL). Finally, the SPARQL query q retrieves all individuals that have a license.

Example 1

$$\begin{aligned}\mathcal{A} &= \{\texttt{Driver(Alice)}\}\\ \mathcal{T} &= \{\texttt{Driver} \sqsubseteq \exists\texttt{hasLicense}\}\\ q &= \texttt{SELECT ?x WHERE \{ ?x hasLicense ?y \}}\end{aligned}$$

Intuitively, one expects `Alice` to be retrieved as an answer to q. And it would indeed be the case under certain answer semantics, if one considers the natural translation of this query into a UCQ. On the other hand, under the standard semantics of SPARQL 1.1 [8], this query has no answer. This is expected, since the fact that `Alice` has a driving license is not present in the ABox. More surprisingly though, under all SPARQL entailment regimes [6], 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 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. Unfortunately, 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 [8].

We propose in this article to investigate whether and how this dilemma can be solved, for the so-called *set semantics* of SPARQL and certain answers. To this end, we first formulate in Sect. 4 some *requirements* to be met by any reasonable semantics meant to conciliate certain answers and standard SPARQL answers. Then in Sect. 5, we use these requirements to review different semantics. We also show that all requirements can be satisfied, for the fragment of SPARQL with `SELECT`, `UNION` and `OPTIONAL`, and for KBs that admit a unique *canonical model*. Finally, in Sect. 6, we provide combined complexity results for query answering under this semantics, over KBs in *DL-Lite*$_\mathcal{R}$, one of the most popular DLs tailored for query answering, which correspond to the OWL 2 QL standard. We show in particular that upper bounds for this problem match results already known to hold for SPARQL over plain graphs, 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. Before this, Sect. 2 introduces preliminary notions, and Sect. 3 reviews existing semantics for SPARQL over a KB. Proofs can be found in the extended version of this paper (https://arxiv.org/abs/1911.02668).

2 Preliminaries

We assume countably 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 a countably 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 graph $\mathcal{A}$ is a set of triples. For the concrete syntax of SPARQL, we refer to the specification [8]. 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 $A(t)$ rather than $(t, \texttt{rdf:type}, A)$ and for $r \in \mathsf{N_R}$, we use $r(t_1, t_2)$ rather than (t_1, r, t_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 [14] 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 $X \subseteq V$, then:

$$\begin{aligned} \Omega_1 \bowtie \Omega_2 &= \{\omega_1 \cup \omega_2 \mid (\omega_1, \omega_2) \in \Omega_1 \times \Omega_2 \text{ and } \omega_1 \sim \omega_2\} \\ \Omega_1 \setminus \Omega_2 &= \{\omega_1 \mid \omega_1 \in \Omega_1 \text{ and } \omega_1 \not\sim \omega_2 \text{ for all } \omega_2 \in \Omega_2\} \\ \pi_X \Omega &= \{\omega|_X \mid \omega \in \Omega\} \end{aligned}$$

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 [14] for the SUJO fragment (and for set semantics).

Definition 1 (SPARQL answers over a plain graph [14])

If q is a triple pattern, then $\mathsf{sparqlAns}(q, \mathcal{A}) = \{\omega \mid \mathsf{dom}(\omega) = \mathsf{vars}(q)$ *and* $\omega(q) \in \mathcal{A}\}$

$$\begin{aligned} \mathsf{sparqlAns}(q_1 \text{ UNION } q_2, \mathcal{A}) &= \mathsf{sparqlAns}(q_1, \mathcal{A}) \cup \mathsf{sparqlAns}(q_2, \mathcal{A}) \\ \mathsf{sparqlAns}(q_1 \text{ JOIN } q_2, \mathcal{A}) &= \mathsf{sparqlAns}(q_1, \mathcal{A}) \bowtie \mathsf{sparqlAns}(q_2, \mathcal{A}) \\ \mathsf{sparqlAns}(q_1 \text{ OPT } q_2, \mathcal{A}) &= (\mathsf{sparqlAns}(q_1, \mathcal{A}) \bowtie \mathsf{sparqlAns}(q_2, \mathcal{A})) \cup \\ &\quad\ (\mathsf{sparqlAns}(q_1, \mathcal{A}) \setminus \mathsf{sparqlAns}(q_2, \mathcal{A})) \\ \mathsf{sparqlAns}(\text{SELECT}_X\ q, \mathcal{A}) &= \pi_X \mathsf{sparqlAns}(q, \mathcal{A}) \end{aligned}$$

