A Framework for Exploratory Query Answering with Ontologies

Medina Andreşel[1], Yazmín Ibáñez-García[2], and Magdalena Ortiz[1]

Institute of Logic and Computation, TU Wien, Austria
{andresel, ortiz}@kr.tuwien.ac.at
School of Computer Science and Informatics, Cardiff University, Cardiff, UK
ibanezgarciay@cardiff.ac.uk

1 Introduction

In the ontology-based data access paradigm, ontologies act as an interface to data sources, facilitating query formulation by providing a vocabulary closer to the users' understanding. However, accessing data in large and unfamiliar sources can still be hard. In practice, users may find that the formulated queries have too many answers to be informative, or at the other extreme, they have very few or no answers at all. Refining queries can be difficult for users who may not know how to formulate their information needs according to the data. Moreover, changes in the query may have no effect on the answers, or dramatically affect them in unexpected ways. Manual exploration via iterative query evaluation can be computationally very costly if each minor query variation is evaluated independently and from scratch. Towards addressing this problem, we propose a framework for exploring datasets using ontology-mediated queries (OMQs). Rather than formulating a precise conjunctive query (CQ), the user writes a *template* CQ where parts may be marked to indicate that they may be relaxed or constrained during the exploration. From such a template we build what we call a *query space*: a set of OMQs that are ordered according to their specificity (or generality). The OMQs in the space are obtained from the template using a fixed set of reformulation rules that modify CQs to be more general or more specific w.r.t. $\mathcal{O}$ and a given dataset $\mathcal{A}$. Navigating the query space from one query to a more general (or specific one) allows then users to explore the dataset. The reformulation rules were proposed in our previous work [1], but here their application is restricted by the template and yields a finite space of queries that enables data exploration by choosing among the automatically generated reformulations those which modify the answers in a minimal way.

Our framework can be realized in Datalog. We describe how a query space can be generated using Datalog rules. The Datalog program is evaluated over the data and produces a structure (i.e., compilation) that contains all OMQs in the space and their answers, and which can be used to navigate between

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-2663/paper-3.pdf

reformulations and compute answers to individual queries on-the-fly. As a proof-of-concept, we implement the approach and test its potential for exploring data in DBpedia, using the VLog Datalog reasoner [10][1].

2 Preliminaries

We assume that the reader has basic knowledge about Description Logics (DLs)[5]. A *DL ontology* $\mathcal{O}$ is simply a TBox expressed in a specific DL. We will focus on extensions of *DL-Lite*$_\mathcal{R}$ [8] with *complex role inclusions*.

Let us assume an alphabet consisting of countable infinite sets $\mathbf{C}$, $\mathbf{R}$, $\mathbf{I}$ of respectively *concept*, *role*, and *individual* names. Further, we assume the set of role names is the union of two disjoint sets of *simple* and *non-simple* role names. A *DL-Lite*$^{\mathcal{H},\mathcal{R}}$ TBox $\mathcal{T}$ is a finite set of axioms taking the following forms:

$$A \sqsubseteq A', \quad A \sqsubseteq \exists p, \quad \exists p \sqsubseteq A \quad A \sqsubseteq \exists p.A' \quad p \sqsubseteq s, \quad p^- \sqsubseteq s \quad r \cdot s \sqsubseteq p,$$

where $A, A' \in \mathbf{C}$ and $p, r, s \in \mathbf{R}$, and such that: (i) for every $r \cdot s \sqsubseteq p \in \mathcal{T}$, s is simple and p is non-simple; (ii) if $p \sqsubseteq s \in \mathcal{T}$ or $p^- \sqsubseteq s \in \mathcal{T}$, and s is a simple role, then p is also simple. Axioms of the form $r \cdot s \sqsubseteq p$ are called complex role inclusions (CRI). A *DL-Lite*$^{\mathcal{HR}}_{\mathsf{rec\text{-}safe}}$ TBox is a *DL-Lite*$^{\mathcal{H},\mathcal{R}}$ TBox where all the CRIs are of the form $r \cdot s \sqsubseteq r$, and for every such CRI there is no axiom of the form $B \sqsubseteq \exists s$. A *DL-Lite*$_\mathcal{R}$ TBox is a *DL-Lite*$^{\mathcal{H},\mathcal{R}}$ TBox without CRIs. We use ABoxes to formalize datasets. An *ABox* is a finite set of assertions of the forms $A(a)$, and $p(a, b)$, with $a, b \in \mathbf{I}$, $A \in \mathbf{C}$, and $p \in \mathbf{R}$.

