Inside the Query Space of DL Knowledge Bases

Alexandros Chortaras, Michalis Giazitzoglou, and Giorgos Stamou

National Technical University of Athens, Greece
achort@cs.ntua.gr, mikegiatzi@image.ntua.gr, gstam@cs.ntua.gr

Abstract. We structure the query space of DL knowledge bases (KBs) by finding equivalent and partial ordering relations between answer sets of conjunctive queries, and defining a formal graph structure that allows qualitative and quantitative exploration of the query space. We also discuss how to compute such a graph for restricted DL-Lite$_R$ KBs.

1 Introduction

Purely assertional or ontology-enriched KBs are increasingly being used by applications that need access to data and knowledge, and are typically accessed through queries or facet-based interfaces. When interacting with such a KB, the application usually has no idea about the quality or completeness of its coverage, e.g. which queries have an 'adequate', representative number of sufficiently diverse answers indicating that the KB is a good source for answering such queries, or which may produce degenerate answers. General-purpose KBs may in fact have significantly unbalanced contents with respect to topic coverage.

To explore this aspect of KBs, in this paper we propose a theoretical framework for structuring the space of conjunctive queries that can be posed on a DL KB in a semi-lattice structure, called the query space graph (QSG) using query answers. The QSG captures significant information about the coverage of a particular KB, and reveals, apart from any partial and equivalent ordering relations following from the theory (e.g. query containment relations [4]), any additional, potentially informing, epistemic such relations. Provided that the QSG satisfies certain properties, it fully represents the underlying query space and encodes the answers to all queries. Our work draws ideas from Formal Concept Analysis (FCA) [7], but the structure we propose can represent arbitrary queries; in the case of queries consisting only of concept atoms our graph reduces to a concept lattice. We also link the graph construction with the query generation and answering phases. Because the size of the query space explodes combinatorially, computation of all queries is practically infeasible unless their answers are considered, so as to restrict them to those having at least one answer. In this context, we propose an algorithm for computing the query space of restricted DL-Lite$_R$ KBs. Our work shares also ideas with faceted search [12], since a path in the QSG captures conjunctive faceted query navigation [2].

In the rest of the paper, after some preliminaries in Section 2, Section 3 builds the theoretical framework underlying QSGs. Section 4 outlines the general approach to compute a QSG, and Section 5 specializes it for DL-Lite$_R^-$. Section 6 presents some results for DBPedia, and Section 7 concludes the paper.

2 Preliminaries

Let CN, RN, IN be mutually disjoint, finite sets of *concept*, *role* and *individual* names, respectively. The triple $\langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN} \rangle$ is a *vocabulary* $\mathcal{V}$, and a TBox $\mathcal{T}$ (ABox $\mathcal{A}$) over $\mathcal{V}$ and a DL dialect $\mathcal{L}$ is a set of axioms (assertions) that use elements of $\mathcal{V}$ and constructors of $\mathcal{L}$. The pair $\langle \mathcal{V}, \mathcal{L} \rangle$ is a *language*, and $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a *knowledge base* (KB) over some language. In part of the paper we use DL-Lite$^{-}_{\mathcal{R}}$ as dialect $\mathcal{L}$, a subset of DL-Lite$_{\mathcal{R}}$ that excludes axioms of the form $S \sqsubseteq R^{-}$. In particular, it allows only axioms of the form $C \sqsubseteq D$ or $R \sqsubseteq S$, where C, D are concepts and R, S atomic roles. D can be either atomic or of the form $\exists R^{(-)}.\top$, and C can also be of the form $\exists R^{(-)}.A$, where A is atomic. The semantics of KBs are defined in the standard way using interpretations [3]. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ assigns a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to concept C, and a set $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to role r. $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all $C(a) \in \mathcal{A}$, and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all $r(a,b) \in \mathcal{A}$. $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$. An axiom (or assertion) τ is entailed by a TBox $\mathcal{T}$ (a KB $\mathcal{K}$) if it is satisfied in every model of $\mathcal{T}$ ($\mathcal{K}$), and we write $\mathcal{T} \models \tau$ ($\mathcal{K} \models \tau$).

Given a vocabulary $\mathcal{V}$ and a set of variable names VN, a *conjunctive query* (simply, a *query*) Q is an expression $\{ \langle x_1, \dots x_k \rangle \mid \exists y_1 \dots \exists y_l . (a_1 \wedge \dots \wedge a_n) \}$, where $k, l \geq 0$, $n \geq 1$, $x_i, y_i \in \mathsf{VN}$, each a_i is an atom $C(u)$ or $R(u,v)$, where $C \in \mathsf{CN}$, $R \in \mathsf{RN}$, $u, v \in \mathsf{IN} \cup \{x_i\}_{i=1}^{k} \cup \{y_i\}_{i=1}^{l}$ and all x_i, y_i appear in at least one atom. The vector $\boldsymbol{x} = \langle x_1, \dots x_k \rangle$ is the *head* of Q ($\mathsf{head}(Q)$), its elements are the *answer variables* ($\mathsf{avar}(\mathrm{Q})$), and $\{a_1, \dots, a_n\}$ is the *body* of Q ($\mathsf{body}(Q)$). $\mathsf{var}(Q)$ is the set of all variables appearing in Q. Those that appear once only in the body are *unbound*; all others are *bound*. If $\mathsf{avar}(Q) = \varnothing$, the query is *boolean*. In this paper we will talk about non-boolean queries containing only variables ($u, v \notin \mathsf{IN}$). We will often write Q as ${}_{\boldsymbol{x}}\{a_1, ..., a_n\}$, or ${}_{\boldsymbol{x}}Q$, to make explicit the answer variables. Given a KB $\mathcal{K}$, a query Q with head $\langle x_1, \dots x_k \rangle$ and a model $\mathcal{I}$ for $\mathcal{K}$, a vector of individuals $\boldsymbol{c} = \langle c_1, \dots c_k \rangle$, $c_i \in \mathsf{IN}$, is an *answer* to Q in $\mathcal{I}$ iff $\mathcal{K}, \mathcal{I} \models Q(\boldsymbol{a})$, where $Q(\boldsymbol{a}) = Q\{x_i / c_i\}_{i=1}^{k}$. $\boldsymbol{c}$ is a *certain answer* to Q for $\mathcal{K}$, iff it is an answer in all models $\mathcal{I}$ of $\mathcal{K}$. The set of certain answers to Q is $\mathsf{cert}(Q, \mathcal{K})$.

A query Q_1 *subsumes* a query Q_2 ($Q_1 \subseteq_S Q_2$) iff there is a substitution θ s.t. $\mathsf{head}(Q_1\theta) = \mathsf{head}(Q_2)$ and $\mathsf{body}(Q_1\theta) \subseteq \mathsf{body}(Q_2)$. If Q_1, Q_2 are mutually subsumed they are *syntactically equivalent* ($Q_1 \equiv_S Q_2$). We write $Q \in \mathcal{Q}$ iff set $\mathcal{Q}$ contains a query Q' such that $Q' \equiv_S Q$. Two queries Q_1, Q_2 are *similar* if there are renamings σ_1, σ_2 s.t. $\mathsf{head}(Q_1\sigma_1) = \mathsf{head}(Q_2)$ and $\mathsf{head}(Q_2\sigma_2) = \mathsf{head}(Q_1)$. (A substitution $\theta = \{x_i / y_i\}_{i=1}^{n}$ is a *renaming* for a query Q if $x_i \in \mathsf{var}(Q)$, and $\{y_i\}_{i=1}^{n}$ are distinct variables s.t. each y_i equals some x_j or $y_i \notin \mathsf{var}(Q)$ [10].) Given Q, let Q' be a query s.t. $\mathsf{head}(Q') = \mathsf{head}(Q)$ and $\mathsf{body}(Q') \subseteq \mathsf{body}(Q)$. If $\mathsf{body}(Q')$ is a minimal subset of $\mathsf{body}(Q)$ s.t. $Q \subseteq_S Q'$, Q' is a *condensation* of Q ($\mathsf{cond}(Q)$). If the minimal $\mathsf{body}(Q')$ equals $\mathsf{body}(Q)$, Q is *condensed* [8].