2.2 Description Logic KB, UCQs and Certain Answers

As is conventional in the Description Logics (DL) literature, we represent a KB $\mathcal{K}$ as a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A}$ is called the *ABox* of $\mathcal{K}$, which contains assertions about individuals, and $\mathcal{T}$ is called the *TBox* of $\mathcal{K}$, which contains more abstract knowledge. An ABox is a finite set of atoms of the form $A(c)$ or $r(c_1, c_2)$, where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$ and $c, c_1, c_2 \in \mathsf{N_I}$. A TBox is a finite set of logical *axioms*, whose form depends on the particular DL. 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 "any KB" below is a shortcut for "any 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 *unique* canonical model up to isomorphism. 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.

An interpretation (or an ABox) 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 $\mathcal{I}$. Finally, ω is a *certain answer* to q over a KB $\mathcal{K}$ iff $\mathsf{range}(\omega) \subseteq \mathsf{aDom}(\mathcal{K})$ and 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:

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

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 [6]. 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 [10] 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 $\mathcal{I} = \mathcal{A} \cup \{\texttt{hasLicense}(\texttt{Alice}, 12345)\}$. Then $\mathsf{sparqlAns}(q, \mathcal{I}) = \{\{x \mapsto \texttt{Alice}, y \mapsto 12345\}\}$. 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. 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, $\mathsf{eCertAns}(q, \langle \emptyset, \mathcal{A} \rangle) = \{\{\}, \{x \mapsto \mathtt{Alice}\}\}$, whereas $\mathsf{sparqlAns}(q, \mathcal{A}) = \{\{x \mapsto \mathtt{Alice}\}\}$.

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 [14] 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).[1] 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 [12]) 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
$\mathcal{A} = \{\mathtt{teachesTo}(\mathtt{Alice}, \mathtt{Bob}), \mathtt{knows}(\mathtt{Bob}, \mathtt{Carol}), \mathtt{teachesTo}(\mathtt{Alice}, \mathtt{Dan})\}$
$q \;= \text{SELECT}_{\{x,z\}}(\mathtt{teachesTo}(x, y) \text{ OPT } \mathtt{knows}(y, z))$

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

Section 5.3 below defines a 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 [10]), 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 [11] 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 [6], and more recently in [7] and [9].

[1] This is without loss of expressivity, but normalization may cause an exponential blowup.

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.

We will show in Sect. 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 three additional requirements (called OPT *extension*, *variable binding* and *binding provenance*), which generalizes to KBs some basic properties of SPARQL answers over plain graphs. We note that these requirements apply to arbitrary DLs, whereas Sect. 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 Sect. 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 possible 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 [11] for instance, the OPT operator of SPARQL was introduced to "not reject the solutions because some part of the query pattern does not match" [8]. 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 `OPTIONAL`, 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 2 (Definition of $\mathsf{adm}(q)$ for the SUJO fragment)

$$\begin{array}{ll} \multicolumn{2}{l}{\textit{If } q \textit{ is a triple pattern, then } \mathsf{adm}(q) = \{\mathsf{vars}(q)\}} \\ \mathsf{adm}(\textsc{select}_X\ q) & = \{\ X' \cap X \quad \mid X' \in \mathsf{adm}(q)\ \} \\ \mathsf{adm}(q_1\ \textsc{join}\ q_2) & = \{\ X_1 \cup X_2 \ \mid (X_1, X_2) \in \mathsf{adm}(q_1) \times \mathsf{adm}(q_2)\ \} \\ \mathsf{adm}(q_1\ \textsc{opt}\ q_2) & = \mathsf{adm}(q_1) \cup \mathsf{adm}(q_1\ \textsc{join}\ q_2) \\ \mathsf{adm}(q_1\ \textsc{union}\ q_2) & = \mathsf{adm}(q_1) \cup \mathsf{adm}(q_2) \end{array}$$

We can now formulate the corresponding requirement:

Requirement 4 (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 Requirement 4 on variable bindings does not enforce this property. So we add as a simple fifth requirement:

Requirement 5 (Binding provenance)**.** *For any SUJO queries q_1, q_2, KB $\mathcal{K}$ and solution mapping ω:*

$$\textit{if } \omega \in \mathsf{ans}(q_1\ \textsc{union}\ q_2) \textit{ and } \omega \notin \mathsf{ans}(q_2), \textit{ then } \mathsf{dom}(\omega) \in \mathsf{adm}(q_1)$$
$$\textit{if } \omega \in \mathsf{ans}(q_1\ \textsc{union}\ q_2) \textit{ and } \omega \notin \mathsf{ans}(q_1), \textit{ then } \mathsf{dom}(\omega) \in \mathsf{adm}(q_2)$$

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, 4 and 5 (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 or 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. Section 5.2 focuses on DLs with the canonical model property. For these, 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. Finally, Sect. 5.3 builds upon this last observation, and shows that it is possible to define a semantics that satisfies all requirements for the SUJO fragment.

