Towards Reconciling SPARQL and Certain Answers

Shqiponja Ahmetaj
TU Vienna
ahmetaj@dbai.tuwien.ac.at

Wolfgang Fischl
TU Vienna
wfischl@dbai.tuwien.ac.at

Reinhard Pichler
TU Vienna
pichler@dbai.tuwien.ac.at

Mantas Šimkus
TU Vienna
simkus@dbai.tuwien.ac.at

Sebastian Skritek
TU Vienna
skritek@dbai.tuwien.ac.at

ABSTRACT

SPARQL entailment regimes are strongly influenced by the big body of works on ontology-based query answering, notably in the area of Description Logics (DLs). However, the semantics of query answering under SPARQL entailment regimes is defined in a more naive and much less expressive way than the certain answer semantics usually adopted in DLs. The goal of this work is to introduce an intuitive certain answer semantics also for SPARQL and to show the feasibility of this approach. For OWL 2 QL entailment, we present algorithms for the evaluation of an interesting fragment of SPARQL (the so-called well-designed SPARQL). Moreover, we show that the complexity of the most fundamental query analysis tasks (such as query containment and equivalence testing) is not negatively affected by the presence of OWL 2 QL entailment under the proposed semantics.

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages—*Query Languages*; H.2.5 [**Database Management**]: Heterogeneous Databases

General Terms

Theory, Algorithms

Keywords

SPARQL; Certain Answers; DL-Lite; Query Answering; Query Rewriting; Complexity

1. INTRODUCTION

In the recently released recommendation [12], the W3C has defined various SPARQL entailment regimes to allow users to specify implicit knowledge about the vocabulary in an RDF graph (see [10] for a tutorial). The theoretical underpinning to the systems for query answering under rich entailment regimes (see e.g.[16, 6, 30]) is provided by the big body of work on ontology-based query answering, notably in the area of Description Logics (DLs) [3]. Superficially, query answering of basic graph patterns (BGPs) under an entailment regime looks exactly like answering conjunctive queries (CQs) under the corresponding DL. However, there is a huge difference between the two in that SPARQL entailment regimes do not consider non-distinguished variables:

Example 1. Consider an RDF graph G containing a single triple $(b, \mathsf{a}, \mathsf{Prof})$ – stating that b is a professor – and an ontology $\mathcal{O}$ containing a single concept inclusion stating that every professor teaches somebody. In our notation introduced in Section 2, this inclusion will be denoted as $(\mathsf{Prof}, \texttt{rdfs:sc}, \exists\mathsf{teaches})$.
Now consider the following simple SPARQL query:
SELECT $?x$ WHERE $(?x, \mathsf{teaches}, ?y)$.[1]
According to the SPARQL entailment regimes standard [12], this query yields as result the empty set.

Clearly, the empty set as the result in the above example is rather unintuitive: by the inclusion $(\mathsf{Prof}, \texttt{rdfs:sc}, \exists\mathsf{teaches})$, we know for certain that b teaches somebody. However, the SPARQL entailment standard requires that all values assigned to any variable in the BGP must come from the RDF graph – thus treating distinguished variables (which are ultimately output) and non-distinguished variables (which are eventually projected out) in the same way. In contrast, query answering in the DL world is based on the *certain answer semantics*. We thus consider the models of the database (i.e., the RDF graph) and the ontology and we accept all those mappings as solutions that make the query true in every possible model. In the above example, every such model contains a triple $(b, \mathsf{teaches}, u)$ for some value u. Hence, even though there are distinct values u in different models, the projection onto the first component always yields the mapping $\mu = \{?x \to b\}$ as an answer to the query.

Depending on the expressive power of a particular DL, query answering under the certain answer semantics may be computationally very expensive. E.g. the problem is often 2EXPTIME-complete for the so-called *very expressive DLs* [21, 11]. However, there exists a family of DLs with reasonable expressive power and very nice computational properties, namely the DL-Lite family [7]. A member of this family is DL-Lite$_\mathcal{R}$, which provides the theoretical underpinning of the OWL 2 QL entailment regime. Recall that – even without taking any DLs into account – answering CQs as well as basic query analysis tasks like testing containment or equivalence are NP-complete [8]. Somehow surprisingly, this complexity does not increase if we consider CQ answering and CQ containment/equivalence under DL-Lite [7]. In other words, the significant increase of expressive power has no serious effect on the complexity. The **goal of this work** is to introduce an intuitive certain answer semantics also for SPARQL under OWL 2 QL entailment with similarly favourable results as for CQ answering under DL-Lite.

[1]Following [25], we use a more algebraic style notation, denoting triples in parentheses with comma-separated components rather than the blank-separated turtle notation.

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741636.

If we were contented with evaluating BGPs under OWL 2 QL according to the certain answer semantics, we could literally take over all results on CQ answering under DL-Lite. But of course, there are more facets to SPARQL than just BGPs. A crucial extension of BGPs is given by the OPTIONAL operator (henceforth referred to as OPT operator, for short). It allows one to express that a mapping should mandatorily bind some of the variables in the query and, if possible, it should be extended to further variables. If the extension is impossible, we simply keep the unextended mapping – but we do not dismiss it.

The OPT operator has an enormous impact on the complexity of query evaluation: indeed, even if SPARQL is restricted to the use of conjunction and the OPT operator, query evaluation is PSPACE-complete [29]. In [25], the authors therefore introduced *well-designed SPARQL* as an interesting fragment where certain restrictions on the occurrence of variables are imposed. It is shown in [25] that for well-designed SPARQL, the complexity of query evaluation drops from PSPACE-completeness to coNP-completeness. Moreover, as shown in [19], well-designed SPARQL queries have a very nice and intuitive representation by so-called *pattern trees*, i.e.: rooted, unordered trees where the nodes are labelled with BGPs and the tree structure reflects the nesting of OPTs (for details on well-designed SPARQL and pattern trees, see Section 2).

Recall that the natural definition of certain answers to a query is to collect all those mappings that are a solution in all possible models of the data and the ontology. As long as we restrict ourselves to answering conjunctive queries, this notion of certain answers is intuitive and constitutes the generally agreed semantics. In contrast, as soon as we extend CQs by the OPT operator, this approach turns out to be highly unsatisfactory, as the following example illustrates.

Example 2. Consider the SPARQL query:

$$\text{SELECT } ?x, ?z \text{ WHERE } (?x, \mathsf{teaches}, ?y) \\ \text{OPT } (?y, \mathsf{knows}, ?z)$$

over the graph $G = \{(b, \mathsf{teaches}, c)\}$ and empty ontology $\mathcal{O}$. The query yields as only solution the mapping $\mu = \{?x \to b\}$. Clearly, also the graph $G' = G \cup \{(c, \mathsf{knows}, d)\}$ is a model of $(G, \mathcal{O})$. But in G', μ is no longer a solution since μ can be extended to solution $\mu' = \{?x \to b, ?z \to d\}$. Hence, there exists no mapping which is a solution in every possible model of $(G, \mathcal{O})$.

The reason for the unintuitive behaviour in the above example is the non-monotonicity of the OPT operator. However, as observed in [2], even if well-designed SPARQL is non-monotone, at least it is *weakly monotone* in the following sense: if a model M' extends a model M by introducing additional facts, then every solution of a well-designed SPARQL query Q in M *can be extended* to a solution of Q in M'. Hence, the key idea for an appropriate definition of certain answers of well-designed SPARQL under OWL 2 QL entailment will be to consider those mappings as certain which can be extended to a solution in all possible models of the RDF graph and the ontology. Actually, yet a further modification of the semantics definition will be required in order to arrive at a uniquely defined, intuitive semantics without resorting to bag semantics. The details are worked out in Section 3.

The special behaviour of the OPT operator and the modification of the certain answer semantics require an adaptation and extension of the CQ answering algorithms for DL-Lite. Moreover, also the complexity of query evaluation and of containment/equivalence testing for well-designed SPARQL requires a new analysis in the presence of OWL 2 QL entailment. We shall show that, analogously to CQ answering under DL-Lite, the additional expressive power due to OWL 2 QL entailment can be obtained without paying a price in terms of complexity.

Organization and main results. In Section 2, we recall some basic notions and results. A conclusion and outlook to future work are given in Section 8. Our main results are detailed in Sections 3 – 7:

- In Section 3, we discuss the difficulties with a literal adoption of certain answer semantics in the presence of the non-monotone OPT operator. To overcome these difficulties, we propose a modified definition of certain answers for well-designed SPARQL queries. By relating the resulting semantics to SPARQL evaluation without entailment regimes we illustrate the naturalness of our definition.

- Several adaptations and extensions of the rewriting-based CQ evaluation under DL-Lite [7] are needed in order to incorporate the OPT operator. In Sections 4 and 5, we present two different approaches to answering well-designed SPARQL queries under OWL 2 QL entailment: the first one proceeds in a modular way by rewriting each BGP individually. A new data structure – so-called guides – has to be introduced to ensure the consistency of local solutions for each BGP when combining them to a solution of the overall SPARQL query. Our second algorithm takes a holistic approach by rewriting the entire well-designed SPARQL query at once. The CQ-rewriting under DL-Lite has to be significantly extended to take the peculiarities of the OPT operator into account. The two algorithms are discussed in Section 6.

- In Section 7, we analyse the complexity of well-designed SPARQL evaluation under OWL 2 QL entailment as well as the complexity of basic query analysis tasks such as query containment and query equivalence. We show that – analogously to CQs under DL-Lite – the complexity remains essentially the same no matter whether we consider well-designed SPARQL with or without OWL 2 QL entailment.

Due to space limitations, we only give proof ideas of our results. All details will be provided in the full version of the paper.

2. PRELIMINARIES

RDF graphs and OWL 2 QL. We let $\mathbf{U}$ denote a countably infinite set of *URIs*. An *RDF triple* t is any tuple $t \in \mathbf{U} \times \mathbf{U} \times \mathbf{U}$. An *RDF graph* G is any (possibly infinite) set $G \subseteq \mathbf{U} \times \mathbf{U} \times \mathbf{U}$ of triples. Note that we do not allow blank nodes, i.e. all RDF triples and graphs are *ground*. The *active domain* of an RDF graph G, denoted $\mathsf{dom}(G)$, is the set of all URIs that appear in G.

We next define *membership and inclusion triples*, which are special triples to enrich RDF graphs with OWL 2 QL or RDFS semantic information. Let $\mathbf{C}$, $\mathbf{A}$ and $\mathbf{R}$ be countably infinite and mutually disjoint subsets of $\mathbf{U}$ that denote *constants*, *atomic concepts* (a.k.a. *classes*) and *atomic roles* (a.k.a. *properties*), respectively. A *basic role* Q is either an atomic role from $\mathbf{R}$ or an expression R^-, where $R \in \mathbf{R}$. A *basic concept* B is either an atomic concept $A \in \mathbf{A}$ or an expression $\exists Q$, where Q is a basic role.

We further assume the dedicated URIs [2]

- "a" to assert membership of a constant in a concept,

- "rdfs:sc" to assert the inclusion relation between concepts,

- "rdfs:sp" to assert the inclusion relation between roles,

- "owl:disjointWith" to assert *concept disjointness*, and

- "owl:propertyDisjointWith" to assert *role disjointness*.

[2]For the special RDF property rdf:type we use the common abbreviation "a". For the RDFS properties rdfs:subClassOf and rdfs:subPropertyOf we use the abbreviations "rdfs:sc" and "rdfs:sp", resp.

A *membership assertion* (*MA*) is any triple of the form (c, a, B) or (c, Q, c'), where c, c' are constants from $\mathbf{C}$, B is a basic concept and Q is a basic role. For a role R, we simplify $(R^-)^-$ to R. Given an RDF graph G, we write

- $G \models (c, Q, c')$ if $(c, Q, c') \in G$ or $(c', Q^-, c) \in G$,
- $G \models (c, \mathsf{a}, A)$ if $(c, \mathsf{a}, A) \in G$, and
- $G \models (c, \mathsf{a}, \exists Q)$ if $G \models (c, Q, c')$ for some c'.