We define the following operations: i) The *intersection* of two similar queries Q_1, Q_2 is the query $Q_1 \sqcap Q_2 \doteq \mathsf{cond}(_{\,\mathsf{head}(Q_1)}\{\mathsf{body}(Q_1) \cup \mathsf{body}(Q_2\sigma)\})$ where σ is a renaming s.t. $\mathsf{head}(Q_1) = \mathsf{head}(Q_2\sigma)$ and $\mathsf{var}(Q_2\sigma) \cap \mathsf{var}(Q_1) = \mathsf{avar}(Q_1)$, that renames all non-answer variables of Q_2 to variables not appearing in Q_1. Clearly, $Q_1 \sqcap Q_2 = Q_2 \sqcap Q_1$. We will write ${}_{\boldsymbol{x}}Q_1 \sqcap {}_{\boldsymbol{x}}Q_2$, assuming that $\mathsf{var}({}_{\boldsymbol{x}}Q_1) \cap \mathsf{var}({}_{\boldsymbol{x}}Q_2)$

equals the variables in $\boldsymbol{x}$. ii) The τ-*concatenation* of query Q_1 with query Q_2, where τ is a renaming of a subset of $\mathsf{var}(Q_2)$ to a subset of $\mathsf{var}(Q_1)$, is the query $Q_1 \circ_\tau Q_2 \doteq \mathsf{cond}(\,_{\mathsf{head}(Q_1)}\{\mathsf{body}(Q_1) \cup \mathsf{body}(Q_2\sigma\tau)\}\,)$ where σ is a renaming that renames all variables of Q_2 that are not renamed by τ to variables not appearing in Q_1. Dropping τ, we will write $\,_{\boldsymbol{x}}Q_1 \circ_{\boldsymbol{x}'} Q_2$, assuming that $\mathsf{var}(\,_{\boldsymbol{x}}Q_1)$ contains all variables in $\boldsymbol{x}'$, and $\mathsf{var}(\,_{\boldsymbol{x}}Q_1) \cap \mathsf{var}(\,_{\boldsymbol{x}'}Q_2))$ equals the variables in $\boldsymbol{x}'$. If $\boldsymbol{x} = \boldsymbol{x}'$, $\,_{\boldsymbol{x}}Q_1 \circ \,_{\boldsymbol{x}}Q_2 = \,_{\boldsymbol{x}}Q_1 \sqcap \,_{\boldsymbol{x}}Q_2$. iii) The *projection* of a query $\,_{\boldsymbol{x}}Q$ on $\boldsymbol{y}$, where the variables in $\boldsymbol{y}$ are a subset of the variables in $\boldsymbol{x}$, is the query $\,_{\boldsymbol{x}}Q|_{\boldsymbol{y}} \doteq \,_{\boldsymbol{y}}\{\mathsf{body}(\,_{\boldsymbol{x}}Q)\}$.

Given a query Q, let $\mathsf{gra}(Q)$ ($\bar{\mathsf{gra}}(Q)$) be the (directed) graph with nodes $\mathsf{var}(Q)$ having the edge $\{z_1, z_2\}$ (resp. (z_1, z_2)) iff $R(z_1, z_2) \in \mathsf{body}(Q)$. Q is *connected* iff $\mathsf{gra}(Q)$ is connected, and *(strictly) tree-shaped* iff $\mathsf{head}(Q) = \langle x \rangle$ and $\mathsf{gra}(Q)$ ($\bar{\mathsf{gra}}(Q)$) is a (directed) tree with root x. By $\mathcal{Q}_{\mathcal{V}}^{\langle k, n_1 : n_2 \rangle}$ we denote the syntactically equivalent-free set of all condensed, connected, non-boolean queries with $k \geq 1$ answer variables, head of length k, n variables where $k \leq n_1 \leq n \leq n_2$, constructable over $\mathcal{V}$. We write $\mathcal{Q}_{\mathcal{V}}^{\langle k, n \rangle}$ for $\mathcal{Q}_{\mathcal{V}}^{\langle k, k:n \rangle}$, and $\mathcal{Q}_{\mathcal{V}}^{k}$ for $\bigcup_{j=1}^{+\infty} \mathcal{Q}_{\mathcal{V}}^{\langle k, j \rangle}$.

3 Structuring the Query Space

In general, to structure a set of similar queries we need a partial ordering relation $\preceq$. Using query subsumption we can write $Q_1 \preceq_S Q_2$ iff $Q_2 \subseteq_S Q_1$, whereas using query containment we can write $Q_1 \preceq_{\mathcal{T}} Q_2$, iff for given a TBox $\mathcal{T}$ we have $\mathsf{cert}(Q_1, \langle \mathcal{T}, \mathcal{A} \rangle) \subseteq \mathsf{cert}(Q_2, \langle \mathcal{T}, \mathcal{A} \rangle)$ for any ABox $\mathcal{A}$. These partial ordering relations capture general properties of the queries that follow from their structure or the axioms of the underlying TBox. In concrete KBs however, comprising particular ABoxes, arise more specific, incidental 'strict containment' or 'equivalence' relations. To capture such relations we need a strict partial ordering relation that takes into account the actual ABox contents.

Definition 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB over some $\langle \mathcal{V}, \mathcal{L} \rangle$, and Q_1, Q_2 two (similar) queries in $\mathcal{Q}_{\mathcal{V}}^{k}$, for some k. Q_1 and Q_2 are* answer equivalent *w.r.t. $\mathcal{K}$ ($Q_1 \equiv_{\mathcal{K}} Q_2$), if $\mathsf{cert}(Q_1, \mathcal{K}) = \mathsf{cert}(Q_2, \mathcal{K})$. Also, Q_1 is* answer smaller *than Q_2 w.r.t. $\mathcal{T}$ ($Q_1 \prec_{\mathcal{K}} Q_2$), if $\mathsf{cert}(Q_1, \mathcal{K}) \subset \mathsf{cert}(Q_2, \mathcal{K})$.*

The smaller-or-equivalent relation $\preceq_{\mathcal{K}}$ can then be defined as usual. If $Q_1 \preceq_S Q_2$ then $Q_1 \preceq_{\mathcal{T}} Q_2$ for any $\mathcal{T}$; given $\mathcal{T}$, if $Q_1 \preceq_{\mathcal{T}} Q_2$ then $Q_1 \preceq_{\mathcal{K}} Q_2$ regardless of the particular $\mathcal{A}$. $\equiv_{\mathcal{K}}$ and $\prec_{\mathcal{K}}$ can be considered as 'epistemic-like' equivalence and strict partial ordering relations, induced by a particular ABox.