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

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	REQ5
Ahmetaj et al. [1]	pwdPT ($\subseteq$ SJO)	✓	x	?	✓	✓
Entailment regime (Definition 3)	UJO	✓	✓	✓	✓	✓
	SJ/SUJO	x	✓	✓	✓	✓
Canonical (Definition 4)	O/SUJO	✓	✓	x	✓	✓
Restricted (Definition 5)	SUJO	✓	✓	✓	x	x
Max. adm. can. (Definition 8)	SUJO	✓	✓	✓	✓	✓

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 Sect. 2).

Because the specification of SPARQL entailment regimes [6] is too low-level for the scope of this paper, we provide a more abstract characterization of this approach for the SUJO fragment. 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 3 (Entailment Regime Answers)

$$\begin{array}{lll} \multicolumn{3}{l}{\textit{If } q \textit{ is a triple pattern, then } \mathsf{eRAns}(q,\mathcal{K}) = \mathsf{certAns}(q,\mathcal{K})} \\ \mathsf{eRAns}(q_1 \text{ UNION } q_2,\mathcal{K}) & = & \mathsf{eRAns}(q_1,\mathcal{K}) \cup \mathsf{eRAns}(q_2,\mathcal{K}) \\ \mathsf{eRAns}(q_1 \text{ JOIN } q_2,\mathcal{K}) & = & \mathsf{eRAns}(q_1,\mathcal{K}) \bowtie \mathsf{eRAns}(q_2,\mathcal{K}) \\ \mathsf{eRAns}(q_1 \text{ OPT } q_2,\mathcal{K}) & = & (\mathsf{eRAns}(q_1,\mathcal{K}) \bowtie \mathsf{eRAns}(q_2,\mathcal{K})) \cup \\ & & (\mathsf{eRAns}(q_1,\mathcal{K}) \setminus \mathsf{eRAns}(q_2,\mathcal{K})) \\ \mathsf{eRAns}(\text{SELECT}_X\ q,\mathcal{K}) & = & \pi_X \mathsf{eRAns}(q,\mathcal{K}) \end{array}$$

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 3) 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
$\mathcal{A} = \{\texttt{Driver(Alice)}\}$
$\mathcal{T} = \{\texttt{Driver} \sqsubseteq \exists\texttt{hasLicense}\}$
$q \ = \text{SELECT}_{\{x\}}\ (\texttt{Driver}(x)\ \text{JOIN}\ \texttt{hasLicense}(x, y))$
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 its canonical model (up to isomorphism).

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 Requirements 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 4 (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 comply with SPARQL answers over a plain graph (Requirement 2).

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 $\mathsf{canAns}(q, \mathcal{K}) = \mathsf{sparqlAns}(q, \mathsf{can}(\mathcal{K})) \triangleright \mathsf{aDom}(\mathcal{K}) = \emptyset$. So $\mathsf{canAns}(\texttt{Driver}(x), \mathcal{K}) \not\preceq_g \mathsf{canAns}(q, \mathcal{K})$, which immediately 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 4, 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 want 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 5 (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 Requirements 4 and 5. We call the resulting solution mappings *maximal admissible canonical answers*, noted $\mathsf{mCanAns}(q, \mathcal{K})$.

We start with the PJO fragment (i.e. queries without UNION) for simplicity, since for this fragment, Requirement 5 is trivially satisfied. 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 6 (Maximal Admissible Canonical Answers (SJO))

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

In order to generalize this definition to queries with UNION, we need to enforce Requirement 5. To this end, the provenance of each solution mapping needs to be taken into account. We define the set of *branches* of a SUJO query q, denoted with $\mathsf{branch}(q)$, 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 $\mathsf{branch}(q) = \{A(x)$ OPT $R_1(x, y), A(x)$ OPT $R_2(x, z)\}$. The function $\mathsf{branch}(q)$ is defined inductively over q as expected:

Definition 7 (Branches of a SUJO query q)