We define the notion of *inclusions*. An *RDFS inclusion* is any triple having the form $(B, \texttt{rdfs:sc}, A)$ or $(Q_1, \texttt{rdfs:sp}, Q_2)$, where B is a basic concept, A is an atomic concept, and Q_1, Q_2 are basic roles. An *OWL 2 QL inclusion* is any RDFS inclusion or a triple of the form (i) $(B_1, \texttt{rdfs:sc}, B_2)$, (ii) $(B_1, \texttt{owl:disjointWith}, B_2)$ or (iii) $(Q_1, \texttt{owl:propertyDisjointWith}, Q_2)$, where B_1, B_2 are basic concepts and Q_1, Q_2 are basic roles. Given an RDF graph G, we write

- $G \models (B_1, \texttt{rdfs:sc}, B_2)$ if $G \models (a, \mathsf{a}, B_1)$ implies $G \models (a, \mathsf{a}, B_2)$ for any URI a,
- $G \models (Q_1, \texttt{rdfs:sp}, Q_2)$ if $G \models (a, Q_1, b)$ implies $G \models (a, Q_2, b)$ for any pair a, b of URIs,
- $G \models (B_1, \texttt{owl:disjointWith}, B_2)$ if $G \models (a, \mathsf{a}, B_1)$ implies $G \not\models (a, \mathsf{a}, B_2)$ for any URI a, and
- $G \models (Q_1, \texttt{owl:propertyDisjointWith}, Q_2)$ if $G \models (a, Q_1, b)$ implies $G \not\models (a, Q_2, b)$ for any pair a, b of URIs.

Concepts $\exists Q$ in triples $(B, \texttt{rdfs:sc}, \exists Q)$ are to simplify presentation; we can write $(\exists Q, \texttt{rdfs:sc}, A)$, $(\exists Q^-, \texttt{rdfs:sc}, A)$ instead of $(Q, \texttt{rdfs:domain}, A)$, $(Q, \texttt{rdfs:range}, A)$, respectively. The role R^- is a convenient representation of the role r_1, where $(r_1, \texttt{owl:inverseOf}, R)$. Similar, triples of the form $(B, \texttt{rdfs:sc}, \exists Q)$ represent the following OWL 2 triples:

$$(B, \texttt{rdfs:sc}, c_1), (c_1, \mathsf{a}, \texttt{owl:Restriction}),$$
$$(c_1, \texttt{owl:onProperty}, Q),$$
$$(c_1, \texttt{owl:someValuesFrom}, \texttt{owl:Thing}).$$

An *OWL 2 QL ontology* (resp., *RDFS ontology*) $\mathcal{O}$ is any set of OWL 2 QL (resp., RDFS) inclusions. W.l.o.g. we assume $(Q_1^-, \texttt{rdfs:sp}, Q_2^-) \in \mathcal{O}$ for each $(Q_1, \texttt{rdfs:sp}, Q_2) \in \mathcal{O}$. Given a graph G and an ontology $\mathcal{O}$, we write $G \models \mathcal{O}$ if $G \models \sigma$ for all $\sigma \in \mathcal{O}$. A *knowledge base* (KB) is any pair $\mathcal{G} = (G, \mathcal{O})$, where G is a graph and $\mathcal{O}$ is an ontology. Let $\mathcal{G} = (G, \mathcal{O})$ be a KB. Given a graph G', we say G' is a *model* of $\mathcal{G}$, denoted by $G' \models \mathcal{G}$ if $G \subseteq G'$ and $G' \models \mathcal{O}$. We say $\mathcal{G}$ is *consistent* if there exists a model G', s.t. $G' \models \mathcal{G}$. Finally, for an RDF triple t, we say $\mathcal{G} \models t$ if $G' \models t$ for every $G' \models \mathcal{G}$. Analogously, $\mathcal{O} \models t$ if $G' \models t$ for every $G' \models \mathcal{O}$.

SPARQL. Let $\mathbf{V}$ be an infinite set of variables, s.t. $\mathbf{U} \cap \mathbf{V} = \emptyset$. We denote variables with a leading question mark, e.g. $?x$. A SPARQL triple pattern t is a tuple in $(\mathbf{U} \cup \mathbf{V}) \times (\mathbf{U} \cup \mathbf{V}) \times (\mathbf{U} \cup \mathbf{V})$. Throughout this paper we will consider only triple patterns of the form $(?x, \mathsf{a}, B)$ or $(?x, Q, ?y)$, where B is a basic concept and Q a basic role.[3]

The set of SPARQL graph patterns is defined recursively as follows: (1) a triple pattern t is a graph pattern ; (2) if P_1 and P_2 are graph patterns, then $(P_1 \text{ UNION } P_2)$, $(P_1 \text{ AND } P_2)$, $(P_1 \text{ OPT } P_2)$ are graph patterns. Note that we follow the algebraic style notation from [25], where an explicit AND operator is used. For a graph pattern P, we write $\mathsf{vars}(P)$, to denote the set of variables occurring in P.

In order to define the semantics of SPARQL graph patterns, we first introduce some additional terminology. A *substitution* is any partial function σ from $\mathbf{V}$ to $\mathbf{U} \cup \mathbf{V}$. We use $\mathsf{dom}(\sigma)$ and $\mathsf{rng}(\sigma)$ to denote the domain and the range of σ, respectively. Given a triple pattern t and a substitution σ, we use $\sigma(t)$ to denote the triple obtained from t by replacing every variable $?x \in \mathsf{vars}(t) \cap \mathsf{dom}(\sigma)$ by $\sigma(?x)$. A *(SPARQL) mapping* is any partial function μ from $\mathbf{V}$ to $\mathbf{U}$. Note that mappings are substitutions. Two mappings μ_1 and μ_2 are *compatible* (written $\mu_1 \sim \mu_2$) if $\mu_1(?x) = \mu_2(?x)$ for all $?x \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$. A mapping μ_1 is *subsumed* by μ_2 (written $\mu_1 \sqsubseteq \mu_2$) if $\mu_1 \sim \mu_2$ and $\mathsf{dom}(\mu_1) \subseteq \mathsf{dom}(\mu_2)$. We write $\mu_1 \sqsubset \mu_2$ if $\mu_1 \sqsubseteq \mu_2$ and $\mu_2 \not\sqsubseteq \mu_1$. For a mapping μ and sets M, M' of mappings, we say $\mu \sqsubseteq M$ if $\mu \sqsubseteq \mu'$ for some $\mu' \in M$, and $M' \sqsubseteq M$ if $\mu \sqsubseteq M$ for every $\mu \in M'$.

We formalize the evaluation of graph patterns over an RDF graph G as a function $[\![\cdot]\!]_G$ that, given a graph pattern, returns a set of mappings. For a graph pattern P, it is defined as follows [25]:

1. $[\![t]\!]_G = \{\mu \mid \mathsf{dom}(\mu) = \mathsf{vars}(t) \text{ and } \mu(t) \in G\}$ for a triple pattern t.
2. $[\![P_1 \text{ AND } P_2]\!]_G = \{\mu_1 \cup \mu_2 \mid \mu_1 \in [\![P_1]\!]_G, \mu_2 \in [\![P_2]\!]_G, \text{ and } \mu_1 \sim \mu_2\}$.
3. $[\![P_1 \text{ OPT } P_2]\!]_G = [\![P_1 \text{ AND } P_2]\!]_G \cup \{\mu_1 \in [\![P_1]\!]_G \mid \forall \mu_2 \in [\![P_2]\!]_G : \mu_1 \not\sim \mu_2\}$.
4. $[\![P_1 \text{ UNION } P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$.

A *basic graph pattern (BGP)* is a set of triple patterns $P = \{t_1, \ldots, t_n\}$ – equivalently written as t_1 AND ... AND t_n. Hence, we have $[\![P]\!]_G = \{\mu \mid \mathsf{dom}(\mu) = \mathsf{vars}(P), \text{ and } \mu(t) \in G \text{ for all } t \in P\}$. Note that, as in [25] and in the vast majority of works on query answering in DLs, we assume set semantics. In this paper we focus on *well-designed* SPARQL graph patterns. A graph pattern P, built only from AND and OPT is well-designed if there does not exist a subpattern $P' = (P_1 \text{ OPT } P_2)$ of P and a variable $?x \in \mathsf{vars}(P_2)$ that occurs in P outside of P', but not in P_1. A graph pattern $P = P_1 \text{ UNION } \ldots \text{ UNION } P_n$ is well-designed if each subpattern P_i is UNION free and well-designed.

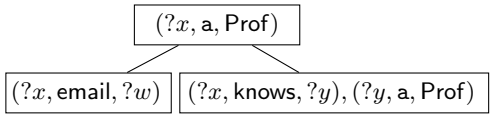
Pattern trees. Another class of graph patterns defined in [25] are graph patterns in OPT *normal form*. A pattern containing only the operators AND and OPT is in OPT normal form if the OPT operator never occurs in the scope of an AND operator. It was shown that every well-designed graph pattern can be transformed into OPT normal form in polynomial time. Moreover, such graph patterns allow for a natural tree representation, formalized by so-called *pattern trees* in [19]. A *pattern tree (PT)* $\mathcal{T}$ is a pair $(T, \mathcal{P})$, where $T = (V, E, r)$ is a rooted, unordered, tree and $\mathcal{P} = \{P_n \mid n \in V\}$ is a labelling of the nodes in V, s.t. P_n is a non-empty set of triple patterns for every $n \in V$. Given $\mathcal{T}$, we write $V(\mathcal{T})$ to denote the set V of vertices. The next example illustrates this idea.

Example 3. The following SPARQL graph pattern asks for all professors $?x$ and, if available, their email address $?w$. Independently of the available information on the email address $?w$, the pattern also tries to retrieve all professors $?y$ that $?x$ knows.

$$\big((?x, \mathsf{a}, \mathsf{Prof}) \text{ OPT } (?x, \mathsf{email}, ?w)\big) \text{ OPT } \big((?x, \mathsf{knows}, ?y) \text{ AND } (?y, \mathsf{a}, \mathsf{Prof})\big)$$

The corresponding pattern tree looks as follows:

[3] Constants in place of $?x$ and $?y$ can be simulated by using fresh variables and auxiliary triples in queries and input RDF graphs.



Let $\mathcal{T} = ((V, E, r), \mathcal{P})$ be a pattern tree. We call a PT $\mathcal{T}' = ((V', E', r'), \{P_n \mid n \in V'(\mathcal{T}')\})$ a *subtree of* $\mathcal{T}$ if (V', E', r') is a subtree of (V, E, r). Throughout this article we only consider subtrees containing the root (i.e. $r = r'$), and will thus refer to them simply as "subtrees", omitting the phrase "containing the root". For a PT $\mathcal{T} = (T, \mathcal{P})$, we denote with $\mathsf{pat}(\mathcal{T})$ the set $\bigcup_{n \in V(\mathcal{T})} P_n$ of triple patterns occurring in $\mathcal{T}$. We write $\mathsf{vars}(\mathcal{T})$ (resp. $\mathsf{vars}(n)$ for $n \in V(\mathcal{T})$) as an abbreviation for $\mathsf{vars}(\mathsf{pat}(\mathcal{T}))$ (resp. $\mathsf{vars}(\mathsf{pat}(n))$).

A *well-designed pattern tree (wdPT)* is a pattern tree $\mathcal{T} = (T, \mathcal{P})$ where for every variable $?x \in \mathsf{vars}(\mathcal{T})$, the nodes $\{n \in V(\mathcal{T}) \mid ?x \in \mathsf{vars}(n)\}$ induce a connected subgraph of T.