Based on the above, we can define an equivalence and strict partial ordering relation for any $\mathcal{Q} \subseteq \mathcal{Q}_{\mathcal{V}}^{k}$. An equivalence relation $\equiv$ partitions $\mathcal{Q}$ to a set of equivalence classes $\mathsf{EC}_{\equiv}(\mathcal{Q})$. We call the equivalence class that contains all queries with zero answers *null*, and its members *null queries*. We will say that a pair $(\equiv, \prec)$ on a set $\mathcal{Q}$ is a *structure* if for any $Q_1, Q_2 \in \mathcal{Q}$ s.t. $Q_1 \equiv Q_2$ we have $Q_1 \not\prec Q_2$ and $Q_2 \not\prec Q_1$, and inversely if $Q_1 \prec Q_2$ or $Q_2 \prec Q_1$, then $Q_1 \not\equiv Q_2$. Clearly, $(\equiv_{\mathcal{K}}, \prec_{\mathcal{K}})$ is a structure.

Definition 2. *A set $\mathcal{Q} \subseteq \mathcal{Q}_{\mathcal{V}}^{k}$ is* complete *iff for any $Q_1, Q_2 \in \mathcal{Q}$, we have $Q_1 \sqcap Q_2 \in \mathcal{Q}$.*

The *completion* of a finite set $\mathcal{Q}$ is constructed by iteratively extending $\mathcal{Q}$ with the intersections of its elements until reaching a fixpoint. This terminates because intersections are condensed queries. Since each equivalence class $\tilde{\mathcal{Q}} \in \mathsf{EC}_{\equiv}(\mathcal{Q})$ of a complete set $\mathcal{Q}$ contains all intersections of its elements, there is a unique $Q \in \tilde{\mathcal{Q}}$, called the *representative* of $\tilde{\mathcal{Q}}$, s.t. $Q' \subseteq_S Q$ for all $Q' \in \tilde{\mathcal{Q}}$. The representative is the syntactically most 'specific' query of the equivalent class.

Definition 3. *Let $\mathcal{Q} \subseteq \mathcal{Q}^k_{\mathcal{V}}$ and $(\equiv, \prec)$ be a structure. The* representative ordering *of $\mathcal{Q}$ is the partially ordered set $(\hat{\mathcal{Q}}_{\equiv}, \prec)$, where $\hat{\mathcal{Q}}$ is the completion of $\mathcal{Q}$, and $\hat{\mathcal{Q}}_{\equiv} = \bigcup_{\tilde{\mathcal{Q}} \in \mathsf{EC}_{\equiv}(\hat{\mathcal{Q}})} \{Q \in \tilde{\mathcal{Q}} \mid Q' \subseteq_S Q \text{ for all } Q' \in \tilde{\mathcal{Q}}\}$ (the set of representatives).*

Example 1. Consider that $\mathcal{Q} = \{{}_x\{A(x)\}, {}_x\{A(x), B(x)\}, {}_x\{C(x), D(x)\}\}$ and ${}_x\{C(x), D(x)\} \prec {}_x\{A(x)\}$ and ${}_x\{A(x)\} \equiv {}_x\{A(x), B(x)\}$. The completion of $\mathcal{Q}$ is $\hat{\mathcal{Q}} = \{{}_x\{A(x)\}, {}_x\{A(x), B(x)\}, {}_x\{C(x), D(x)\}, {}_x\{A(x), C(x), D(x)\}, {}_x\{A(x), B(x), C(x), D(x)\}\}$. The equivalence classes are $\{{}_x\{A(x)\}, {}_x\{A(x), B(x)\}\}$ and $\{{}_x\{C(x), D(x)\}, {}_x\{A(x), C(x), D(x)\}, {}_x\{A(x), B(x), C(x), D(x)\}\}$, thus $\hat{\mathcal{Q}}_{\equiv} = \{{}_x\{A(x), B(x)\}, {}_x\{A(x), B(x), C(x), D(x)\}\}$ with ${}_x\{A(x), B(x), C(x), D(x)\} \prec {}_x\{A(x), B(x)\}$.

Lemma 1. *Let $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ be the representative ordering of some $\mathcal{Q} \subseteq \mathcal{Q}^k_{\mathcal{V}}$. For any $Q_1, Q_2 \in \hat{\mathcal{Q}}_{\equiv}$ we have $Q_1 \prec Q_2$ iff $Q_2 \subset_S Q_1$.*

Proof. Assume that $Q_1 \prec Q_2$ and $Q_2 \not\subset_S Q_1$. Since $Q_1, Q_2 \in \hat{\mathcal{Q}}_{\equiv}$ we have $Q_1, Q_2 \in \hat{\mathcal{Q}}$, where $\hat{\mathcal{Q}}$ is the completion of $\mathcal{Q}$, and so $Q_1 \sqcap Q_2 \in \hat{\mathcal{Q}}$. Because $Q_1 \prec Q_2$, $Q_1 \sqcap Q_2$ must belong to the same equivalence class as Q_1. But $Q_2 \not\subset_S Q_1$, and so $Q_1 \subset_S Q_1 \sqcap Q_2 \neq Q_1$. Thus, $Q_1 \notin \hat{\mathcal{Q}}_{\equiv}$ which is a contradiction. Inversely, if $Q_2 \subset_S Q_1$, given that $Q_2 \not\equiv Q_1$, by definition we get $Q_1 \prec Q_2$.

We say that a query $Q \in \hat{\mathcal{Q}}_{\equiv}$ is *minimal* (*maximal*) w.r.t. to some property if it satisfies the property and there is no $Q' \in \hat{\mathcal{Q}}_{\equiv}$ s.t. $Q' \prec Q$ ($Q \prec Q'$) satisfying the same property.

Definition 4. *Let $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ be the representative ordering of some $\mathcal{Q} \subseteq \mathcal{Q}^k_{\mathcal{V}}$, and let $Q \in \mathcal{Q}^k_{\mathcal{V}}$. The* top (bottom) cover *of Q in $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ is the set of all minimal (maximal) $Q' \in \hat{\mathcal{Q}}_{\equiv}$ s.t $Q' \subseteq_S Q$ ($Q \subseteq_S Q'$).*

The covers may be empty, e.g. if $\hat{\mathcal{Q}}_{\equiv} = \{{}_x\{A(x)\}\}$ and $Q = {}_x\{B(x)\}$.

Lemma 2. *Let $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ be the representative ordering of some $\mathcal{Q} \subseteq \mathcal{Q}^k_{\mathcal{V}}$, and let $Q \in \mathcal{Q}^k_{\mathcal{V}}$. If the top and bottom covers, $\mathcal{Q}^{\top}$ and $\mathcal{Q}^{\bot}$, respectively, of Q are non-empty, then there is a $\tilde{Q} \in \hat{\mathcal{Q}}_{\equiv}$ s.t. $Q' \preceq Q \preceq \tilde{Q}$ for all $Q' \in \mathcal{Q}^{\bot}$.*

Proof. Let $\bar{Q}$ be the intersection of the queries in $\mathcal{Q}^{\top}$. By definition, $\bar{Q} \subseteq_S Q$, and hence $Q \preceq \bar{Q}$. By completeness there is a unique $\tilde{Q} \in \bar{\mathcal{Q}}_{\equiv}$ s.t. $\tilde{Q} \equiv \bar{Q}$, and so $Q \preceq \tilde{Q}$. Finally, by definition, for all $Q' \in \mathcal{Q}^{\bot}$ we have $Q \subseteq_S Q'$, hence $Q' \preceq Q$.