The semantics of DL is defined in terms of interpretations. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a nonempty *domain* $\Delta^\mathcal{I}$, and *interpretation function* $\cdot^\mathcal{I}$; $\mathcal{I}$ is a model of $\mathcal{O}$ ($\mathcal{I} \models \mathcal{O}$), if $\mathcal{I}$ satisfies every axiom in $\mathcal{O}$. An interpretation $\mathcal{I}$ *satisfies* an ABox $\mathcal{A}$, if $a^\mathcal{I} \in A^\mathcal{I}$ for all $A(a) \in \mathcal{A}$ and $(a^\mathcal{I}, b^\mathcal{I}) \in p^\mathcal{I}$, for all $p(a, b) \in \mathcal{A}$. Given a DL ontology $\mathcal{O}$ and an ABox $\mathcal{A}$, we denote by $\mathcal{I}_{\mathcal{O},\mathcal{A}}$ *the canonical model of* $(\mathcal{O}, \mathcal{A})$, which models $(\mathcal{O}, \mathcal{A})$ and can be homomorphically maped to any other model of $(\mathcal{O}, \mathcal{A})$. For *DL-Lite*$_\mathcal{R}$ ontologies, if $(\mathcal{O}, \mathcal{A})$ has a model, then it has a canonical model [8].

Ontology-mediated Querying. A *conjunctive query* q is a first order formula with free variables $\boldsymbol{x}$ that takes the form $\exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$, with φ a conjunction of *atoms* of the form $A(x)$, $r(x, y)$, and $x = a$, where $A \in \mathbf{C}$, $r \in \mathbf{R}$, x, y variables from $\boldsymbol{x} \cup \boldsymbol{y}$ and a is an individual. A *term* is either an individual from $\mathbf{I}$ or a variable. The free variables in a query are called the *answer variables*. We will write $q(\boldsymbol{x})$ to make explicit reference to the answer variables of q, and $\varphi(\boldsymbol{x}, \boldsymbol{y})$ to denote the set of atoms in q. The *arity of* $q(\boldsymbol{x})$ is defined as the length of $\boldsymbol{x}$, denoted $|\boldsymbol{x}|$.

Let $\mathcal{I}$ be an interpretation and $q(\boldsymbol{x})$ a CQ. A tuple $\boldsymbol{a}$ from $\Delta^\mathcal{I}$ of length $|\boldsymbol{x}|$ is an *answer to* q *in* $\mathcal{I}$ if there is a map π from the terms in q to $\Delta^\mathcal{I}$ such that (i) $\pi(\boldsymbol{x}) = \boldsymbol{a}$, (ii) $\pi(b) = b^\mathcal{I}$ for each individual b, (iii) $\mathcal{I} \models P(\pi(\boldsymbol{z}))$ for each $P(\boldsymbol{z})$

[1] Implementation, evaluation and proofs at https://github.com/medinaandresel/DL2020

in q, and (iv) $\pi(t) = \pi(t')$ for each $t = t'$ in q. We use $ans(q, \mathcal{I})$ to denote the set of all answers to q in $\mathcal{I}$.

An ontology mediated query (OMQ) is a pair $(q(\boldsymbol{x}), \mathcal{O})$ with $\mathcal{O}$ an ontology and $q(\boldsymbol{x})$ a CQ. Given $\mathcal{A}$, a tuple of individuals $\boldsymbol{a}$ from $\mathcal{A}$ with $|\boldsymbol{a}| = |\boldsymbol{x}|$ is a *certain answer* of $(q(\boldsymbol{x}), \mathcal{O})$ over $\mathcal{A}$ if $\boldsymbol{a} \in ans(q, \mathcal{I})$ for each model $\mathcal{I}$ of $\mathcal{O}$ satisfying $\mathcal{A}$, and we use $ans(q, \mathcal{O}, \mathcal{A})$ to denote all such tuples. The *OMQ answering problem* is: Given $(\mathcal{O}, q(\boldsymbol{x}))$, a dataset $\mathcal{A}$, and $\boldsymbol{a}$ a tuple of individuals of length equal to $|\boldsymbol{x}|$, is $\boldsymbol{a}$ a certain answers of $(q(\boldsymbol{x}), \mathcal{O})$ over $\mathcal{A}$. For $(q_1, \mathcal{O})$ and $(q_2, \mathcal{O})$ of the same arity, we write $q_1 \subseteq_{\mathcal{O},\mathcal{A}} q_2$ if $ans(q_1, \mathcal{O}, \mathcal{A}) \subseteq ans(q_2, \mathcal{O}, \mathcal{A})$. In this case, we say that q_1 is a *specialization* of q_2 w.r.t. $\mathcal{O}, \mathcal{A}$ and, conversely, q_2 a *generalization of* q_1 w.r.t. $(\mathcal{O}, \mathcal{A})$.