Recall that wdPTs are a *representation* of well-designed SPARQL graph patterns. Also, for every wdPT $\mathcal{T}$, there exists a straight forward translation into a well-designed graph pattern $P_{\mathcal{T}}$ in OPT normal form. Moreover, all well-designed graph patterns represented by $\mathcal{T}$ are equivalent to $P_{\mathcal{T}}$ (see [19] for details). For some RDF graph G, we thus define $[\![\mathcal{T}]\!]_G = [\![P_{\mathcal{T}}]\!]_G$.

Projection. Projection in SPARQL is realized via the SELECT result modifier on top of graph patterns (or equivalently of pattern trees). For a mapping μ and a set $\mathcal{X}$ of variables, let $\mu|_{\mathcal{X}}$ denote the projection of μ to the variables in $\mathcal{X}$, that is, the mapping μ' defined as $\mathsf{dom}(\mu') := \mathcal{X} \cap \mathsf{dom}(\mu)$ and $\mu'(?x) := \mu(?x)$ for all $?x \in \mathsf{dom}(\mu')$. We extend projection to a set M of mappings as $M|_{\mathcal{X}} = \{\mu|_{\mathcal{X}} \mid \mu \in M\}$. The result of projecting a graph pattern P to $\mathcal{X}$ is defined as $[\![(P, \mathcal{X})]\!]_G = \{\mu|_{\mathcal{X}} \mid \mu \in [\![P]\!]_G\}$. Analogously, we define $[\![(\mathcal{T}, \mathcal{X})]\!]_G = \{\mu|_{\mathcal{X}} \mid \mu \in [\![\mathcal{T}]\!]_G\}$ for a wdPT $\mathcal{T}$. We refer to the pair $(\mathcal{T}, \mathcal{X})$ as a *projected wdPT (pwdPT)*.

3. CERTAIN ANSWER SEMANTICS

In this section, we propose an intuitive definition of certain answers for well-designed SPARQL queries or, more precisely, for pwdPTs. Moreover, we shall establish the relationship of these certain answers with the canonical model – an important tool in the DL world. A natural first idea (inspired by investigations in [2] on the relationship between the official SPARQL semantics and the open world assumption for RDF data) was already mentioned in Section 1, namely: rather than requesting that certain answers must be a solution in every possible model, at least one should be able to extend certain answers to a solution in every possible model. However, this idea alone still does not yield a satisfactory result as the following example shows.

Example 4. Consider the following query

$$\text{SELECT } ?x, ?z \text{ WHERE } (?x, \mathsf{teaches}, ?y) \text{ OPT } (?y, \mathsf{knows}, ?z)$$

over the graph $G = \{(a, \mathsf{teaches}, b), (b, \mathsf{knows}, c), (a, \mathsf{teaches}, d)\}$ and empty ontology $\mathcal{O}$. As possible models of $(G, \mathcal{O})$ we have all supergraphs of G. Hence, $\mu = \{?x \to a, ?z \to c\}$ is a certain answer and so is $\mu' = \{?x \to a\}$ ($?y$ is bound to d).

Now let $G' = \{(a, \mathsf{teaches}, b), (b, \mathsf{knows}, c)\}$. If we take as certain answers all mappings that can be extended to some solution in every possible model, then μ' is still a certain answer.

The problem in the above example is that SPARQL queries with projection and/or the UNION operator may have "subsumed" solutions, i.e., solutions such that also a proper extension is a solution. But then – with set semantics – we cannot recognize the reason why some subsumed solution (such as μ' in the above example) is possibly not a solution in some possible model: Note that in the above example, there are models for G in which μ' is not a solution, e.g., in the model $G'' = G \cup \{(d, \mathsf{knows}, c)\}$. In this model, μ is the only solution – with multiplicity 2 though.

In our first step towards reconciling SPARQL and certain answers, we decide to stick to *set semantics* in order to be able to build upon the broad DL-literature. Indeed, apart from very few exceptions (such as [18]), there hardly exist any works on query answering in DLs under *bag semantics*.

A key idea in our definition of certain answers is to only allow "maximal" solutions. For a mapping μ and some property A, we shall say that μ is $\sqsubseteq$*-maximal w.r.t.* A if μ satisfies A, and there is no μ' such that $\mu \sqsubset \mu'$ and μ' satisfies A.

Definition 1. Let $\mathcal{G} = (G, \mathcal{O})$ be a KB and $(\mathcal{T}, \mathcal{X})$ a pwdPT. A mapping μ is a *certain answer* to $(\mathcal{T}, \mathcal{X})$ over $\mathcal{G}$ if it is a $\sqsubseteq$-maximal mapping such that (1) $\mu \sqsubseteq [\![(\mathcal{T}, \mathcal{X})]\!]_{G'}$ for every model G' of $\mathcal{G}$, and (2) $\mathsf{vars}(\mathcal{T}') \cap \mathcal{X} = \mathsf{dom}(\mu)$ for some subtree $\mathcal{T}'$ of $\mathcal{T}$. We denote by $\mathsf{cert}(\mathcal{T}, \mathcal{X}, \mathcal{G})$ the set of all certain answers to $(\mathcal{T}, \mathcal{X})$ over $\mathcal{G}$.

Hence, in Example 2, only μ is a certain answer over graph G, while the subsumed solution μ' is dismissed.

The above definition ensures an important property of solutions to pwdPTs, namely: the domain of every solution must correspond to the free variables of some subtree in the pwdPT. However, for the design of algorithms to actually compute the certain answers for some entailment regime, it will turn out convenient if, in the first place, we do not need to check this condition. This leads to the following definition of *certain pre-answers* for pwdPT.

Definition 2. Let $\mathcal{G} = (G, \mathcal{O})$ be a KB and $(\mathcal{T}, \mathcal{X})$ a pwdPT. A mapping μ is a *certain pre-answer* to $(\mathcal{T}, \mathcal{X})$ over $\mathcal{G}$ if μ is a $\sqsubseteq$-maximal mapping such that $\mu \sqsubseteq [\![(\mathcal{T}, \mathcal{X})]\!]_{G'}$ for every model G' of $\mathcal{G}$. We denote by $\mathsf{certp}(\mathcal{T}, \mathcal{X}, \mathcal{G})$ the set of all certain pre-answers to $(\mathcal{T}, \mathcal{X})$ over $\mathcal{G}$.

We note that models of a KB $\mathcal{G} = (G, \mathcal{O})$ are invariant under renaming of URIs that do not occur in G. Hence, certain pre-answers and certain answers have mappings only to constants in $\mathsf{dom}(G)$. The following example shows the difference between them. In Theorem 1 below, we shall then establish their precise relationship.

Example 5. Consider the following query

$$\text{SELECT } ?x, ?y, ?z \text{ WHERE } (?x, \mathsf{a}, \mathsf{Prof}) \text{ OPT } ((?x, \mathsf{teaches}, ?y) \text{ AND } (?x, \mathsf{knows}, ?z))$$

and the following KB: $G = \{(a, \mathsf{a}, \mathsf{Prof}), (a, \mathsf{knows}, c)\}$ and $\mathcal{O} = \{(\mathsf{Prof}, \mathtt{rdfs:sc}, \exists \mathsf{teaches})\}$. The mapping $\mu = \{?x \to a, ?z \to c\}$ is the certain pre-answer, whereas $\mu' = \{?x \to a\}$ is the certain answer.

THEOREM 1. *Let $\mathcal{G}$ be a KB and $(\mathcal{T}, \mathcal{X})$ a pwdPT. Then $\mu \in \mathsf{cert}(\mathcal{T}, \mathcal{X}, \mathcal{G})$ iff μ is a $\sqsubseteq$-maximal mapping such that:*

(1) $\mu \sqsubseteq \mathsf{certp}(\mathcal{T}, \mathcal{X}, \mathcal{G})$, and

(2) $\mathsf{vars}(\mathcal{T}') \cap \mathcal{X} = \mathsf{dom}(\mu)$ for some subtree $\mathcal{T}'$ of $\mathcal{T}$.

PROOF SKETCH. '$\Rightarrow$' Let $\mu \in \mathsf{cert}(\mathcal{T}, \mathcal{X}, \mathcal{G})$. Since, $\mu \sqsubseteq [\![(\mathcal{T}, \mathcal{X})]\!]_{G'}$ for every graph $G' \models \mathcal{G}$, $\mu \sqsubseteq \mathsf{certp}(\mathcal{T}, \mathcal{X}, \mathcal{G})$. The mapping μ is $\sqsubseteq$-maximal trivially from Definition 1. '$\Leftarrow$' Let μ be $\sqsubseteq$-maximal and satisfying (1) and (2) We prove Definition 1 condition (1): Since $\mu \sqsubseteq \mathsf{certp}(\mathcal{T}, \mathcal{X}, \mathcal{G})$, $\mu \sqsubseteq [\![(\mathcal{T}, \mathcal{X})]\!]_{G'}$ for every $G' \models \mathcal{G}$; condition (2): Follows from the $\sqsubseteq$-maximality of μ w.r.t. to (1) and (2). $\square$

By the above theorem, the set of certain answers of some pwdPT can be easily computed from the set of certain pre-answers by retaining only those mappings that bind all free variables of some subtree. Hence, in the rest of the paper, we shall focus on certain pre-answers. The computation of the certain answers is then a simple post-processing step.

As a further simplification in our algorithms, we shall also ignore for a while the distinction between distinguished and non-distinguished variables. Indeed, it is easy to see that to compute the certain pre-answers of a pwdPT $(\mathcal{T}, \mathcal{X})$, it suffices to first compute the certain pre-answers of $\mathcal{T}$ (without projection, i.e. $\mathcal{X} = \mathsf{vars}(\mathcal{T})$) and then to project over $\mathcal{X}$ as shown in the following proposition.

PROPOSITION 1. *Let $\mathcal{G}$ be a KB and $(\mathcal{T}, \mathcal{X})$ a pwdPT. Then $\mu \in \mathsf{certp}(\mathcal{T}, \mathcal{X}, \mathcal{G})$ iff μ is a $\sqsubseteq$-maximal mapping s.t. (1) $\mu \sqsubseteq \mathsf{certp}(\mathcal{T}, \mathcal{G})$, and (2) $\mathsf{dom}(\mu) \subseteq \mathcal{X}$.*

Thus, from now on, we consider only certain *pre-answers* for well-designed pattern trees *without projection*.

In many Description Logics, the *canonical model* provides the basis for actually computing the certain answers for CQs or unions of CQs. Since our ontology is a variant of the DL *DL-Lite*$_\mathcal{R}$, we will also employ canonical models, which we define similarly to [5]. The canonical model of a consistent KB $\mathcal{G} = (G, \mathcal{O})$ is a graph G' that can be homomorphically mapped (preserving the constants occurring G) into any other model G'' of $\mathcal{G}$. From now on, we assume $\mathcal{G}$ is consistent.