This allows us to use $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ to bound the answers to any query $Q \in \mathcal{Q}^k_{\mathcal{V}}$ even if $Q \notin \mathcal{Q}$, provided that $\mathcal{Q}^{\top}$, $\mathcal{Q}^{\bot}$ exist. If $Q \in \mathcal{Q}$, we have a stronger result.

Lemma 3. *Let $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ be the representative ordering of some $\mathcal{Q} \subseteq \mathcal{Q}^k_{\mathcal{V}}$. The bottom cover $\mathcal{Q}^{\perp}$ of any $Q \in \mathcal{Q}$ is a singleton.*

Proof. Since $Q \in \mathcal{Q}$, there is a unique $\tilde{\mathcal{Q}} \in \mathsf{EC}_{\equiv}(\hat{\mathcal{Q}})$, s.t. $Q \in \tilde{\mathcal{Q}}$. The representative Q' is s.t. $Q' \equiv Q$ and $Q \subseteq_S Q'$. This is clearly maximal and is in $\mathcal{Q}^{\perp}$. For any other $Q'' \in \hat{\mathcal{Q}}_{\equiv}$ s.t. $Q \subseteq_S Q''$, we have $Q'' \prec Q'$, hence it cannot be in $\mathcal{Q}^{\perp}$.

This unique element of $\mathcal{Q}^{\perp}$ is the *anchor* of Q in $(\hat{\mathcal{Q}}_{\equiv}, \prec)$, and the representative ordering of $\mathcal{Q}$ fully describes $\mathcal{Q}$: any query $Q \in \mathcal{Q}$ is represented in $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ by its unique anchor, and the answers to Q are the answers to its anchor.

Example 2. Let ${}_x\{A(x)\} \prec {}_x\{S(x,y)\} \prec {}_x\{S(x,y), R(y,z), B(z)\}$ and $\mathcal{Q}$ consist of these queries. Then $\hat{\mathcal{Q}}_{\equiv} = \{\{{}_x\{A(x)\}\}, \{{}_x\{A(x), S(x,y)\}\}, \{{}_x\{A(x), S(x,y), R(y,z), B(z)\}\}$ with the obvious ordering. Let Q be ${}_x\{S(x,y), R(y,z)\}$. We have that $Q \notin \mathcal{Q}$, and the bottom cover of Q is $\{{}_x\{A(x), S(x,y), R(y,z), B(z)\}$. Note that ${}_x\{S(x,y), R(y,z), B(z)\} \in \mathcal{Q}$ has the same bottom cover.

This shows that 'anchors' can exist also for queries that do not belong in $\mathcal{Q}$. Thus, only if we know that a query belongs to $\mathcal{Q}$ we can be sure that the answers of the anchor are also exactly its answers.

If for the representative ordering $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ we define the meet operation $\wedge$ so that $Q_1 \wedge Q_2$ is the maximal query $Q \in \hat{\mathcal{Q}}_{\equiv}$ s.t. $Q_1 \sqcap Q_2 \subseteq_S Q$, then $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ becomes a meet semi-lattice. Given a finite $\mathcal{Q}$, $\hat{\mathcal{Q}}_{\equiv}$ is finite, thus the semi-lattice has a bottom. To capture the minimum needed partial ordering relations for a representative ordering we use a directed acyclic graph.

Definition 5. *Let $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ be the representative ordering of some $\mathcal{Q} \subseteq \mathcal{Q}^k_{\mathcal{V}}$. The* query space graph *(QSG) of $\mathcal{Q}$ is the transitive reduction of the directed acyclic graph induced by $(\hat{\mathcal{Q}}_{\equiv}, \prec)$.*

Because $(\hat{\mathcal{Q}}_{\equiv}, \prec)$ is a finite semi-lattice, the QSG is finite with a bottom, but may have more than one roots. The QSG is shortcut-free, i.e. if there is an edge (Q_1, Q_2) there is no path from Q_1 to Q_2 in $\mathcal{G}$ other than (Q_1, Q_2).

4 Query Space Graph Computation

Now, we will describe how, given a language $\langle \mathcal{V}, \mathcal{L} \rangle$ and a KB $\mathcal{K}$, we can compute the QSG for the $(\equiv_{\mathcal{K}}, \prec_{\mathcal{K}})$ structure and the set $\mathcal{Q}^{\langle 1,k \rangle *}_{\mathcal{V}}$ of tree-shaped queries with one answer variable and up to k variables. To do this, we need to perform two tasks: compute the set of queries for which to build the QSG and then arrange them into an actual QSG. We will start from the latter.

A given set of queries $\mathcal{Q}$ can be arranged into a graph, by first computing the equivalence classes (the nodes of the graph) and then adding the edges induced by the partial ordering. Such a graph will not be a QSG unless $\mathcal{Q}$ is complete. If not, all intersections of the queries in $\mathcal{Q}$ have to be iteratively computed. This can be done by exploiting the already produced graph, and let it guide the query intersection process in a way similar to FCA algorithms [9].

To compute the actual queries, we can compute first all possible tree shapes, the so-called *rooted trees* (e.g. there are 1, 1, 2, 4, 9, 20, 48, 115, 286, 719 such trees with $1 \dots 10$ nodes, [1]). We represent a rooted tree with $n > 0$ nodes as a set of edges (i, j) (i is the parent of j), where i, j are integers representing nodes. We assume that if j is a descendent of i, then $j > i$, so that the root is the smallest of the integers being used (usually 0). The tree $T = \varnothing$ consists only of a root. To compute all rooted trees with up to t nodes we can start e.g. from tree $\{(0, 1)\}$, and iteratively connect a new node to each existing node, taking care to discard equivalent representations of the same rooted trees (reflections).

Let $\mathbb{D}$ ($\mathbb{C}$) be the set of all non syntactically equivalent queries of the form ${}_x\{C_i(x)\}_{i=1}^k$, $C_i \in \mathsf{CN}$, $k \geq 1$ ($k = 1$), i.e. the queries consisting of concept atoms. Clearly, $\mathbb{D} = \mathcal{Q}_{\mathcal{V}}^{\langle 1,1\rangle *}$ and $|\mathbb{D}| = \sum_{i=1}^{|\mathsf{CN}|} \binom{|\mathsf{CN}|}{i} = 2^{|\mathsf{CN}|} - 1$. For a query Q known to be in $\mathbb{D}$ ($\mathbb{C}$), we use the adornment notation ${}^{\mathbb{D}}_{x}Q$ (${}^{\mathbb{C}}_{x}Q$). Similarly, let $\mathbb{S}$ ($\mathbb{R}$) be the set of all non syntactically equivalent queries of the form ${}_{x,y}\{R_i(\boldsymbol{x}_i)\}_{i=1}^k$, $R_i \in \mathsf{RN}$, $k \geq 1$ ($k = 1$), and $\boldsymbol{x}_i = (x, y)$ or (y, x), i.e. the queries consisting of role atoms. $|\mathcal{QS}| = \sum_{i=1}^{|\mathsf{RN}|} 2^i \binom{|\mathsf{RN}|}{i} = 3^{|\mathsf{RN}|} - 1$. For strictly tree-shaped queries, where all $\boldsymbol{x}_i$s must be the same, we denote the respective sets by $\mathbb{S}*$ and $\mathbb{R}*$. For a query Q known to be in $\mathbb{S}$ ($\mathbb{R}$), we use the notation ${}^{\mathbb{S}}_{x,y}Q$ (${}^{\mathbb{R}}_{x,y}Q$), and similarly for the starred versions. $\mathbb{D}$ ($\mathbb{S}$) can be obtained as the node set of the QSG of the completion of $\mathbb{C}$ ($\mathbb{R}$). These sets are the material to put on the edges and nodes of a rooted tree to get tree-shaped queries. The tree $T = \varnothing$ gives rise to the queries ${}^{\mathbb{D}}_{x_0}Q$; any tree with nodes $\{0, \dots, k\}$, $k \geq 1$, gives rise to the queries