If q is a triple pattern, then $\mathsf{branch}(q) = \{q\}$
$\mathsf{branch}(\text{SELECT}_X\ q) = \{\ \text{SELECT}_X\ q' \mid q' \in \mathsf{branch}(q)\ \}$
$\mathsf{branch}(q_1\ \text{JOIN}\ q_2) = \{\ q'_1\ \text{JOIN}\ q'_2 \mid (q'_1, q'_2) \in \mathsf{branch}(q_1) \times \mathsf{branch}(q_2)\ \}$
$\mathsf{branch}(q_1\ \text{OPT}\ q_2) = \{\ q'_1\ \text{OPT}\ q'_2 \mid (q'_1, q'_2) \in \mathsf{branch}(q_1) \times \mathsf{branch}(q_2)\ \}$
$\mathsf{branch}(q_1\ \text{UNION}\ q_2) = \mathsf{branch}(q_1) \cup \mathsf{branch}(q_2)$

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. [15, Example 1]). Or formally, for any SUJO query q and graph $\mathcal{A}$:

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

So if $q' \in \mathsf{branch}(q)$, 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 6 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 8 (Maximal Admissible Canonical Answers (SUJO))

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

It can be easily verified that Definitions 6 and 8 coincide for SJO queries, since in this case $\mathsf{branch}(q) = \{q\}$. Proposition 2 shows that maximal admissible canonical answers satisfy all requirements expressed in the previous section.

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, 4 and 5.*

Table 2. Combined complexity of $\text{EVAL}_{\mathsf{sparqlAns}}$ and $\text{EVAL}_{\mathsf{mCanAns}}$. "-c" stands for complete, and "A/B" for all fragments between A and B.

Fragment	$\text{EVAL}_{\mathsf{sparqlAns}}$	$\text{EVAL}_{\mathsf{mCanAns}}$
UJ/SUJ	NP-c	NP-c
Well-designed JO	coNP-c	coNP-c
Well-designed SJO*	Σ_2^{P}-c	Σ_2^{P}-c
OJ/SUJO	PSpace-c	PSpace-c

6 Complexity

We now provide complexity results for query answering under the semantics defined in Sect. 5.3, 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 problem $\text{EVAL}_{\mathsf{mCanAns}}$ below. We also 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, or of the query and TBox) as future work.

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

Complexity of SPARQL query evaluation over plain graphs has been extensively studied (see [13] for a recent overview). When these results are tight, they provide us immediate lower bounds. Indeed, from Proposition 1, certain canonical answers satisfy Requirement 2, so $\text{EVAL}_{\mathsf{mCanAns}}$ is at least as hard as the problem $\text{EVAL}_{\mathsf{sparqlAns}}$ below.

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

Table 2 reproduces results for $\text{EVAL}_{\mathsf{sparqlAns}}$ 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 $\text{EVAL}_{\mathsf{sparqlAns}}$ has been shown to be PSpace-complete for the OJ fragment already, in [15]. Particular attention has also been paid to so-called *well-designed* SJO and JO queries (see [14] for a definition), which have a natural representation as *pattern trees* [12], with a significant reduction from PSpace to Σ_2^{P} and coNP-completeness respectively. For SJO, we follow [12] 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 [15], thus contrasting with projection-free UCQs.

So for each fragment, we investigate whether $\text{EVAL}_{\mathsf{mCanAns}}$ matches the upper bounds for $\text{EVAL}_{\mathsf{sparqlAns}}$. The results are summarized in Table 2. Interestingly, all upper bounds are matched. This means that for these fragments, 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 [5], which matches the (NP) upper bound for UCQs over a plain graph.

Before explaining these results, we isolate a key observation:

Proposition 3. *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. (Sketch.) If q is a JO query, we compute a family $\mathsf{base}(q)$ of sets of variables s.t. $|\mathsf{base}(q)| = O(|q|)$, and s.t. each $V \in \mathsf{adm}(q)$ is the union of some elements of $\mathsf{base}(q)$ and conversely, i.e. $\mathsf{adm}(q) = \{\bigcup \mathcal{B} \mid \mathcal{B} \in 2^{\mathsf{base}(q)}\}$. The family $\mathsf{base}(q)$ can be computed inductively as follows:

- 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 $q_2)$

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 [15] for hardness of EVAL$_{\mathsf{sparqlAns}}$ in this fragment, deciding $X \in \mathsf{adm}(q)$ for any X and UJ query q is NP-hard (we refer to the the extended version of this paper for details).