In addition to the URIs in $\mathbf{C}$, we introduce a new set of URIs $\mathbf{A}_\mathcal{G} \subseteq \mathbf{U}$ with $\mathbf{A}_\mathcal{G} \cap \mathbf{C} = \emptyset$. The URIs in $\mathbf{A}_\mathcal{G}$ have the following form: $aQ_1Q_2 \dots Q_n$, where $n \geq 1$, $a \in \mathsf{dom}(G)$ and $Q_1, \dots, Q_n$ are basic roles. The set $\mathbf{A}_\mathcal{G}$ consists of those URIs which satisfy (1) $\mathcal{G} \models (a, \mathsf{a}, \exists Q_1)$; (2) for every $i \in \{1, \dots, n-1\}$, $\mathcal{O} \models (\exists Q_i^-, \mathtt{rdfs:sc}, \exists Q_{i+1})$ and $Q_i^- \neq Q_{i+1}$. We denote by $\mathsf{tail}(u)$ the last role in a URI $u \in \mathbf{A}_\mathcal{G}$, and define $\mathsf{can}_\mathcal{O}(G)$ for basic roles Q, Q' and a basic concept B:

$$G \cup \tag{1}$$

$$\{(a, \mathsf{a}, B) \mid \mathcal{G} \models (a, \mathsf{a}, B)\} \cup \tag{2}$$

$$\{(a, Q, b) \mid \mathcal{G} \models (a, Q, b)\} \cup \tag{3}$$

$$\{(u, \mathsf{a}, B) \mid u \in \mathbf{A}_\mathcal{G}, \mathcal{O} \models (\exists \mathsf{tail}(u)^-, \mathtt{rdfs:sc}, B)\} \cup \tag{4}$$

$$\{(u_1, Q, u_2) \mid u_2 \in \mathbf{A}_\mathcal{G}, u_2 = u_1Q', \mathcal{O} \models (Q', \mathtt{rdfs:sp}, Q)\} \cup \tag{5}$$

$$\{(u_2, Q, u_1) \mid u_2 \in \mathbf{A}_\mathcal{G}, u_2 = u_1Q', \mathcal{O} \models (Q', \mathtt{rdfs:sp}, Q^-)\} \tag{6}$$

We denote by $\mathsf{comp}_\mathcal{O}(G)$ the triples obtained from the union of Equations (1)-(3). Notice that $\mathsf{comp}_\mathcal{O}(G) = \mathsf{can}_{\mathcal{O}_{RDFS}}(G)$, where $\mathcal{O}_{RDFS}$ is the set of all RDFS inclusions σ, s.t. $\mathcal{O} \models \sigma$ (for a similar result see [9] or [27]). We will write $\mathsf{can}(G)$ (resp., $\mathsf{comp}(G)$) instead of $\mathsf{can}_\mathcal{O}(G)$ (resp., $\mathsf{comp}_\mathcal{O}(G)$) if $\mathcal{O}$ is clear from the context. Note that $\mathsf{can}(G)$ can be infinite in size. The following well-known result from databases and DLs can be shown for $\mathsf{can}(G)$ (see e.g. the proof of Theorem 2 in [24]).

PROPOSITION 2. *Let $\mathcal{G} = (G, \mathcal{O})$ be a KB. Then*

(a) $\mathsf{can}(G) \models \mathcal{G}$, and

(b) for any G', s.t. $G' \models \mathcal{G}$, there is a homomorphism h from $\mathsf{can}(G)$ to G' s.t. for every $a \in \mathsf{dom}(G)$, $\mathsf{h}(a) = a$.

In Theorem 2, we shall establish the relationship between certain pre-answers to wdPTs and the canonical model. To this end, we first introduce the following useful definition.

Definition 3. For a mapping μ, we let $\mu\!\downarrow\!= \mu|_{\{?x \mid \mu(?x) \in \mathbf{C}\}}$, i.e. $\mu\!\downarrow$ is the restriction of μ to mappings into constants. Assume a set M of mappings. Then $M\!\downarrow\!= \{\mu\!\downarrow \mid \mu \in M\}$.

Moreover, $\mathrm{MAX}(M) = \{\mu \in M \mid \nexists \mu' \in M \text{ s.t. } \mu \sqsubset \mu'\}$, i.e. $\mathrm{MAX}(M)$ is the set of $\sqsubseteq$-maximal mappings in M.

THEOREM 2. *Let $\mathcal{G} = (G, \mathcal{O})$ be a KB and $\mathcal{T}$ a wdPT. Then, $\mathsf{certp}(\mathcal{T}, \mathcal{G}) = \mathrm{MAX}([\![\mathcal{T}]\!]_{\mathsf{can}(G)}\!\downarrow)$.*

PROOF SKETCH. It suffices to prove $\mathsf{certp}(\mathcal{T}, \mathcal{G}) \sqsubseteq [\![\mathcal{T}]\!]_{\mathsf{can}(G)}\!\downarrow$, and $\mathsf{certp}(\mathcal{T}, \mathcal{G}) \sqsupseteq [\![\mathcal{T}]\!]_{\mathsf{can}(G)}\!\downarrow$. For '$\sqsubseteq$' let μ be a mapping in $\mathsf{certp}(\mathcal{T}, \mathcal{G})$. Then, μ is subsumed by an answer of $\mathcal{T}$ in every model. Hence, also by an answer μ' in $\mathsf{can}(G)$. Since, μ only maps variables to constants in $\mathsf{dom}(G)$, μ is still subsumed by $\mu'\!\downarrow$. For '$\sqsupseteq$' let μ be a mapping in $[\![\mathcal{T}]\!]_{\mathsf{can}(G)}\!\downarrow$. By Proposition 2, μ can be homomorphically mapped into every model of $\mathcal{G}$, hence μ is part of a solution in every model, i.e. also maximal solutions. □

4. MODULAR QUERY REWRITING

In this section we provide our first method to compute certain answers to well-designed pattern trees under OWL 2 QL entailment. In particular, we present a *query rewriting* procedure to compute $\mathsf{certp}(\mathcal{T}, \mathcal{G})$ for a given wdPT $\mathcal{T}$ and a KB $\mathcal{G} = (G, \mathcal{O})$. To this end, we extend the well-known query reformulation algorithm of Calvanese et al. for answering *conjunctive queries (CQs)* over DL knowledge bases $K = (T, A)$, where T is a DL-Lite TBox and A is an ABox [7]. The algorithm of Calvanese et al. transforms a CQ q and a TBox T into a union Q of CQs such that, given any ABox A, computing the certain answers to q over $K = (T, A)$ is equivalent to evaluating Q over A, where A is seen as a plain relational database.

In the presented approach, given a wdPT $\mathcal{T}$ and an ontology $\mathcal{O}$ we apply a rewriting at each node of $\mathcal{T}$, obtaining a tree $\mathcal{T}'$ of sets of BGPs. Roughly speaking, given any RDF graph G, computing $\mathsf{certp}(\mathcal{T}, \mathcal{G})$ then corresponds to evaluating $\mathcal{T}'$ in a bottom-up fashion over G. Since nodes of $\mathcal{T}$ are rewritten in isolation but may share variables, the rewriting involves some bookkeeping, which we describe next. In particular, $\mathcal{T}'$ will be expressed as an *enriched query*.

Definition 4. Let Paths be the set of all words of the form $?xQ_1 \dots Q_n$, where $Q_1, \dots, Q_n$ are basic roles with $n \geq 0$. Given a set of variables $L \subseteq \mathbf{V}$, an *L-guide* is a partial function γ from variables to words in Paths such that (a) $\mathsf{dom}(\gamma) \cap L = \emptyset$, and (b) for any $?x \in \mathsf{dom}(\gamma)$ with $\gamma(?x) = ?zQ_1 \dots Q_n$ we have that $?z \in L$. An *enriched BGP* is any tuple $\langle P, \gamma \rangle$, where P is a BGP and γ is a $\mathsf{vars}(P)$-guide. Expressions constructed from enriched BGPs by applying the AND, OPT and UNION connectives are called *enriched queries*. We use $E, E_1, E_2, \dots$ to denote enriched queries.

We are ready to define the standard semantics of enriched queries. Intuitively, the answer to an enriched BGP $\langle P, \gamma \rangle$ over a graph G is obtained by first evaluating P using the standard semantics and then extending the obtained mappings with additional assignments prescribed by γ, potentially introducing assignments to elements from $\mathbf{A}_\mathcal{G}$. The semantics is then generalized to full enriched queries using the standard operations on mappings.

Definition 5. Assume a mapping μ and a $\mathsf{dom}(\mu)$-guide γ. We let $\mathsf{expand}(\mu, \gamma) = \mu \cup \{?x \to \mu(?y)Q_1 \dots Q_n \mid \gamma(?x) = ?yQ_1 \dots Q_n\}$. Assume an RDF graph G. Then the evaluation of enriched queries is defined inductively as follows:

no	Triple t	Inclusion α	$\mathsf{gr}(t,\alpha)$
r1	$(?x, \mathsf{a}, A)$	$(A_1, \texttt{rdfs:sc}, A)$	$\langle (?x, \mathsf{a}, A_1), \emptyset \rangle$
r2	$(?x, \mathsf{a}, A)$	$(\exists Q, \texttt{rdfs:sc}, A)$	$\langle (?x, Q, ?x'), \emptyset \rangle$
r3	$(?x, Q, ?y)$	$(A, \texttt{rdfs:sc}, \exists Q)$	$\langle (?x, \mathsf{a}, A), ?y \rightarrow ?x Q \rangle$
r4	$(?x, Q, ?y)$	$(\exists Q_1, \texttt{rdfs:sc}, \exists Q)$	$\langle (?x, Q_1, ?x'), ?y \rightarrow ?x Q \rangle$
r5	$(?x, Q, ?y)$	$(Q', \texttt{rdfs:sp}, Q)$	$\langle (?x, Q', ?y), \emptyset \rangle$

Table 1: The result $\mathsf{gr}(t,\alpha)$ of applying an inclusion α to a triple t. Here $?x'$ denotes a fresh variable.

- $[\![E]\!]_G = \{\mathsf{expand}(\mu,\gamma) \mid \mu \in [\![P]\!]_G\}$ in case $E = (P,\gamma)$ is an enriched BGP.
- $[\![E_1 \text{ AND } E_2]\!]_G = \{\mu_1 \cup \mu_2 \mid \mu_1 \in [\![E_1]\!]_G, \mu_2 \in [\![E_2]\!]_G, \text{ and } \mu_1 \sim \mu_2\}$.
- $[\![E_1 \text{ OPT } E_2]\!]_G = [\![E_1 \text{ AND } E_2]\!]_G \cup \{\mu_1 \in [\![E_1]\!]_G \mid \forall \mu_2 \in [\![E_2]\!]_G : \mu_1 \nsim \mu_2\}$.
- $[\![E_1 \text{ UNION } E_2]\!]_G = [\![E_1]\!]_G \cup [\![E_2]\!]_G$.

Our rewriting procedure relies on the following "lifting" property of OWL 2 QL. For a guide γ, let $||\gamma||$ denote the maximum $|w|$ over all $w \in \mathsf{rng}(\gamma)$. Given a wdPT $\mathcal{T}$ and an ontology $\mathcal{O}$, we let $\mathsf{d}(\mathcal{T},\mathcal{O}) = 2k_1 + k_2$, where k_1 is the number of atomic roles occurring in $\mathcal{T}$ and $\mathcal{O}$, and k_2 is the number of triples in $\mathcal{T}$.

PROPOSITION 3. *Assume a wdPT $\mathcal{T}$, a KB $\mathcal{G} = (G,\mathcal{O})$, and a mapping $\mu \in [\![\mathcal{T}]\!]_{\mathsf{can}(G)}$. Then there exists a mapping $\mu' \in [\![\mathcal{T}]\!]_{\mathsf{can}(G)}$ such that (a) $\mu{\downarrow} = \mu'{\downarrow}$, and (b) $|w| \leq \mathsf{d}(\mathcal{T},\mathcal{O})$ for every $w \in \mathsf{rng}(\mu')$.*

Intuitively, the above says that any mapping μ for a wdPT $\mathcal{T}$ in the canonical model of $\mathcal{G}$ can be transformed into a mapping μ' that is confined to the first $\mathsf{d}(\mathcal{T},\mathcal{O})$ levels of the canonical model yet agrees with μ on the assignments to constants. The property follows from the well known fact that in each branch of the canonical model of $\mathcal{G}$ one finds only a small number of *types* (see e.g. [4]). We note that the above property does not hold in the presence of full OWL 2.

With the semantics and the desired properties of enriched queries established, we can describe the rewriting algorithm. The procedure exhaustively rewrites triples of an enriched BGP by applying inclusions "backwards". Assume a BGP P. We say an inclusion α is *applicable* to a triple t w.r.t. P if t and α have the following forms:

(i) $t = (?x, \mathsf{a}, A)$ and $\alpha = (B, \texttt{rdfs:sc}, A)$;

(ii) $t = (?x, Q, ?y)$, where $?y$ does not appear in any other triple of P, and α has $\exists Q$ in the third position;

(iii) $t = (?x, Q, ?y)$, $?x$ does not appear in any other triple of P, and α has $\exists Q^-$ in the third position;

(iv) $t = (?x, Q, ?y)$ and $\alpha = (Q', \texttt{rdfs:sp}, Q)$.