$$\left([\,{}^{\mathbb{D}}_{x_0}Q \circ] \left((\dots (({}^{\mathbb{S}}_{\hat{x}_1,x_1}Q_1 [\circ\, {}^{\mathbb{D}}_{x_1}\tilde{Q}_1]) \circ \dots) \circ ({}^{\mathbb{S}}_{\hat{x}_k,x_k}Q_k [\circ\, {}^{\mathbb{D}}_{x_k}\tilde{Q}_k]) \right) \right) |_{x_0} \tag{1}$$

where $\hat{x}_i = x_{\mathsf{parent}(i)}$. The optional parts in square brackets specify concepts in the tree nodes. Due to the recurrence of ${}^{\mathbb{S}}_{\hat{x}_i,x_i}Q\,[\circ\, {}^{\mathbb{D}}_{x_i}\tilde{Q}]$, we define the *frame link*

$$\mathbb{L} = \mathbb{S} \cup \left\{ {}^{\mathbb{S}}_{x,y}Q_1 \circ {}^{\mathbb{D}}_{y}Q_2 \mid {}^{\mathbb{S}}_{x,y}Q_1 \in \mathbb{S},\, {}^{\mathbb{D}}_{y}Q_2 \in \mathbb{D} \right\} \tag{2}$$

i.e the set of queries with two variables, some roles connecting them and possibly some concepts on the second variable. By ${}^{\mathbb{L}}_{x,y}Q$ we will denote a query in $\mathbb{L}$. We define similarly (i.e. by replacing $\mathbb{S}$ with $\mathbb{S}*$ in Eq. 2) $\mathbb{L}*$ and ${}^{\mathbb{L}*}_{x,y}Q$.

Given the above, a simple approach to compute the queries induced by a rooted tree T is to compute $\mathbb{L}$ (or $\mathbb{L}*$), create a copy of it for each edge, appropriately renaming variables, and take all possible combinations between frame link elements. An alternative, backwards iterative approach is based on the observation that, if the rooted trees of sizes up to $k-1$ are available, a rooted tree of size k can be directly constructed from one or more such trees. In particular, a query with shape a rooted tree of size k can be constructed either by a one step *extension* of an existing query with shape a rooted tree S of size $k-1$ (by renaming variables, adding a new root and connecting the new to the old root) or by *merging* two or more existing queries with shape trees S_i (by renaming variables and joining them on their roots). In the first case, if ${}_y\mathcal{Q}$ is the set of all queries with shape S, its extension is obtained by the unary operator

$${}_x{\diamond}_y\mathcal{Q} = \left\{ ({}^{\mathbb{L}}_{x,y}Q \circ {}_yQ)|_x \mid {}^{\mathbb{L}}_{x,y}Q \in \mathbb{L} \text{ and } {}_yQ \in {}_y\mathcal{Q} \right\}$$

In the second case, let ${}_x\mathcal{Q}_i$ be the set of all queries with shape S_i. Merging is an n-ary operator $\otimes$, for $n \geq 1$:

$$ {}_x\mathcal{Q}_1 \otimes {}_x\mathcal{Q}_2 \otimes \ldots \otimes {}_x\mathcal{Q}_n = \{ {}_xQ_1 \sqcap {}_xQ_2 \sqcap \ldots \sqcap {}_xQ_n \mid {}_xQ_i \in {}_x\mathcal{Q}_i \} $$

Given the above, the set of all queries represented by Formula 1 are

$$ \mathbb{Q}_T = [\{ {}^{\mathbb{D}}_{x_0}Q \in \mathbb{D} \} \otimes]\, {}_{x_0}\tilde{\mathcal{Q}} \tag{3} $$

i.e. the optional merging of the queries including only concept atoms on the root variable, with the set ${}_{x_0}\tilde{\mathcal{Q}}$ of all queries with the shape of T. These are obtained for $i = 0$ from ${}_{x_i}\tilde{\mathcal{Q}} = \bigotimes_{j \in \mathsf{children}(i)} {}_{x_i}\hat{\mathcal{Q}}_{ij}$ which expresses the merging of the queries represented by the several branches of the tree assuming the root is node i. Each branch is either a set of queries in $\mathbb{L}$, if the branch has length 1, or the extension of the set of queries represented by the first child in the branch:

$$ {}_{x_i}\hat{\mathcal{Q}}_{ij} = \begin{cases} \left\{ {}^{\mathbb{L}}_{x_i,x_j}Q|_{x_i} \mid {}^{\mathbb{L}}_{x_i,x_j}Q \in \mathbb{L} \right\} & \text{if } \mathsf{children}(j) = \varnothing \\ {}_{x_i}\Diamond\, {}_{x_j}\tilde{\mathcal{Q}} & \text{otherwise} \end{cases} \tag{4} $$

Since there are infinite Ts, $\bigcup_T \mathbb{Q}_T$ is infinite. To guarantee a finite set of queries, we must set additional criteria, such as compute only queries with up to k variables. Such a set will not be complete. A QSG on a non-complete set will fully describe only the queries in the set (they will have unique anchors), but this approximation can in practice be useful. Finally, to arrange the queries in a QSG, we need their answers. If we can answer arbitrary queries, we could compute first the queries and then evaluate them. However, the number of queries explodes combinatorially, yet most of them will have zero answers. Thus, assuming that we are not interested in null queries, we can try to combine query generation with answering to avoid generating and evaluating many of the null queries.

5 Building a QSG for a DL-Lite$_R^-$ Knowledge Base

We will now discuss how we can construct and answer tree-shaped queries for normalized DL-Lite$_R^-$ KBs. We are interested only in non-null queries. Our approach will be based on the rewriting approach to query answering, which is particularly applicable to the DL-Lite family [5]: given a query Q and a DL-Lite KB $\langle \mathcal{T}, \mathcal{A} \rangle$, there is a set of queries $\mathsf{rewr}(Q)$, called the *rewritings* of Q, such that $\mathsf{cert}(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = \bigcup_{Q' \in \mathsf{rewr}(Q,\mathcal{T})} \mathsf{cert}(Q', \langle \varnothing, \mathcal{A} \rangle)$ i.e. the consequences of $\mathcal{T}$ are incorporated in the several Q's, which can then be answered ignoring $\mathcal{T}$ (for brevity, we will omit the KB arguments from $\mathsf{cert}(\cdot)$ and $\mathsf{rewr}(\cdot)$). Rewritings are 'transitive': A rewriting of a rewriting of Q is a rewriting of Q. It follows that if $Q' \in \mathsf{rewr}(Q)$ and $Q'' \in \mathsf{rewr}(Q')$ then $\mathsf{cert}(Q'') \subseteq \mathsf{cert}(Q') \subseteq \mathsf{cert}(Q)$.