3 Framework for Exploratory Querying

We start by formalizing a suitable notion of query space.

Definition 1. *An exploratory* query space *for a given ABox $\mathcal{A}$ is a preordered set $\mathcal{Q} = \langle Q, \preceq \rangle$ of queries mediated by an ontology $\mathcal{O}$, such that $(q_1, \mathcal{O}) \preceq (q_2, \mathcal{O})$ implies $q_1 \subseteq_{(\mathcal{O},\mathcal{A})} q_2$. We will write $q_1 \preceq q_2$, whenever $\mathcal{O}$ is clear from the context.*

We now introduce *query templates*, which is the starting point to build exploratory query spaces. Syntactically, they are CQs whose atoms may be marked if we want to consider more specialized($\cdot^{\mathsf{s}}$) or generalized($\cdot^{\mathsf{g}}$) versions of them.

Definition 2 (Query Templates). *A* query template *$\Psi[\boldsymbol{x}]$ is an expression $\exists \boldsymbol{y}.\tau_1 \wedge \cdots \wedge \tau_n$, where each τ_i is an atom of the form:*

$$A(x) \mid r(x, y) \mid x = a \mid A^{\mathsf{s}}(x) \mid A^{\mathsf{g}}(x) \mid r^{\mathsf{s}}(x, y) \mid r^{\mathsf{g}}(x, y) \mid x = a^{\mathsf{s}} \mid x = a^{\mathsf{g}},$$

where $x, y \in \boldsymbol{x} \cup \boldsymbol{y}$, $\boldsymbol{x}$ are the answer variables*, $A \in \mathbf{C}$ or $\top$, $r \in \mathbf{R}$ and $a \in \mathbf{I}$.*

Example 1. We can succintly describe CQs that retrieve (possibly special types of) events in Rhodes or in more general locations, using template:

$$\Psi[x] = \exists z\ Event^{\mathsf{s}}(x) \wedge hasLoc^{\mathsf{g}}(x, z) \wedge z = \mathsf{Rhodes}^{\mathsf{g}}.$$

Given some Ψ as starting point, we will use *reformulation rules* to build a set of related CQs. To guarantee that these queries can be ordered according to their specificity (or generality), these rules will be guided by a set of *reformulation axioms*. In general, $\mathcal{O}$ can be used to guide the rules application. However, to take also the data into account, we allow other sets $\mathcal{R}$ of reformulation axioms as well. For example, $\mathcal{R}$ can be a subset of $\mathcal{O}$ that the user considers *relevant*. It may contain assertions from $\mathcal{A}$, or other axioms implied by $\mathcal{O}$ and $\mathcal{A}$, enabling *data-driven* query reformulations in the style of [1]. The *reformulation rules* are presented in Table 1. We use '_' as a placeholder for a fresh variable; $\tilde{y}$ denotes the condition $(y \notin vars(\Psi') \cup \boldsymbol{x}) \vee (y = \_)$; $\hat{y}$ denotes that y is not an answer variable in the template; and $r^{\times}$ stands for any of r^{s}, r^{g}, or r.