Let $\mathsf{App}(P)$ denote the binary relation such that $(t,\alpha) \in \mathsf{App}(P)$ iff an inclusion α is applicable to a triple t w.r.t. P.

In Algorithm 1 we present the rewriting routine RefBGP, which reformulates an input BGP P_0 by taking into account the inclusions of $\mathcal{O}$. It returns pr, which is a union of enriched BGPs. The procedure is parametrized by an integer k, which restricts the length of guide ranges and thus guarantees termination. As we shall see, Proposition 3 ensures that we do not lose completeness. The procedure employs the function $\mathsf{gr}(t,\alpha)$, which returns a tuple $\langle t', \gamma \rangle$, where t' is a triple and γ is a singleton guide, both obtained from t by applying α as described by Table 1. $P[t/t']$ denotes the BGP obtained from P by replacing the triple t with t'. Given a BGP P and a substitution σ, we let $\sigma(P) = \{\sigma(t) \mid t \in P\}$. A *unifier* of a pair t_1, t_2 of triples is a substitution σ such that $\sigma(t_1) = \sigma(t_2)$. We call σ a *most general unifier (MGU)* of t_1, t_2 if for any unifier σ' of t_1, t_2 there exists a unifier σ'' of t_1, t_2 such that $\sigma' = \sigma \circ \sigma''$.

Algorithm 1: RefBGP

input : a BGP P_0, an inclusion set $\mathcal{O}$, and an integer k
output : a set pr of enriched BGPs

```
pr ← {⟨P_0, ∅⟩}
repeat
    pr' ← pr
    foreach ⟨P, γ⟩ ∈ pr' s.t. ||γ|| ≤ k do
        foreach t ∈ P and α ∈ O s.t. (t, α) ∈ App(P) do
            ⟨t', γ'⟩ ← gr(t, α)
            pr ← pr ∪ {⟨P[t/t'], expand(γ', γ)⟩}
        foreach pair t_1, t_2 in P with a MGU σ do
            pr ← pr ∪ {⟨σ(P), expand(σ, γ)⟩}
until pr' = pr
return pr
```

Using an adaptation of the proof in [7], we can show that evaluating the output of RefBGP over a plain RDF graph corresponds to evaluating the original query over an initial segment of the canonical model. More formally:

PROPOSITION 4. *Let P be a BGP, $\mathcal{G} = (G,\mathcal{O})$ a KB, k an integer and pr the union of enriched BGPs returned by $\mathsf{RefBGP}(P,\mathcal{O},k)$. Then $([\![pr]\!]_G)|_{\mathsf{vars}(\mathcal{T})} = [\![P]\!]_{G'}$, where G' is the restriction of $\mathsf{can}(G)$ to URIs $a \in \mathbf{U}$ such that $|a| \leq k$.*

We can now generalize the rewriting from BGPs to full wdPTs. Given a wdPT $\mathcal{T}$ and an ontology $\mathcal{O}$, we simply apply RefBGP to the BGP of every node in $\mathcal{T}$.

Definition 6. Let $\mathcal{T}$ be a wdPT and $\mathcal{O}$ an ontology. Let $\mathsf{RefPT}(\mathcal{T},\mathcal{O})$ denote the enriched query obtained by replacing in $\mathcal{T}$ every BGP P by $\mathsf{RefBGP}(P,\mathcal{O},\mathsf{d}(\mathcal{T},\mathcal{O}))$.

We arrive at the main technical result of this section.

THEOREM 3. *Let $\mathcal{T}$ be a wdPT and $\mathcal{G} = (G,\mathcal{O})$ be a KB. Let $M = ([\![\mathsf{RefPT}(\mathcal{T},\mathcal{O})]\!]_G)|_{\mathsf{vars}(\mathcal{T})}$. Then*

$$\mathsf{certp}(\mathcal{T},(G,\mathcal{O})) = \mathrm{MAX}(M{\downarrow}).$$

PROOF (SKETCH). First note that by using Proposition 4 and induction on the structure of $\mathcal{T}$, one can show ($\star$) $M = [\![\mathcal{T}]\!]_{G'}$, where G' is the restriction of $\mathsf{can}(G)$ to URIs $a \in \mathbf{U}$ such that $|a| \leq \mathsf{d}(\mathcal{T},\mathcal{O})$.

Due to Theorem 2, to prove the present theorem it suffices to show that $([\![\mathcal{T}]\!]_{\mathsf{can}(G)}){\downarrow} = M{\downarrow}$.

Take an arbitrary mapping $\mu^* \in ([\![\mathcal{T}]\!]_{\mathsf{can}(G)}){\downarrow}$. Then there exists a mapping $\mu \in ([\![\mathcal{T}]\!]_{\mathsf{can}(G)})$ such that $\mu{\downarrow} = \mu^*{\downarrow}$. Moreover, due to Proposition 3 we can w.l.o.g. assume that $|w| \leq \mathsf{d}(\mathcal{T},\mathcal{O})$ for every $w \in \mathsf{rng}(\mu)$. Then due to ($\star$), $\mu' \in M$. Thus also $\mu^* \in M{\downarrow}$.

Take an arbitrary mapping $\mu^* \in M{\downarrow}$. Then there exists $\mu \in M$ such that $\mu{\downarrow} = \mu^*{\downarrow}$. Then due to ($\star$), $\mu \in [\![\mathcal{T}]\!]_{G'}$. Thus trivially $\mu \in [\![\mathcal{T}]\!]_{\mathsf{can}(G)}$ and $\mu^* \in ([\![\mathcal{T}]\!]_{\mathsf{can}(G)}){\downarrow}$. □

We put our algorithm to work in the following example.

Example 6. Let $\mathcal{O}$ be the following set of inclusions:

$$(\mathsf{Person}, \texttt{rdfs:sc}, \exists\mathsf{knows}), \tag{7}$$

$$(\mathsf{Prof}, \texttt{rdfs:sc}, \exists\mathsf{teaches}), \tag{8}$$

which states that every Person object knows somebody and that every Prof object teaches somebody. Let G be an RDF graph containing the following triples: $(a, \mathsf{a}, \mathsf{Person}), (b, \mathsf{a}, \mathsf{Prof})$. We want to derive answers for the following query $\mathcal{T}$ over the above knowledge base:

$$(?x, \mathsf{teaches}, ?y) \text{ OPT } \big((?z, \mathsf{teaches}, ?y) \text{ OPT } (?u, \mathsf{knows}, ?y)\big)$$

This query asks for all $?x, ?z, ?u$ s.t. $?x$ teaches some $?y$ and optionally all $?z$, s.t. $?z$ teach the same $?y$ as $?x$ and optionally all $?u$ that know $?y$. We derive new queries by rewriting each node of the pattern tree corresponding to the above query separately.

1. *We start with the first BGP* $(?x, \mathsf{teaches}, ?y)$*:* We use $t = (?x, \mathsf{teaches}, ?y)$. Equation 8 is applicable to t, hence in Line 6 $\langle t', \gamma' \rangle = \langle (?x, \mathsf{a}, \mathsf{Prof}), ?y \rightarrow ?x \cdot \mathsf{teaches} \rangle$, which will be the enriched query that we add to pr in Line 7. Since our BGP consists of a single triple pattern, we will not unify any pairs of triple patterns in Line 8 and 9. No further application of inclusions leads to a new query. Hence, RefBGP$((?x, \mathsf{teaches}, ?y), \mathcal{O}, \mathsf{d}(\mathcal{T}, \mathcal{O}))$ outputs:

$$\langle (?x, \mathsf{teaches}, ?y), \emptyset \rangle \tag{9}$$

$$\langle (?x, \mathsf{a}, \mathsf{Prof}), ?y \rightarrow ?x \cdot \mathsf{teaches} \rangle \tag{10}$$

2. *Similar as above for the second BGP:* RefBGP$((?z, \mathsf{teaches}, ?y), \mathcal{O}, \mathsf{d}(\mathcal{T}, \mathcal{O}))$ outputs:

$$\langle (?z, \mathsf{teaches}, ?y), \emptyset \rangle \tag{11}$$

$$\langle (?z, \mathsf{a}, \mathsf{Prof}), ?y \rightarrow ?z \cdot \mathsf{teaches} \rangle \tag{12}$$

3. *And similar as above for the third BGP:* RefBGP$((?u, \mathsf{knows}, ?y), \mathcal{O}, \mathsf{d}(\mathcal{T}, \mathcal{O}))$ outputs:

$$\langle (?u, \mathsf{knows}, ?y), \emptyset \rangle \tag{13}$$

$$\langle (?u, \mathsf{a}, \mathsf{Person}), ?y \rightarrow ?u \cdot \mathsf{knows} \rangle \tag{14}$$

Evaluating each of the enriched queries (9)-(14) over G gives the following mappings: Query (10) evaluated over G outputs $\{?x \rightarrow b, ?y \rightarrow b \cdot \mathsf{teaches}\}$, Query (12) $\{?z \rightarrow b, ?y \rightarrow b \cdot \mathsf{teaches}\}$ and Query (14) $\{?u \rightarrow a, ?y \rightarrow a \cdot \mathsf{knows}\}$. Combining the answers via Definition 5 gives the mapping: $\{?x \rightarrow b, ?z \rightarrow b\}$.

5. HOLISTIC QUERY REWRITING

The modular approach from the previous section follows the general philosophy of SPARQL entailment regimes from [12] which also proceeds by first treating each BGP individually. However, a possible disadvantage of our modular approach is that we need to maintain additional data structures (in particular, the "guides") to ensure consistency when combining the partial solutions from different nodes in the pattern tree to an overall solution. As a consequence, our whole algorithm has to be implemented from scratch because standard tools cannot handle these additional data structure. In this section, we therefore present a second approach to SPARQL evaluation under OWL 2 QL entailment. Its main goal is to make use of standard technology as far as possible. We thus aim at a transformation of OWL 2 QL entailment under our novel certain answer semantics into SPARQL evaluation with respect to RDFS Entailment. For the latter task, strong tools (e.g. [13, 22]) are available. For the actual rewriting, we shall follow the rewriting from [9] (see also [28] for a similar approach) which incorporates several improvements compared with the original algorithm from [7] used in the previous section. The most important conceptual difference between our two algorithms, however, is that the algorithm presented below proceeds in a holistic way, i.e., our query rewriting always takes the entire pattern tree into account. Further differences between our modular and holistic algorithm will be discussed in Section 6.

As the modular rewriting, the holistic rewriting may unify and thus eliminate original variables; this needs to be memorized in order to construct correct answers in the end. Thus the rewriting operates on pairs $\langle \mathcal{T}, \beta \rangle$, where $\mathcal{T}$ is a wdPT and β a partial function from variables to sets of variables. Intuitively, $?x \in \beta(?u)$ says that $?x$ was substituted by the variable $?u$ during the rewriting. The evaluation of a pair $\langle \mathcal{T}, \beta \rangle$ over an RDF graph is defined as follows:

$$[\![\langle \mathcal{T}, \beta \rangle]\!]_G = \{\mathsf{expand}(\mu, \beta) \mid \mu \in [\![\mathcal{T}]\!]_G\}, \text{ where}$$

$$\mathsf{expand}(\mu, \beta) = \{?z \rightarrow \mu(?u) \mid ?z \in \beta(?u), ?u \in \mathsf{dom}(\mu)\}$$

The rewriting of $\langle \mathcal{T}, \beta \rangle$ is obtained by applying exhaustively the following procedure.