We focus on DL-Lite$_R^-$ KBs, because the absence of axioms $R \sqsubseteq S^-$ guarantees that the edges of a tree-shaped query do not change direction during rewriting. In DL-Lite$_R^-$, the rewriting set of an atomic query is a set (union) of atomic queries: $\mathsf{rewr}({}_x\{C(x)\}) = \{ {}_x\{A(x)\} \mid \mathcal{T} \models A \sqsubseteq C \} \cup {}_x\{R(x,y)\} \mid \mathcal{T} \models$

$\exists R \sqsubseteq C\} \cup {}_x\{R(y,x)\} \mid \mathcal{T} \models \exists R^- \sqsubseteq C\}\,\}$, $\mathsf{rewr}({}_x\{R(x,y)\}) = \{\,{}_x\{B(x)\} \mid \mathcal{T} \models B \sqsubseteq \exists R\} \cup {}_x\{S(x,y)\} \mid \mathcal{T} \models \exists S \sqsubseteq \exists R\} \cup {}_x\{S(y,x)\} \mid \mathcal{T} \models \exists S^- \sqsubseteq \exists R\}\,\}$, and $\mathsf{rewr}({}_{x,y}\{R(x,y)\}) = \{\,{}_{x,y}\{S(x,y)\} \mid \mathcal{T} \models S \sqsubseteq R\}$.

An *unfolding* of query Q is a rewriting obtained by substituting each atom in $\mathsf{body}(Q)$ by one of its rewritings, according to the above, taking care to appropriately name newly introduced variables. An unfolding of Q (before condensation) retains all bound variables of Q. The set of all unfoldings of Q is $\mathsf{unfold}(Q)$. Apart from unfoldings, Q may have also *shrinking* rewritings. Shrinkings arise due to axioms of the form $A \sqsubseteq \exists R.C$. Given such an axiom, we have ${}_x\{A(x)\} \in \mathsf{rewr}(\{R(x,y), C(y)\}_x)$ (in fact $\{A(x)\}_x$ is also in $\mathsf{rewr}(\{S(x,y), D(y)\}_x)$ if $\mathcal{T} \models R \sqsubseteq S$ and $\mathcal{T} \models C \sqsubseteq D$). Query ${}_x\{A(x)\}$ is a shrinking of ${}_x\{R(x,y), C(y)\}$ and is an element of $\mathsf{shrink}_y(Q)$ because it contains *all* variables of Q apart from bound variable y, which has been eliminated by reasoning using an axiom of the form $A \sqsubseteq \exists R.C$. Let

$$\mathsf{shrink}(Q) = \{Q\} \cup \bigcup_{v \in \mathsf{var}(Q)} \left(\bigcup_{Q' \in \mathsf{shrink}_v(Q)} \mathsf{shrink}(Q') \right) \tag{5}$$

It can be proved [6, 11] that $\mathsf{rewr}(Q) = \bigcup_{Q' \in \mathsf{shrink}(Q)} \mathsf{unfold}(Q')$. If $\mathsf{var}(Q) = \mathsf{avar}(Q)$, then $\mathsf{rewr}(Q) = \mathsf{unfold}(Q')$, since answer variables cannot be eliminated.

Applying iteratively shrinking on a query using Eq. 5, we may end up with a query containing only concept atoms on the answer variable. If all of them have a common unfolding, by unfolding the query we will end up with a query consisting of a single concept atom.

Definition 6. *The* kernel *of a tree-shaped query ${}_xQ$ is the set*

$$\mathsf{kernel}({}_xQ) = \{\,{}^{\mathbb{C}}_x Q' \in \mathsf{rewr}({}_xQ) \mid \textit{there is no } {}^{\mathbb{C}}_x Q'' \not\equiv_S {}^{\mathbb{C}}_x Q' \textit{ s.t. } {}^{\mathbb{C}}_x Q' \in \mathsf{rewr}({}^{\mathbb{C}}_x Q'')\,\}$$

Theorem 1. *If ${}_xQ_1, \ldots, {}_xQ_n$ are tree-shaped queries, then*

$$\mathsf{cert}({}_xQ_1 \sqcap \ldots \sqcap {}_xQ_n) = \bigcap_{i=1}^{n} \mathsf{cert}({}_xQ_i) \quad \textit{and}$$

$$\mathsf{kernel}({}_xQ_1 \sqcap \ldots \sqcap {}_xQ_n) = \bigcup_{{}_xQ'_1 \in \mathsf{kernel}({}_xQ_1)} \cdots \bigcup_{{}_xQ'_n \in \mathsf{kernel}({}_xQ_n)} \mathsf{kernel}({}_xQ'_1 \sqcap \ldots \sqcap {}_xQ'_n)$$

Proof. Because all ${}_xQ_i$s share only the answer variable x, their intersection (and the intersections of their rewritings) has no shrinking on x. Thus, the rewritings of ${}_xQ_1 \sqcap \ldots \sqcap {}_xQ_n$ are all combinations of the rewritings of each ${}_xQ_i$: $\mathsf{rewr}({}_xQ_1 \sqcap \ldots \sqcap {}_xQ_n) = \bigcup_{{}_xQ'_1 \in \mathsf{rewr}({}_xQ_1)} \cdots \bigcup_{{}_xQ'_n \in \mathsf{rewr}({}_xQ_n)} {}_xQ'_1 \sqcap \ldots \sqcap {}_xQ'_n$. *Because these rewritings share only variable x we get the desired results.*

The above allows us to compute $\mathsf{cert}({}_xQ_1 \sqcap \ldots \sqcap {}_xQ_n)$ and $\mathsf{kernel}({}_xQ_1 \sqcap \ldots \sqcap {}_xQ_n)$ given the certain answers and kernels of the merged queries. Because a merging may be used to construct further queries, it should be kept either if it has a non-empty kernel (which can cause queries constructed as extensions of it be non-null), or it is itself non-null.

Theorem 2. *Let* ${}_xQ = ({}^{\mathbb{L}*}_{x,y}Q_1 \circ {}_yQ_2)|_x$*, where* ${}^{\mathbb{L}*}_{x,y}Q_1 \in \mathbb{L}$*, and* ${}_yQ_2$ *is a strictly tree-shaped query with no concept atoms on* y*. Then*

$$\mathsf{cert}({}_xQ) = \left(\mathsf{cert}({}^{\mathbb{L}*}_{x,y}Q_1) \bowtie_{2=1} \mathsf{cert}({}_yQ_2)\right)|_1 \cup \bigcup_{{}^{\mathbb{C}}_x\tilde{Q} \in \mathsf{kernel}({}_xQ)} \mathsf{cert}({}^{\mathbb{C}}_x\tilde{Q})$$

where $\mathsf{kernel}({}_xQ) = \bigcup_{{}^{\mathbb{C}}_y\tilde{Q}_2 \in \mathsf{kernel}({}_yQ_2)} \mathsf{kernel}(({}^{\mathbb{L}*}_{x,y}Q_1 \circ {}^{\mathbb{C}}_y\tilde{Q}_2)|_x)$*,* $\bowtie_{i=j}$ *denotes joining of the left- and right-hand side tuples on the* i*-th and* j*-th element respectively, and* $|_i$ *denotes projection on the* i*-th element of the tuples.*