(R1)	If $B \sqsubseteq A \in \mathcal{R}$	then: $\Psi' \wedge A^{\mathsf{s}}(x) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge B^{\mathsf{s}}(x)$	$\Psi' \wedge B^{\mathsf{g}}(x) \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge A^{\mathsf{g}}(x)$
(R2)	If $A \sqsubseteq \exists r \in \mathcal{R}$	then: $\Psi' \wedge r^{\mathsf{s}}(x, \tilde{y}) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge A^{\mathsf{s}}(x)$	$\Psi' \wedge A^{\mathsf{g}}(x) \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge r^{\mathsf{g}}(x, \_)$
(R3)	If $\exists r \sqsubseteq A \in \mathcal{R}$	then: $\Psi' \wedge A^{\mathsf{s}}(x) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge r^{\mathsf{s}}(x, \_)$	$\Psi' \wedge r^{\mathsf{g}}(x, \tilde{y}) \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge A^{\mathsf{g}}(x)$
(R4)	If $r \sqsubseteq p \in \mathcal{R}$	then: $\Psi' \wedge p^{\mathsf{s}}(x, y) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge r^{\mathsf{s}}(x, y)$	$\Psi' \wedge r^{\mathsf{g}}(x, y) \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge p^{\mathsf{g}}(x, y)$
(R5)	If $s^- \sqsubseteq p \in \mathcal{R}$	then: $\Psi' \wedge p^{\mathsf{s}}(x, y) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge s^{\mathsf{s}}(y, x)$	$\Psi' \wedge s^{\mathsf{g}}(x, y) \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge p^{\mathsf{g}}(y, x)$
(R6)	If $\{B \sqsubseteq \exists s.A,$ $r \cdot s \sqsubseteq r\} \subseteq \mathcal{R}$	then: $\Psi' \wedge r^{\times}(x, \hat{y}) \wedge A^{\mathsf{s}}(\hat{y}) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge r^{\times}(x, y) \wedge B^{\mathsf{s}}(y)$	$\Psi' \wedge r^{\times}(x, \hat{y}) \wedge B^{\mathsf{g}}(\hat{y}) \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge r^{\times}(x, y) \wedge A^{\mathsf{g}}(y)$
(R7)	If $A(a) \in \mathcal{R}$	then: $\Psi' \wedge A^{\mathsf{s}}(x) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge x = a^{\mathsf{s}}$	$\Psi' \wedge x = a^{\mathsf{g}} \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge A^{\mathsf{g}}(x)$
(R8)	If $r(a, b) \in \mathcal{R}$	then: $\Psi' \wedge r^{\mathsf{s}}(x, \tilde{y}) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge x = a^{\mathsf{s}}$	$\Psi' \wedge x = a^{\mathsf{g}} \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge r^{\mathsf{g}}(x, \_)$
		$\Psi' \wedge r^{\mathsf{s}}(\tilde{x}, y) \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge y = b^{\mathsf{s}}$	$\Psi' \wedge x = b^{\mathsf{g}} \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge r^{\mathsf{g}}(\_, x)$
(R9)	If $\{s(a, b),$ $r \cdot s \sqsubseteq r\} \subseteq \mathcal{R}$	then: $\Psi' \wedge r^{\mathsf{s}}(x, \hat{y}) \wedge \hat{y} = b \overset{\mathbf{s}}{\rightsquigarrow} \Psi' \wedge r^{\mathsf{s}}(x, y) \wedge y = a^{\mathsf{s}}$	
		$\Psi' \wedge r^{\mathsf{g}}(x, \hat{y}) \wedge \hat{y} = a^{\mathsf{g}} \overset{\mathbf{g}}{\rightsquigarrow} \Psi' \wedge r^{\mathsf{g}}(x, y) \wedge y = b^{\mathsf{g}}$	

Table 1. Rules to derive CQs from query template $\Psi[\boldsymbol{x}]$.

.

Definition 3. *A* reformulation axiom *is either an ABox assertion or a TBox axiom in DL-Lite$^{\mathcal{HR}}_{\mathsf{rec\text{-}safe}}$. Given a template Ψ and a set of reformulation axioms $\mathcal{R}$, we say that Ψ'* is derivable from Ψ using $\mathcal{R}$ *if there is a sequence of reformulations $\Psi \overset{*}{\rightsquigarrow} \Psi_1 \ldots \overset{*}{\rightsquigarrow} \Psi_n = \Psi'$, with $n \geq 1$ and $* \in \{\mathsf{s}, \mathsf{g}\}$, using the rules in Table 1. We will use $\Psi \rightsquigarrow_{\mathcal{R}} \Psi'$, to indicate that Ψ' is derivable from Ψ w.r.t. $\mathcal{R}$, and we will write $\mathsf{cq}(\Psi)$ to denote the CQ obtained from Ψ by removing all the s and g labels. Then we define:*