Definition 7. For a wdPT $\mathcal{T}$ together with a function β and an ontology $\mathcal{O}$, we write $\langle \mathcal{T}, \beta \rangle \rightarrow_{\mathcal{O}} \langle \mathcal{T}', \beta' \rangle$ if $\mathcal{T}'$ and β' can be obtained from $\mathcal{T}$ and β by the following steps:

(S1) non-deterministically pick a variable $?x \in \mathsf{vars}(\mathcal{T})$ and a role Q.

(S2) pick a concept B_Q, s.t. $\mathcal{O} \models (B_Q, \texttt{rdfs:sc}, \exists Q)$ and Q is a basic role. If no such concept B_Q exists, continue with (S1).

(S3) Set $\langle \mathcal{T}', \beta' \rangle = \langle \mathcal{T}, \beta \rangle$. Drop from $\mathcal{T}'$ every subtree whose root has an triple of the form

- (a) $(?z, Q_1, ?x)$ such that $\mathcal{O} \not\models (Q, \texttt{rdfs:sp}, Q_1)$, or
- (b) $(?x, Q_1, ?z)$ such that $\mathcal{O} \not\models (Q, \texttt{rdfs:sp}, Q_1^-)$, or
- (c) $(?x, \mathsf{a}, A)$ such that $\mathcal{O} \not\models (\exists Q^-, \texttt{rdfs:sc}, A)$.

Note that $?z$ is an arbitrary variable.

(S4) Let $\mathsf{neighbours}_{\mathcal{T}'}(?x) = \{?z \mid (?x, P, ?z) \in \mathsf{pat}(\mathcal{T}') \text{ or } (?z, P, ?x) \in \mathsf{pat}(\mathcal{T}')\}$. Take a fresh variable $?u$ and replace in $\mathcal{T}'$ all $?z \in \mathsf{neighbours}_{\mathcal{T}'}(?x)$ with $?u$. In other words, collapse all neighbours of $?x$ into $?u$. Moreover, let $\beta'(?u) = \bigcup_{?z \in \mathsf{neighbours}_{\mathcal{T}'}(?x)} \beta(?z)$.

(S5) In all nodes n of $\mathcal{T}'$ where $?x \in \mathsf{vars}(n)$:

- if B_Q is an atomic concept: add $(?u, \mathsf{a}, B_Q)$.
- if B_Q is of the form $\exists Q'$: add $(?u, Q', ?y)$, where $?y$ is a new variable, i.e. not yet occurring in $\mathcal{T}'$.

(S6) In $\mathcal{T}'$, drop every triple t where $?x \in \mathsf{vars}(t)$.

We write $\langle \mathcal{T}, \beta \rangle \rightarrow^*_{\mathcal{O}} \langle \mathcal{T}', \beta' \rangle$ if $\langle \mathcal{T}', \beta' \rangle$ can be obtained from $\langle \mathcal{T}, \beta \rangle$ by finitely many rewrite iterations. We let $\mathsf{rew}_{\mathcal{O}}(\mathcal{T}, \beta) = \{\langle \mathcal{T}', \beta' \rangle \mid \langle \mathcal{T}, \beta \rangle \rightarrow^*_{\mathcal{O}} \langle \mathcal{T}', \beta' \rangle\}$.

Example 7. Let us revisit the KB and the query of Example 6. Let $\mathcal{T}$ denote the wdPT corresponding to this, and let $\beta_{\mathcal{T}}$ be defined as in Theorem 4. Then, the set $\mathsf{rew}_{\mathcal{O}}(\mathcal{T}, \beta_{\mathcal{T}})$ is obtained as follows. We start with the query $\langle \mathcal{T}, \beta_T \rangle$ and proceed as in Definition 7. The steps that change the PT are depicted in Figure 1.

(S1) Let us pick variable $?y$ and the role teaches.

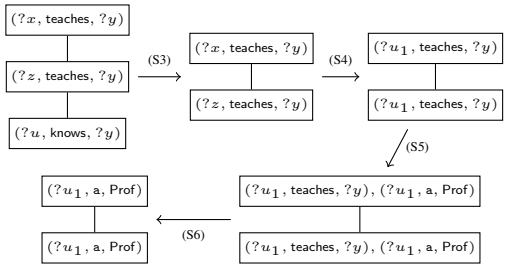


Figure 1: Holistic rewriting given in Example 7.

(S2) Since $\mathcal{O} \models (\mathsf{Prof}, \mathtt{rdfs:sc}, \exists\mathsf{teaches})$, we pick the concept Prof.

(S3) $\mathcal{T}'$ contains a node with $(?u, \mathsf{knows}, ?y)$, since $\mathcal{O} \not\models (\mathsf{teaches}, \mathtt{rdfs:sp}, \mathsf{knows}^-)$, we drop from $\mathcal{T}'$ the subtree rooted at the node containing $(?u, \mathsf{knows}, ?y)$.

(S4) Now, $\mathsf{neighbours}_{\mathcal{T}'}(?y) = \{?x, ?z\}$. We introduce a fresh variable $?u_1$ and replace all occurrences of $?x$ and $?z$ in $\mathcal{T}'$ with $?u_1$. In addition, we set $\beta'(?u_1) = \beta(?x) \cup \beta(?z) = \{?x, ?z\}$.

(S5) In all nodes of $\mathcal{T}'$ with an occurrence of $?y$ we add $(?u_1, \mathsf{a}, \mathsf{Prof})$.

(S6) We now drop all triples where $?y$ occurs.

After this step, we will have the pattern tree depicted in the lower left of Figure 1 together with the function β', where $\beta'(?u_1) = \{?x, ?z\}$. Due to lack of space, we omit further rewriting steps. The evaluation of the query depicted in Figure 1 over the Graph G given in Example 6 outputs the mapping $\{?u_1 \to b\}$. Then, $\mathsf{expand}(\{?u_1 \to b\}, \beta') = \{?x \to b, ?z \to b\}$, which is the answer also obtained in Example 6.

In order to compute the certain pre-answers $\mathsf{certp}(\mathcal{T}, \mathcal{G})$, it just remains to eliminate all non-maximal mappings:

THEOREM 4. *Let $\mathcal{G} = (G, \mathcal{O})$ be a KB and $\mathcal{T}$ a wdPT. Let $\beta_{\mathcal{T}}$ be the function s.t. for every $?x \in \mathsf{vars}(\mathcal{T})$: $\beta_{\mathcal{T}}(?x) = \{?x\}$. Let $M = (\llbracket \mathsf{rew}_{\mathcal{O}}(\mathcal{T}, \beta_{\mathcal{T}}) \rrbracket_{\mathsf{comp}(G)})|_{\mathsf{vars}(\mathcal{T})}$. Then,*

$$\mathsf{certp}(\mathcal{T}, (G, \mathcal{O})) = \mathrm{MAX}(M).$$

PROOF IDEA. For soundness, one shows the correctness of every rewriting step $\langle \mathcal{T}, \beta \rangle \to_{\mathcal{O}} \langle \mathcal{T}', \beta' \rangle$. For this, one shows if $\mu \in \llbracket \langle \mathcal{T}', \beta' \rangle \rrbracket_{\mathsf{can}(G)}$, then there is a $\mu' \in \llbracket \langle \mathcal{T}, \beta \rangle \rrbracket_{\mathsf{can}(G)}$, s.t. $\mu \sqsubseteq \mu'$. For completeness, one shows for a mapping $\mu \in \mathsf{certp}(\mathcal{T}, \mathcal{G})$ that a sequence of rewriting steps to a query $\langle \mathcal{T}', \beta' \rangle$, s.t. $\mu \in \llbracket \langle \mathcal{T}', \beta' \rangle \rrbracket_{\mathsf{comp}(G)}$, can be found. □

6. DISCUSSION

In the previous sections, we have presented two algorithms for obtaining certain answers of well-designed SPARQL queries over OWL 2 QL ontologies. The obvious difference between the two algorithms is that the first one proceeds by rewriting each node of the pattern tree individually while the second one targets the complete pattern tree at once. We now aim at a more detailed analysis of the characteristics of each of the two algorithms. In particular, we want to identify typical settings which favour one or the other. To this end, we look at a simple ontology consisting of a single inclusion triple: $(A, \mathtt{rdfs:sc}, \exists R)$.

First, consider the following pattern tree with 2 nodes:

$$\boxed{(?x, P, ?y_1), \ldots, (?x, P, ?y_n)}$$
$$\boxed{(?z_1, R, ?y_1), \ldots, (?z_n, R, ?y_n)}$$

- *Modular rewriting* gives a total number of queries whose order of magnitude is $O(2^n)$, since:
  a) In the *root node* we cannot apply any inclusion triple to the BGP. Hence, the algorithm just outputs one query.
  b) In the *child node* we can immediately apply the inclusion $(A, \mathtt{rdfs:sc}, \exists R)$ to any of the n triples. This can be done in any order, i.e. the total number of queries obtained is $O(2^n)$.
- *Holistic rewriting* outputs only the original query. Whatever variable $?y_i$ we pick in Step (S1), we will drop the complete query in Step (S3), since $\mathcal{O} \not\models (R, \mathtt{rdfs:sp}, P)$. Picking any other variable $?x$ or $?z_i$ does not lead to further queries either.

For this query, the holistic approach by far outperforms the modular rewriting. Observe that all $?y_i$ variables are shared by the two nodes in the wdPT. Hence, if some $?y_i$ is mapped into the anonymous part of the canonical model, we would need to infer that the R edge is also a P edge, which is of course not true. Consequently, there exists no rewriting of the entire wdPT. However, in the modular approach, each node of the wdPT is processed individually. Hence, we do not detect that it is impossible to map the variable $?y_i$ in the root node to the same anonymous element as $?y_i$ in the child node.

Second, we consider the following wdPT with n nodes:

$$\boxed{(?x_1, P, ?z), (?x_1, R, ?y_1)}$$
$$\vdots$$
$$\boxed{(?x_n, P, ?z), (?x_n, R, ?y_n)}$$

- *Modular rewriting* gives a total number of $2 * n$ queries since, in each of the n nodes, we apply $(A, \mathtt{rdfs:sc}, \exists R)$ independently.
- *Holistic rewriting* will do the same as the modular rewriting, but it copies with each rewriting also all other nodes, which then in turn must again be rewritten. Therefore, the holistic rewriting outputs a total number of queries whose order of magnitude is $O(2^n)$.

For the second query, clearly, the modular rewriting by far outperforms the holistic rewriting. We observe that now, the $?y_i$ variables which have to be picked, only appear once in the tree. Hence, the nodes are independent of each other. This is in sharp contrast to the first query, where the two nodes shared all of the $?y_i$ variables.

These two examples show that neither of the two algorithms is uniformly better than the other: depending on the setting, each of the two algorithms may produce exponentially fewer rewritings than the other. The way in which the same variables occur in multiple nodes looks like the key feature to distinguish situations where one or the other algorithm is preferable. The main strength of the holistic approach is to recognize incompatible occurrences of the same variable in different nodes, which allows the pruning of subtrees in the wdPT. The main strength of the modular approach is to avoid the same rewriting of subtrees in the wdPT for different rewritings in a parent node by treating each node individually. At any rate, it seems advantageous to have both algorithms in one's portfolio.

7. COMPUTATIONAL COMPLEXITY

Having proposed two concrete algorithms for computing the certain answers to well-designed SPARQL queries according to our

new semantics, we will next study the computational complexity of the *evaluation problem*. We show that the certain answer semantics does not increase its complexity compared to the setting without ontologies. Formally, we thus study the following problem.

OWL 2 QL-EVALUATION	
Input:	pwdPT $(\mathcal{T}, \mathcal{X})$, KB $\mathcal{G}$, mapping μ.
Question:	$\mu \in \mathsf{cert}(\mathcal{T}, \mathcal{X}, \mathcal{G})$?

For establishing several of the complexity results presented in this section, including the one on OWL 2 QL-EVALUATION, it is convenient to first consider the following problem.