Proof. If a rewriting ${}_x\tilde{Q}$ *of* ${}_xQ$ *contains* y*, no shrinking has been applied on* y*, so* ${}_x\tilde{Q} = ({}_{x,y}\tilde{Q}_1 \circ {}_y\tilde{Q}_2)|_x$*,* ${}_{x,y}\tilde{Q}_1 \in \mathsf{rewr}({}^{\mathbb{L}*}_{x,y}Q_1)$ *and* ${}_y\tilde{Q}_2 \in \mathsf{rewr}({}_yQ_2)$*. Such rewritings contribute answers* $\mathsf{cert}({}^{\mathbb{L}*}_{x,y}Q_1) \bowtie_{2=1} \mathsf{cert}({}_yQ_2))|_1$*. If* ${}_x\tilde{Q}$ *doesn't contain* y *but contains a variable* z*, it was obtained by shrinking directly* ${}_xQ$ *on* y*, or by shrinking on* y *some other shrinking of* ${}_xQ$ *(on other variables), or it is a rewriting of these. Let* ${}^{\mathbb{L}*}_{x,y}Q_1 = {}^{\mathbb{S}*}_{x,y}Q_{1A}\,[\circ^{\mathbb{D}}_y Q_{1B}]$*. Because there are no axioms* $S \sqsubseteq R^-$*,* y *occurs at the same place in all role atoms. Thus,* ${}^{\mathbb{S}*}_{x,y}Q_{1A}$ *contains only atoms of the form* $S(x,y)$ *(or* $S(y,x)$*), and* ${}_yQ_2$ *only atoms of the form* $S(z_i,y)$ *(or* $S(y,z_i)$*). This is impossible (*${}_xQ$ *is strictly tree-shaped), so no shrinking on* y *is applicable.*

If ${}_x\tilde{Q}$ *results by shrinking on* y *a shrinking* ${}_xQ'$ *of* ${}_xQ$ *on other variables,* ${}_xQ'$ *cannot contain variables other than* x *and* y*, otherwise we could shrink directly on* y*. Thus,* ${}_xQ' = ({}^{\mathbb{S}*}_{x,y}Q_{1A}\,[\circ^{\mathbb{D}}_y Q_{1B}] \circ {}^{\mathbb{D}}_y Q_3)|_x$*, where* ${}^{\mathbb{D}}_y Q_3$ *is a result of shrinking on the other variables. For shrinking on* y *to be applicable on* ${}_xQ'$ *it must be the case that for all* $E(y) \in \mathsf{body}({}^{\mathbb{D}}_y Q_{1B})$ *we have* $C \sqsubseteq E$*, and for all* $S(x,y) \in \mathsf{body}({}^{\mathbb{S}*}_{x,y}Q_{1A})$ *we have* $\mathcal{T} \models R \sqsubseteq S$*. There can be no* $S'(y,x) \in \mathsf{body}({}^{\mathbb{S}*}_{x,y}Q_{1A})$*. These are necessary for all such* $E(y), S(x,y)$ *to give their place to* $A(x)$ *in the shrinking due to* $A \sqsubseteq \exists R.C$*. Thus, we must have* ${}_x\{A(x)\} \in \mathsf{kernel}({}^{\mathbb{L}*}_{x,y}Q_1)$*. But for all* $E(y) \in \mathsf{body}({}^{\mathbb{D}}_y Q_3)$*, i.e. for all* ${}_y\{E(y)\} \in \mathsf{kernel}({}_yQ_2)$ *we must also have* $C \sqsubseteq E$*, i.e.* ${}_x\{A(x)\} \in \mathsf{kernel}(({}^{\mathbb{L}*}_{x,y}Q_1 \circ {}_y\{E(y)\})|_x)$*. Taking the union for all such* E *and* A *we get the second part of the desired result.*

If ${}_x\tilde{Q}$ *is an unfolding of the above rewritings, its answers are included in the certain answers of the query from which it has been obtained and we can ignore it.*

We call $(\mathsf{cert}({}^{\mathbb{L}*}_{x,y}Q_1) \bowtie_{2=1} \mathsf{cert}({}_yQ_2))|_1$ the *direct answers* of ${}_xQ$, because they are obtained directly by joining the answers of ${}^{\mathbb{L}*}_{x,y}Q_1$ and ${}_yQ_2$. So, the theorem tells us that for the extension we have to stick to a strictly tree-shaped query ${}_xQ$, compute its direct answers, its kernel $({}^{\mathbb{L}*}_{x,y}Q_1 \circ {}_yQ_2)|_x$ (needing for that only the kernel of ${}_yQ_2$), find the answers to the kernel queries, and add them to the direct answers.

To implement Formula 3 we still need to compute queries ${}^{\mathbb{L}*}_{x,y}Q$ and ${}^{\mathbb{L}*}_{x,y}Q|_x$, used in the second and first branch of Eq. 4, respectively. For this, we need first all queries ${}^{\mathbb{L}*}_{x,y}Q$ that may give rise to non-null queries. According to Th. 2 these are those that have at least one direct answer, and those whose projection on x has a non-empty kernel. To compute the set of queries with at least one direct answer, say $\mathcal{Z}$, we can implement Eq. 2, by computing all intersections of the nodes of the QSG for $\mathbb{S}*$ with the nodes of the QSG for $\mathbb{D}$. Then, we must

update $\mathcal{Z}$ with the queries having non-empty kernels. The queries of the form ${}^{\mathbb{R}*}_{x,y}Q_1 \circ {}^{\mathbb{C}}_{y}Q_2$ that have ${}_x\{A(x)\}$ in their kernel due to $A \sqsubseteq \exists R.C$, are in the set

$$\begin{aligned}\mathbb{E}^{A\sqsubseteq\exists R.C}_{x,y} = &\{\{S(x,y), D(y)\}_{x,y} \mid A \sqsubseteq \exists R.C,\, \mathcal{T} \models R \sqsubseteq S \text{ and } \mathcal{T} \models C \sqsubseteq D\} \cup \\ &\{\{S(y,x), D(y)\}_{x,y} \mid A \sqsubseteq \exists R^-.C,\, \mathcal{T} \models R \sqsubseteq S \text{ and } \mathcal{T} \models C \sqsubseteq D\}\end{aligned}$$

As in the case of $\mathbb{D}$ and $\mathbb{C}$, we can obtain the set $\mathbb{F}^{A\sqsubseteq\exists R.C}_{x,y}$ of the equivalence classes of the queries in ${}^{\mathbb{L}}_{x,y}Q$ that have the same answers and have ${}_x\{A(x)\}$ in their kernel due to axiom $A \sqsubseteq \exists R.C$, as the node set of the QSG for $\mathbb{E}^{A\sqsubseteq\exists R.C}_{x,y}$. Note that since we are interested in the kernels, we must compute the QSG including the null equivalence class. Because the kernel does not contribute any answers (since the head is $\langle x, y\rangle$), those of these queries that have at least one answer should already be represented by an equivalence class in $\mathcal{Z}$. However, that equivalence class may represent a more general query, with additional conjuncts, which has therefore an empty or different kernel. Thus, we need to enrich $\mathcal{Z}$ with all the queries in the several sets $\mathbb{F}^{A\sqsubseteq\exists R.C}_{x,y}$. Now, to use Eq. 4, we need also to know all queries ${}^{\mathbb{L}*}_{x,y}Q|_x$ that may give rise to non-null queries. Again, these are those that have at least one direct answer, and those that have a non-empty kernel. To compute them, we have to take the projections on x of all queries in $\mathcal{Z}$, and as before, extend this set with the projections of the queries in each $\mathbb{F}^{A\sqsubseteq\exists R.C}_{x,y}$ on x, by including however also any answers to the kernel.