We now sketch the argument used to derive upper bounds for the SUJO, well-designed SJO* and UJ fragments (proofs can be found in the extended version). 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 certain canonical answers in this case is more involved (compare Definitions 6 and 8 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 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 decide whether $\omega \in \mathsf{mCanAns}(q, \mathcal{K})$ consists in guessing an extension ω' or ω, then verify *(i)*, and then verify *(ii)*. From Proposition 3 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 decide *(ii)*, it is sufficient to guess $\mathcal{G}$, then verify that $\mathcal{G}$ is a subgraph of $\mathsf{can}(\mathcal{K})$, and then decide whether $\omega' \in \mathsf{sparqlAns}(q, \mathcal{G})$. Since $\text{EVAL}_{\mathsf{sparqlAns}}$ is in Σ_2^{P}, whether $\omega' \in \mathsf{sparqlAns}(q, \mathcal{G})$ can be nondeterministically decided 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 [14], deciding $\omega \in \mathsf{sparqlAns}(q, \mathcal{A}')$ is in coNP.

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 graph 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 for the fragment of SPARQL with `SELECT`, `UNION` and `OPTIONAL` and DLs with the canonical model property. More precisely, we defined a semantics that matches all requirements. We also provided combined complexity results for query answering over a *DL-Lite*$_\mathcal{R}$ KB under this semantics.

This work is still at an early stage, for multiple reasons. First, the semantics we defined is arguably ad-hoc, with a procedural flavor, and it would be interesting to investigate whether it can be characterized in a more declarative fashion. It must also be emphasized that if query answers defined by this semantics comply with all requirements, whether the converse holds (i.e. whether there may be answers that comply with all requirements, but are not returned under this semantics) is still an open question.

Data complexity may also be investigated, as well as algorithmic aspects, in particular *FO-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 [16]. Other DLs and/or fragments of SPARQL may also be considered.

Finally, and more importantly, additional requirements may be identified, possibly violated by the semantics we defined. If so, a key question is whether such an extended set of requirements can still be matched, for reasonably expressive DLs and fragments of SPARQL. A negative answer would constitute an argument for the SPARQL entailment regimes (or the extension of the OWL 2 QL regime proposed in [10]) as a default solution.

Acknowledgments. This research has been partially supported by the EU H2020 project INODE, by the Italian PRIN project HOPE, and by the Free University of Bozen-Bolzano through the projects QUADRO, KGID and GeoVKG.

References

1. Ahmetaj, S., Fischl, W., Pichler, R., Šimkus, M., Skritek, S.: Towards reconciling SPARQL and certain answers. In: Proceedings of the 24th International Conference on World Wide Web, pp. 23–33. ACM (2015)
2. Arenas, M., Pérez, J.: Querying semantic web data with SPARQL. In: Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 305–316. ACM (2011)
3. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. **36**, 1–69 (2009)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J. Autom. Reason. **39**(3), 385–429 (2007)
6. Glimm, B., Ogbuji, C.: SPARQL 1.1 entailment regimes. Technical report, W3C, March 2013
7. Gutierrez, C., Hernández, D., Hogan, A., Polleres, A.: Certain answers for SPARQL? In: AMW (2016)
8. Harris, S., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 query language. W3C recommendation, W3C (2013)
9. Hernández, D., Gutierrez, C., Hogan, A.: Certain answers for SPARQL with blank nodes. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 337–353. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_20
10. Kontchakov, R., Rezk, M., Rodríguez-Muro, M., Xiao, G., Zakharyaschev, M.: Answering SPARQL queries over databases under OWL 2 QL entailment regime. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 552–567. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_35
11. Kostylev, E.V., Cuenca Grau, B.: On the semantics of SPARQL queries with optional matching under entailment regimes. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8797, pp. 374–389. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11915-1_24
12. Letelier, A., Pérez, J., Pichler, R., Skritek, S.: Static analysis and optimization of semantic web queries. ACM Trans. Database Syst. (TODS) **38**(4), 25 (2013)
13. Mengel, S., Skritek, S.: On tractable query evaluation for SPARQL. arXiv preprint arXiv:1712.08939 (2017)
14. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. (TODS) **34**(3), 16 (2009)
15. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL query optimization. In: Proceedings of the 13th International Conference on Database Theory, pp. 4–33. ACM (2010)
16. Xiao, G., et al.: 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, pp. 5511–5519, July 2018