PARTIAL OWL 2 QL-EVALUATION	
Input:	pwdPT $(\mathcal{T}, \mathcal{X})$, KB $\mathcal{G}$, mapping μ.
Question:	$\mu \sqsubseteq \mathsf{cert}(\mathcal{T}, \mathcal{X}, \mathcal{G})$?

THEOREM 5. *The problem* PARTIAL OWL 2 QL-EVALUATION *is* NP*-complete.*

PROOF SKETCH. The *membership* can be shown using Theorem 4: First, compute $\mathsf{comp}(G)$ in polynomial time. Next, guess a subtree $\langle \mathcal{T}', \beta' \rangle$ of a rewriting $\langle \mathcal{T}'', \beta'' \rangle \in \mathsf{rew}_{\mathcal{O}}(\mathcal{T}, \beta_{\mathcal{T}})$ (with $\mathsf{dom}(\mu) \subseteq \mathsf{vars}(\langle \mathcal{T}', \beta' \rangle)$) and an extension λ of μ s.t. $\lambda(\mathsf{pat}(\mathcal{T}')) \subseteq \mathsf{comp}(G)$.

Hardness follows immediately from the NP-completeness of the query evaluation problem for CQs. □

Deciding OWL 2 QL-EVALUATION can now be reduced to several calls of PARTIAL OWL 2 QL-EVALUATION.

THEOREM 6. *The problem* OWL 2 QL-EVALUATION *is* DP*-complete. Hardness holds even if the ontology is empty.*

PROOF IDEA. For *membership*, note that OWL 2 QL-EVALUATION $((\mathcal{T}, \mathcal{X}), (G, \mathcal{O}), \mu)$ can be solved by deciding:

1. $\mu \sqsubseteq \mathsf{cert}(\mathcal{T}, \mathcal{X}, (G, \mathcal{O}))$, and
2. for all $\mu' \colon \bar{\mathcal{X}} \to \mathsf{dom}(G)$ with $\bar{\mathcal{X}} \subseteq \mathcal{X}$ and $\mu \sqsubset \mu' \colon$ $\mu' \not\sqsubseteq \mathsf{cert}(\mathcal{T}, \mathcal{X}, (G, \mathcal{O}))$?

Step (1) is in NP as shown in Theorem 5. For step (2), the co-problem is in NP: It can be solved by first guessing a mapping μ' (of polynomial size) and then checking if $\mu' \sqsubseteq \mathsf{cert}(\mathcal{T}, \mathcal{X}, (G, \mathcal{O}))$. Guessing μ' and the witness for $\mu' \sqsubseteq \mathsf{cert}(\mathcal{T}, \mathcal{X}, (G, \mathcal{O}))$ can be done in one step. Thus step (2) is in coNP.

The *hardness* is shown in the full version. □

We continue by studying typical query analysis tasks, i.e., subsumption ($\sqsubseteq$), containment ($\subseteq$), and equivalence ($\equiv$) between pwdPTs. First of all, we have to adapt these notions to our setting including ontologies. That is, let $(\mathcal{T}_1, \mathcal{X}_1)$ and $(\mathcal{T}_2, \mathcal{X}_2)$ be pwdPTs, and $\mathcal{O}$ be an ontology. We say that $(\mathcal{T}_1, \mathcal{X}_1) \circ (\mathcal{T}_2 \mathcal{X}_2)$ (for $\circ \in \langle \sqsubseteq_{\mathcal{O}}, \subseteq_{\mathcal{O}}, \equiv_{\mathcal{O}} \rangle$) if $\mathsf{cert}(\mathcal{T}_1, \mathcal{X}_1, (G, \mathcal{O})) \circ' \mathsf{cert}(\mathcal{T}_2, \mathcal{X}_2, (G, \mathcal{O}))$ (for $\circ' \in \langle \sqsubseteq, \subseteq, = \rangle$, respectively). This allows us to define the query analysis problems studied in this section.

OWL 2 QL-SUBSUMPTION/OWL 2 QL-CONTAINMENT/ OWL 2 QL-EQUIVALENCE	
Input:	pwdPTs $(\mathcal{T}_1, \mathcal{X}_1)$, $(\mathcal{T}_2, \mathcal{X}_2)$, ontology $\mathcal{O}$.
Question:	$(\mathcal{T}_1, \mathcal{X}_1) \sqsubseteq_{\mathcal{O}}$ / $\subseteq_{\mathcal{O}}$ / $\equiv_{\mathcal{O}}$ $(\mathcal{T}_2, \mathcal{X}_2)$?

These problems can be decided by using (PARTIAL) OWL 2 QL-EVALUATION. For characterizing the relationships between these problems, we have to introduce some additional notation first. Let $(\mathcal{T}, \mathcal{X})$ be a pwdPT. We use $\mathsf{fvars}(\mathcal{T}) = \mathsf{vars}(\mathcal{T}) \cap \mathcal{X}$ to denote the set of free variables in $\mathcal{T}$. Moreover, let $\mathsf{db}(.)$ be a bijective function that assigns to each variable $?x \in \mathsf{vars}(\mathcal{T})$ a unique, new URI $\mathsf{db}(?x)$, and that maps constants onto themselves. Then the *frozen RDF graph* G for $\mathcal{T}$ is the set of triples $G = \{\mathsf{db}(t) \mid t \in \mathsf{pat}(\mathcal{T})\}$. Finally, for a subtree $\mathcal{T}'$ of $\mathcal{T}$, let $\mu_{\mathcal{T}'}$ be defined as $\mathsf{dom}(\mu_{\mathcal{T}'}) := \mathsf{fvars}(\mathcal{T}')$ and $\mu_{\mathcal{T}'}(?x) := \mathsf{db}(?x)$ for each $?x \in \mathsf{dom}(\mu)$.

This allows us to characterize the *subsumption* problem in terms of PARTIAL OWL 2 QL-EVALUATION.

THEOREM 7. *Given pwdPTs $(\mathcal{T}_1, \mathcal{X}_1)$, $(\mathcal{T}_2, \mathcal{X}_2)$, and an ontology $\mathcal{O}$, we have $(\mathcal{T}_1, \mathcal{X}_1) \sqsubseteq_{\mathcal{O}} (\mathcal{T}_2, \mathcal{X}_2)$ iff for every subtree $\mathcal{T}_1'$ of $\mathcal{T}_1$ we have $\mu_{\mathcal{T}_1'} \sqsubseteq \mathsf{cert}(\mathcal{T}_2, \mathcal{X}_2, (\mathsf{db}(\mathcal{T}_1'), \mathcal{O}))$.*

Intuitively, the idea is that given some G and $\mu \in \mathsf{cert}(\mathcal{T}_1, \mathcal{X}_1, (G, \mathcal{O}))$, an extension $\mu' \sqsubseteq \mathsf{cert}(\mathcal{T}_2, \mathcal{X}_2, (G, \mathcal{O}))$ can be obtained by a combination of μ and the mapping witnessing $\mu_{\mathcal{T}_1'} \sqsubseteq \mathsf{cert}(\mathcal{T}_2, \mathcal{X}_2, (\mathsf{db}(\mathcal{T}_1'), \mathcal{O}))$.

The *containment* problem on the other hand can be characterized in terms of OWL 2 QL-SUBSUMPTION and OWL 2 QL-EVALUATION. In fact, the following result presents a characterization of $\not\subseteq_{\mathcal{O}}$, i.e. a characterization of the co-problem, since we think it is more intuitive.

THEOREM 8. *Given pwdPTs $(\mathcal{T}_1, \mathcal{X}_1)$, $(\mathcal{T}_2, \mathcal{X}_2)$, and an ontology $\mathcal{O}$. Then $(\mathcal{T}_1, \mathcal{X}_1) \not\subseteq_{\mathcal{O}} (\mathcal{T}_2, \mathcal{X}_2)$ iff*

1. *$\mathcal{X}_1 \neq \mathcal{X}_2$, or*
2. *$(\mathcal{T}_1, \mathcal{X}_1) \not\sqsubseteq_{\mathcal{O}} (\mathcal{T}_2, \mathcal{X}_2)$, or*
3. *there exist subtrees $\mathcal{T}_1'$ of $\mathcal{T}_1$ and $\mathcal{T}_2'$ of $\mathcal{T}_2$ with $\mathsf{fvars}(\mathcal{T}_1') \subsetneq \mathsf{fvars}(\mathcal{T}_2')$ s.t. $\mu_{\mathcal{T}_1'} \in \mathsf{cert}(\mathcal{T}_1, \mathcal{X}_1, (\mathsf{db}(\mathcal{T}_1') \cup \mathsf{db}(\mathcal{T}_2'), \mathcal{O}))$.*

We briefly discuss the idea of this characterization: By properties 1 and 2, containment cannot hold. Now assume $(\mathcal{T}_1, \mathcal{X}_1) \sqsubseteq_{\mathcal{O}} (\mathcal{T}_2, \mathcal{X}_2)$. Then clearly the only reason for some certain answer μ of $\mathcal{T}_1$ not being a certain answer of $\mathcal{T}_2$ is that some proper extension μ' of μ is actually a certain answer of $\mathcal{T}_2$. This situation is covered by the third property: $\mathcal{T}_1'$ represents μ, $\mathcal{T}_2'$ represents μ'. Finally $\mu_{\mathcal{T}_1'} \in \mathsf{cert}(\mathcal{T}_1, \mathcal{X}_1, (\mathsf{db}(\mathcal{T}_1') \cup \mathsf{db}(\mathcal{T}_2'), \mathcal{O}))$ ensures that while μ can be extended to μ' on $\mathcal{T}_2$, this is not possible on $\mathcal{T}_1$.

The *equivalence* problem can of course be solved by checking mutual containment. Moreover, it is easy to see that $(\mathcal{T}_1, \mathcal{X}_1) \equiv_{\mathcal{O}} (\mathcal{T}_2, \mathcal{X}_2)$ iff $(\mathcal{T}_1, \mathcal{X}_1) \sqsubseteq_{\mathcal{O}} (\mathcal{T}_2, \mathcal{X}_2)$ and $(\mathcal{T}_2, \mathcal{X}_2) \sqsubseteq_{\mathcal{O}} (\mathcal{T}_1, \mathcal{X}_1)$. Just consider the case that for some RDF graph G, assuming mutual subsumption, there is some $\mu \in \mathsf{cert}(\mathcal{T}_1, \mathcal{X}_1, (G, \mathcal{O}))$, s.t. $\mu \notin \mathsf{cert}(\mathcal{T}_2, \mathcal{X}_2, (G, \mathcal{O}))$. Then there must be $\mu' \in \mathsf{cert}(\mathcal{T}_2, \mathcal{X}_2, (G, \mathcal{O}))$ with $\mu \sqsubset \mu'$. Thus either μ' or some extension of it must be contained in $\mathsf{cert}(\mathcal{T}_1, \mathcal{X}_1, (G, \mathcal{O}))$, which contradicts $\mu \in \mathsf{cert}(\mathcal{T}_1, \mathcal{X}_1, (G, \mathcal{O}))$.

Observe that Theorems 7 and 5 give an immediate Π_2^P upper bound on the complexity of OWL 2 QL-SUBSUMPTION. Also, this result combined with Theorem 6 provides a Π_2^P upper bound for OWL 2 QL-CONTAINMENT (via a Σ_2^P bound for the co-problem). Finally, these results imply the same upper bound for OWL 2 QL-EQUIVALENCE. By proving matching lower bounds, we show that this is in fact the exact complexity of all three problems.