6 Evaluation

We implemented the QSG computation for DL-Lite^-_R KBs in Java, and we applied it on the DBPedia KB (2016-10 version), which contains 785 concepts, 1,111 properties (in the dbo namespace), 5,150,432 concept and 24,431,982 role assertions. To test scalability we created several subsets of the KB by keeping only the concepts and roles (and the relevant assertions) that, viewed as atomic queries, had certain minimum numbers of answers (ranging from 1 to 10^6), and tried to compute the QSG for $\mathcal{Q}^{\langle 1,k\rangle *}$, $k = 1, 2, 3$. The number of nodes of the computed QSGs and computation time, as a function of the number of kept concepts and roles are shown in Figure 1. Using a machine with 64GB RAM we were able to compute $\mathcal{Q}^{\langle 1,1\rangle *}$ and $\mathcal{Q}^{\langle 1,2\rangle *}$ for the entire DBPedia, but for $\mathcal{Q}^{\langle 1,3\rangle *}$ memory was enough only for small subsets. We note that the size of the QSG, after a short superlinear growth at the beginning, grows linearly or sublinearly in the size of the ontology. Thus, in practice, the combinatorial explosion seems to occur only when we increase the size of the query but not the size of the ontology. (In computing $\mathsf{rewr}({}_x\{C(x)\})$ we ignored axioms of the form $\exists R^- \sqsubseteq C$. Although theoretically incorrect, it turned out to be more reasonable because in DBPedia such axioms are used as domain and range constraints; if they participate in the reasoning process they propagate a large number of inconsistencies.)

An example path in the graph for $\mathcal{Q}^{\langle 1,3\rangle *}$ is $Q_0 = {}_{x_0}\{\mathsf{producer}(x_0, x_1)\}$ (157,591 answers), $Q_1 = Q_0 \circ {}_{x_1}\{\mathsf{Agent}(x_1)\}$ (135,354), $Q_2 = Q_1 \circ {}_{x_1}\{\mathsf{Person}(x_1)\}$ (119,578), $Q_3 = Q_2 \circ {}_{x_0}\{\mathsf{Work}(x_0)\}$ (119,577), $Q_4 = Q_3 \circ {}_{x_1}\{\mathsf{occupation}(x_1, x_2)\}$

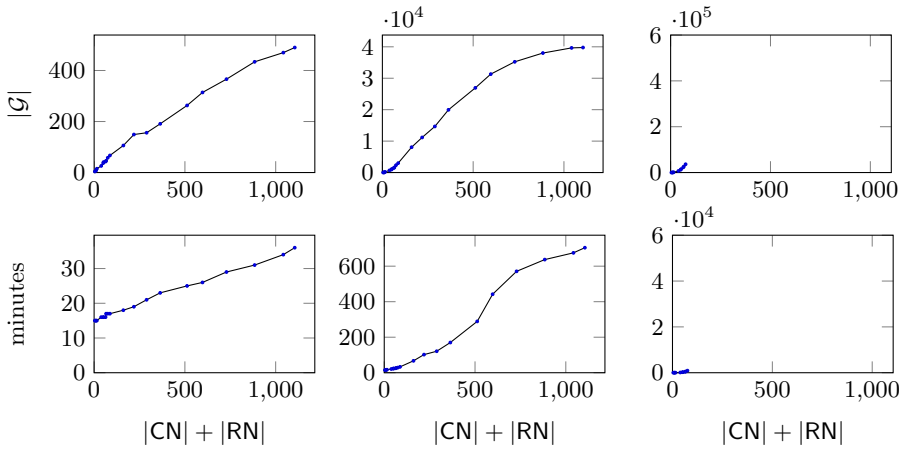

Fig. 1: QSG size and computation time for $\mathcal{Q}^{\langle 1,1\rangle *}$, $\mathcal{Q}^{\langle 1,2\rangle *}$, $\mathcal{Q}^{\langle 1,3\rangle *}$, respectively.

(81,320), $Q_5 = Q_4 \circ_{x_1} \{\mathsf{Athlete}(x_1)\}$ (11), $Q_6 = Q_5 \circ_{x_2} \{\mathsf{PersonFunction}(x_2)\}$ (10), $Q_7 = Q_6 \circ_{x_0} \{\mathsf{Album}(x_0), \mathsf{MusicalWork}(x_0)\}$ (1). Each edge refines the query, and we discover that from the 157,591 producer assertions there is only one producer that is an athlete and has produced a music album. The graph revealed also unbalances on the topic coverage. E.g. from the about 1,243,400 persons, 360,941 were athletes, 318,392 athletic team members, and 18,622 soccer managers.

7 Conclusion

We defined a theoretical framework for modeling epistemic equivalence and ordering relations between queries of a DL KB, arranging them in a graph, and discussed how to compute such a graph for DL-Lite$_R^-$ KBs. The experimental results showed that the computation of restricted versions of the QSG might be possible even for relatively big datasets. Given the complexity and memory needs of the computation, future work will target improving the prototype implementation or approximative computation. Efficient computation for more expressive DL dialects, starting from the full DL-Lite$_R$, should also be investigated. We see several applications of our framework, from construction of balanced datasets for machine learning, to semantic clustering and autocomplete suggestions.

8 Acknowledgements

We acknowledge support of this work by the project 'Apollonis: Greek Infrastructure for Digital Arts, Humanities and Language Research and Innovation' (MIS 5002738) which is implemented under the Action 'Reinforcement of the Research and Innovation Infrastructure', funded by the Operational Programme 'Competitiveness, Entrepreneurship and Innovation' (NSRF 2014-2020) and co-financed by Greece and the European Union (European Regional Development Fund).

References

1. The on-line encyclopedia of integer sequences. Number of unlabeled rooted trees with n nodes (or connected functions with a fixed point)., `http://oeis.org/A000081`
2. Arenas, M., Grau, B.C., Kharlamov, E., Marciuska, S., Zheleznyakov, D.: Faceted search over RDF-based knowledge graphs. J. Web Semant. **37-38**, 55–74 (2016)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
4. Bienvenu, M., Lutz, C., Wolter, F.: Query containment in description logics reconsidered. In: KR. AAAI Press (2012)
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. Reasoning **39**(3), 385–429 (2007)
6. Chortaras, A., Trivela, D., Stamou, G.B.: Optimized query rewriting for OWL 2 QL. In: CADE. Lecture Notes in Computer Science, vol. 6803, pp. 192–206. Springer (2011)
7. Ganter, B., Wille, R.: Formal concept analysis - mathematical foundations. Springer (1999)
8. Gottlob, G., Fermüller, C.G.: Removing redundancy from a clause. Artif. Intell. **61**(2), 263–289 (1993)
9. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intell. **14**(2-3), 189–216 (2002)
10. Nienhuys-Cheng, S., de Wolf, R. (eds.): Foundations of Inductive Logic Programming, Lecture Notes in Computer Science, vol. 1228. Springer (1997)
11. Trivela, D., Stoilos, G., Chortaras, A., Stamou, G.B.: Optimising resolution-based rewriting algorithms for OWL ontologies. J. Web Semant. **33**, 30–49 (2015)
12. Tunkelang, D.: Faceted Search. Synthesis Lectures on Information Concepts, Retrieval, and Services, Morgan & Claypool Publishers (2009)