- *$Q^{\mathcal{R}}_{\Psi}$ is the set of all CQs $\{q(\boldsymbol{x}) \mid \Psi[\boldsymbol{x}] \rightsquigarrow_{\mathcal{R}} \Psi'[\boldsymbol{x}], \mathsf{cq}(\Psi'[\boldsymbol{x}]) = q(\boldsymbol{x})\}$.*
- *For each pair $q_1, q_2 \in Q^{\mathcal{R}}_{\Psi}$, we set $q_1 \preceq^{\mathcal{R}}_{\Psi} q_2$ if $q_1 = q_2$, or there are templates Ψ_1 and Ψ_2 such that $\mathsf{cq}(\Psi_i) = q_i$ and either $\Psi_2 \overset{\mathsf{s}}{\rightsquigarrow}_{\mathcal{R}} \Psi_1$, or $\Psi_1 \overset{\mathsf{g}}{\rightsquigarrow}_{\mathcal{R}} \Psi_2$.*

Note that, following [1], we are allowing CRIs in our reformulation axioms to be able to reformulate our queries along dimensions that are not captured by the subclass or subrole relations. For example, a *location dimension* that has different granularity levels connected by a 'part-of' rather than by a 'subclass-of' relation. For example, using axioms like $City \sqsubseteq \exists partOf.Country$ and the CRI $hasLoc \cdot partOf \sqsubseteq hasLoc$, we can generalize a query asking for events in a city to one that asks for events in a country instead. We call this particular kind of generalization operation a 'roll up' and its specialization counterpart 'drill down' (see rules **R6** and **R9**).

Rules in Table 1 specialize $A^{\mathsf{s}}(x)$ as either $B(x)$ using (**R1**) or $r(x, \_)$ using (**R3**), which are more specific than A, or as $A(a)$, with a a known instance of concept A using (**R7**). Atoms $p^{\mathsf{s}}(x, y)$ are specialized as $r(x, y)$ with r a role more specific than r using (**R4**) or (**R5**), or $B(x)$ (or $B(y)$), with B a concept that is more specific than $\exists r$ (or $\exists r^-$) with rule (**R2**). The rules work similary for the generalizing counterparts.

Example 2. Let us consider again the template $\Psi[x]$ from Example 1, and the set of reformulation axioms: $\mathcal{R}_1 = \{hasLoc \cdot partOf \sqsubseteq hasLoc, partOf(\mathsf{Rhodes}, \mathsf{Greece})\}$.

Rule (**R9**) is applicable for Ψ and $\mathcal{R}_1$ (since z is not an answer variable):

$$Event^{\mathsf{s}}(x) \wedge hasLoc^{\mathsf{g}}(x,z) \wedge z = \mathsf{Rhodes}^{\mathsf{g}} \overset{\mathbf{g}}{\rightsquigarrow} Event^{\mathsf{s}}(x) \wedge hasLoc^{\mathsf{g}}(x,z) \wedge z = \mathsf{Greece}^{\mathsf{g}}$$

By removing the special markers, the following query is part of $Q_\Psi^{\mathcal{R}_1}$:

$$q_1(x) : \exists z\ Event(x) \wedge hasLoc(x,z) \wedge z = \mathsf{Greece}.$$

Thus, $\mathsf{cq}(\Psi) \preceq_\Psi^{\mathcal{R}_1} q_1$. The semantics of Ψ change if we consider, for example, the set of reformulation axioms $\mathcal{R}_2 = \{Conference \sqsubseteq Event\}$. Using rule (**R1**) we can capture the query $q_2(x) : \exists z\ Conference(x) \wedge hasLoc(x,z) \wedge z = \mathsf{Rhodes}$.

Let $\mathcal{O}$ be written in a language that enjoys the canonical model property. Then we call a set $\mathcal{R}$ of reformulation axioms *compatible with* $(\mathcal{O}, \mathcal{A})$ if $\mathcal{I}_{\mathcal{O},\mathcal{A}} \models \mathcal{R}$. Compatibility of $\mathcal{R}$ ensures that we obtain an exploratory query space.