THEOREM 9. *The problems* OWL 2 QL-SUBSUMPTION*,* OWL 2 QL-CONTAINMENT*, and* OWL 2 QL-EQUIVALENCE *are Π_2^P-complete. Hardness holds even for empty ontologies.*

PROOF IDEA. *Membership* results follow from the above discussion. *Hardness* for OWL 2 QL-SUBSUMPTION follows from the hardness of subsumption without ontologies [19]. The hardness for OWL 2 QL-CONTAINMENT and OWL 2 QL-EQUIVALENCE is shown in the full version. □

Of course, we cannot compare these results directly with the known results for well-designed SPARQL queries in settings without ontologies. In fact, over plain RDF graphs, most of the problems have even a higher complexity: The evaluation problem is Σ_2^P-complete [19] while the containment and equivalence problem are even undecidable [26]. Only the subsumption problem was shown to have the same complexity [26]. The reason for the higher complexity, or even undecidability, respectively, is the existence of subsumed mappings in query answers. Recall that, for any RDF graph G, pwdPT $(\mathcal{T}, \mathcal{X})$ and ontology $\mathcal{O}$, the set $[\![(\mathcal{T}, \mathcal{X})]\!]_G$ may contain subsumed mappings. This is not the case for $\mathsf{cert}(\mathcal{T}, \mathcal{X}, (G, \mathcal{O}))$, due to the restriction to maximal mappings in our definition of certain answers.

For the subsumption problem, this of course does not make a difference, since for two sets of mappings M, M' we clearly have $M \sqsubseteq M'$ if and only if $\mathrm{MAX}(M) \sqsubseteq \mathrm{MAX}(M')$ (observe however, that in the presence of subsumed answers, pairwise subsumption does not guarantee equivalence [19]). However, for a reasonable comparison of the other problems under our new certain answer semantics with the settings without ontologies, we need to study suitable variants of these problems first. We thus investigate the following problem.

MAXEVALUATION	
Input:	pwdPT $(\mathcal{T}, \mathcal{X})$, RDF graph G, mapping μ.
Question:	$\mu \in \mathrm{MAX}([\![(\mathcal{T}, \mathcal{X})]\!]_G)$?

For containment and equivalence, we consider the relations $\subseteq^{MAX}$ and $\equiv^{MAX}$ where, for pwdPTs $(\mathcal{T}_1, \mathcal{X}_1)$, $(\mathcal{T}_2, \mathcal{X}_2)$, we have $(\mathcal{T}_1, \mathcal{X}_1) \circ (\mathcal{T}_2, \mathcal{X}_2)$ (for $\circ \in \langle \subseteq^{MAX}, \equiv^{MAX} \rangle$) if $\mathrm{MAX}([\![(\mathcal{T}_1, \mathcal{X}_1)]\!]_G) \circ \mathrm{MAX}([\![(\mathcal{T}_2, \mathcal{X}_2)]\!]_G)$ (for $\circ \in \langle \subseteq, = \rangle$, resp.) for all RDF graphs G. This gives the following problems.

MAXCONTAINMENT/MAXEQUIVALENCE	
Input:	pwdPTs $(\mathcal{T}_1, \mathcal{X}_1)$, $(\mathcal{T}_2, \mathcal{X}_2)$.
Question:	$(\mathcal{T}_1, \mathcal{X}_1) \subseteq^{MAX} / \equiv^{MAX} (\mathcal{T}_2, \mathcal{X}_2)$?

THEOREM 10. *The problem* MAXEVALUATION *is* DP*-complete. The problems* MAXCONTAINMENT *and* MAXEQUIVALENCE *are* Π_2^P*-complete*

PROOF IDEA. The corresponding problems in the presence of an ontology of course provide an upper bound. *Hardness* results are provided in the full version. □

We have thus pinpointed the complexity of the most relevant reasoning problems related to query evaluation and query analysis: Starting from the problems respecting our new certain answer semantics, we also reconsidered the corresponding problems for settings without ontologies where we remove subsumed mappings from the query answers.

Our results show that for all the problems analysed in this section, applying the certain answer semantics does not increase their computational complexity compared to the case without ontologies.

8. CONCLUSION AND FUTURE WORK

Summary. In this work, we have provided an intuitive definition of certain answers for well-designed SPARQL queries under entailment regimes. As a basis for the design of evaluation algorithms, we have established the relationship between certain answers and canonical models. It should be noted that this part of our work is completely generic in the sense that it applies to any entailment regime. We have then turned our attention to a concrete entailment regime – OWL 2 QL – for which we have presented novel evaluation algorithms together with a detailed complexity analysis. We have thus shown that – analogously to CQ-answering under DL-Lite [7] – one can extend SPARQL evaluation to OWL 2 QL entailment without significantly increasing the complexity.

Related Work. Work related to our findings first and foremost includes the work our approaches are based upon [7, 9, 12, 2] as discussed throughout this paper. There is a huge body of results on query answering under different Description Logics (cf. [7, 27, 9, 23]). However, the queries considered there are typically CQs. So far, query languages comparable to well-designed SPARQL have not yet been studied in the context of Description Logics. The problem of answering SPARQL queries under OWL 2 QL via rewriting has been recently studied in [16], where a translation of the problem into SQL is provided. Unlike our work, the authors do not modify the semantics defined by the entailment regime [12]. Investigating aggregate functions, similar to our observation in the presence of weak monotonicity, the authors of [18] notice that defining certain answers as the intersection over all possible worlds does not provide satisfactory answers. For the counting operator, they solve this problem by defining the certain answer to be the minimum over all possible worlds. In [1], the authors describe a rewriting of SPARQL query answering under OWL 2 QL into Datalog$^{\pm}$. A modification of this translation allows them to remove the restriction to map all variables to actual values from the RDF graph. However, this relaxation applies only to variables occurring in a single BGP, while we allow this for all non-distinguished variables. Also, a discussion of the resulting semantics is missing. Libkin [20] also criticizes the standard notion of certain answers in case of non-monotone queries. Similar to his suggestion to use the greatest lower bounds in terms of informativeness, our approach chooses the most informative solutions as certain answers.

Future Work. Above all, we want to investigate further entailment regimes under our certain answer semantics. The implementation of the rewriting algorithms, the development of suitable benchmarks and alternative methods, e.g. employing materialization [15], is a challenging task as well. It also remains to be seen whether the optimization in [14] can be adopted by the proposed semantics. Moreover, we want to study combinations of our semantics with other approaches: our definition of certain answers is a *relaxation* of the current semantics; it allows to infer additional mappings that logically follow from the knowledge base. Recently, a *stronger* semantics was presented in [17], which discards entire mappings whose possible extensions to optional subqueries would imply inconsistencies in the knowledge base. A semantics that would integrate both principles is also an exciting research problem. Another extension envisaged refers to the restrictions imposed on the fragment of SPARQL considered here: so far, we have only considered *well-designed SPARQL*, which was *further restricted* to queries of the form $(?x, \mathsf{a}, B)$ or $(?x, Q, ?y)$. However, since SPARQL allows for variables in all positions, i.e. also $(?x, ?y, ?z)$ is a valid triple pattern, we will extend our work to allow these triple patterns as well. Even though we obtain nice complexity results, by investigating well-designed SPARQL, for the certain answer semantics this restriction should not be of big importance. Hence, we want to further investigate the certain answer semantics in the presence of standard SPARQL queries over OWL 2 QL ontologies.

Acknowledgements

This work was supported by the Vienna Science and Technology Fund (WWTF), project ICT12-15 and by the Austrian Science Fund (FWF): P25207-N23 and P25518-N23.

9. REFERENCES

[1] M. Arenas, G. Gottlob, and A. Pieris. Expressive languages for querying the semantic web. In *Proc. of PODS 2014*, pages 14–26. ACM, 2014.

[2] M. Arenas and J. Pérez. Querying semantic web data with SPARQL. In *Proc. of PODS 2011*, pages 305–316. ACM, 2011.

[3] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] M. Bienvenu, T. Eiter, C. Lutz, M. Ortiz, and M. Šimkus. Query answering in the description logic $\mathcal{S}$. In *Proc. of DL 2010*. CEUR-WS.org, 2010.

[5] M. Bienvenu, M. Ortiz, M. Šimkus, and G. Xiao. Tractable queries for lightweight description logics. In *Proc. of IJCAI 2013*. IJCAI/AAAI, 2013.

[6] S. Bischof, M. Krötzsch, A. Polleres, and S. Rudolph. Schema-agnostic query rewriting in SPARQL 1.1. In *Proc. of ISWC 2014*, pages 584–600. Springer, 2014.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[8] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of STOC 1977*, pages 77–90. ACM, 1977.

[9] T. Eiter, M. Ortiz, M. Šimkus, T. Tran, and G. Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proc. of AAAI 2012*. AAAI Press, 2012.

[10] B. Glimm. Using SPARQL with RDFS and OWL entailment. In *Reasoning Web 2011, Tutorial Lectures*, pages 137–201. Springer, 2011.

[11] B. Glimm, C. Lutz, I. Horrocks, and U. Sattler. Conjunctive query answering for the description logic SHIQ. *J. Artif. Intell. Res. (JAIR)*, 31:157–204, 2008.

[12] B. Glimm and C. Ogbuji. SPARQL 1.1 Entailment Regimes. W3C Recommendation, W3C, Mar. 2013. `http://www.w3.org/TR/sparql11-entailment`.

[13] F. Goasdoué, I. Manolescu, and A. Roatis. Efficient query answering against dynamic RDF databases. In *In Proc. of EDBT 2013*, pages 299–310. ACM, 2013.

[14] I. Kollia and B. Glimm. Optimizing SPARQL query answering over OWL ontologies. *J. Artif. Intell. Res. (JAIR)*, 48:253–303, 2013.

[15] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in DL-Lite. In *Proc. of KR 2010*. AAAI Press, 2010.

[16] R. Kontchakov, M. Rezk, M. Rodriguez-Muro, G. Xiao, and M. Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *Proc. of ISWC 2014*, pages 552–567. Springer, 2014.

[17] E. V. Kostylev and B. C. Grau. On the semantics of SPARQL queries with optional matching under entailment regimes. In *Proc. of ISWC 2014*, pages 374–389. Springer, 2014.

[18] E. V. Kostylev and J. L. Reutter. Answering counting aggregate queries over ontologies of the DL-Lite family. In *Proc. of AAAI 2013*. AAAI Press, 2013.

[19] A. Letelier, J. Pérez, R. Pichler, and S. Skritek. Static analysis and optimization of semantic web queries. *ACM Trans. Database Syst.*, 38(4):25, 2013.

[20] L. Libkin. Incomplete data: what went wrong, and how to fix it. In *Proc. PODS 2014*, pages 1–13. ACM, 2014.

[21] C. Lutz. The complexity of conjunctive query answering in expressive description logics. In *Proc. of IJCAR 2008*, pages 179–193. Springer, 2008.

[22] B. Motik, Y. Nenov, R. Piro, I. Horrocks, and D. Olteanu. Parallel materialisation of datalog programs in centralised, main-memory RDF systems. In *Proc. AAAI 2014*, pages 129–137. AAAI Press, 2014.

[23] M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008.

[24] M. Ortiz, S. Rudolph, and M. Šimkus. Query answering in the horn fragments of the description logics SHOIQ and SROIQ. In *Proc. of IJCAI 2011*, pages 1039–1044. IJCAI/AAAI, 2011.

[25] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3), 2009.

[26] R. Pichler and S. Skritek. Containment and equivalence of well-designed SPARQL. In *Proc. of PODS 2014*, pages 39–50. ACM, 2014.

[27] R. Rosati. On conjunctive query answering in EL. In *Proc. DL 2007*. CEUR-WS.org, 2007.

[28] R. Rosati and A. Almatelli. Improving query answering over DL-Lite ontologies. In *Proc. of KR 2010*. AAAI Press, 2010.

[29] M. Schmidt, M. Meier, and G. Lausen. Foundations of SPARQL query optimization. In *Proc. of ICDT 2010*, pages 4–33. ACM, 2010.

[30] J. F. Sequeda, M. Arenas, and D. P. Miranker. OBDA: query rewriting or materialization? In practice, both! In *Proc. of ISWC 2014*, pages 535–551. Springer, 2014.