Lemma 1. *Let Ψ be a template, $\mathcal{R}$ a set of reformulation axioms, and $\mathcal{O}$ an ontology in a language that enjoys the canonical model. If $\mathcal{A}$ is such that $\mathcal{R}$ is compatible with $(\mathcal{O}, \mathcal{A})$, then $\langle Q_\Psi^{\mathcal{R}}, \preceq_\Psi^{\mathcal{R}} \rangle$ is exploratory for $\mathcal{A}$.*

Example 3. Let $\mathcal{O} = \{Workshop \sqsubseteq Conference,\ Conference \sqsubseteq Event\}$ and $\mathcal{A} =$

$$\{Workshop(\mathsf{DL}_{2020}),\ hasLoc(\mathsf{DL}_{2020}, \mathsf{Rhodes}),$$
$$hasLoc(\mathsf{DL}_{2020}, \mathsf{Greece}),\ partOf(\mathsf{Rhodes}, \mathsf{Greece})\}.$$

$\{hasLoc \cdot partOf \sqsubseteq hasLoc\}$ is compatible with $(\mathcal{O}, \mathcal{A})$ and the rest of $\mathcal{R}_1$ and $\mathcal{R}_2$ are in $(\mathcal{O}, \mathcal{A})$, so both $(\mathcal{Q}_\Psi^{\mathcal{R}_1}, \preceq_\Psi^{\mathcal{R}_1})$ and $(\mathcal{Q}_\Psi^{\mathcal{R}_2}, \preceq_\Psi^{\mathcal{R}_2})$ are exploratory for $(\mathcal{O}, \mathcal{A})$.

If $\mathcal{R}$ is not a subset of $\mathcal{O} \cup \mathcal{A}$, we need to test its compatibility. When $\mathcal{O}$ is in $DL\text{-}Lite_{\mathsf{rec\text{-}safe}}^{\mathcal{HR}}$, such test can be done in PTIME [2].

3.1 Exploring Query Spaces

The order in a query space enables users to explore the dataset by navigating from one query to a more general (or specific one) on demand. However, obtaining *all* reformulations may not be satisfactory if there are too many of them, and we may need to identify the most relevant ones. For example, assuming that a dataset contains in addition to the assertions in Example 3: $Conference(KR_{2020})$, $hasLoc(KR_{2020}, Rhodes)$, a natural way to filter out the answers to $q(x)$: $Event(x) \wedge hasLoc(x,z) \wedge z = Rhodes$ is to navigate to $q_n(x) : Conference(x) \wedge hasLoc(x,z) \wedge z = Rhodes$. This reformulation however does not change the set of answers – $\{KR_{2020}, DL_{2020}\}$, and it may be more interesting to directly specialize to $q_s(x) : Workshop \wedge hasLoc(x,z) \wedge z = Rhodes$, which drops KR_{2020} as answer. By identifying such queries in the space, the data can be explored in a more effective step-by-step fashion.

Definition 4. *For queries $q_1 \neq q_2$ in $\mathcal{Q} = (Q, \preceq)$ such that $q_1 \preceq q_2$, we say that q_1 is a* neutral *specialization of q_2, written $q_1 \simeq q_2$, if $ans(q_1, \mathcal{O}, \mathcal{A}) = ans(q_2, \mathcal{O}, \mathcal{A})$; it is a* strict *specialization of q_2, written $q_1 \prec q_2$, if $ans(q_1, \mathcal{O}, \mathcal{A}) \subsetneq ans(q_2, \mathcal{O}, \mathcal{A})$. Conversely, q_2 is a* neutral, *respectively* strict, *generalization of q_1. Further,*

- *If* $q_1 \simeq q_2$*, we say that* q_1 *is a* maximal *neutral specialization of* q_2 *if for each* $q' \in Q$ *such that* $q' \preceq q_1$*, it holds that* $q' \prec q_2$*. Conversely,* q_2 *is a* maximal *neutral generalization of* q_1 *if for each* $q' \in Q$ *such that* $q_2 \preceq q'$*, we have* $q_1 \prec q'$*.*
- q_1 *is a* minimal *strict specialization of* q_2*, if* $q_1 \prec q_2$ *and for each* $q' \in Q$ *such that* $q_1 \preceq q' \preceq q_2$ *we have* $q' \simeq q_2$*. Conversely,* q_2 *is a* minimal *strict generalization of* q_1 *if for each* $q' \in Q$ *such that* $q_1 \preceq q' \preceq q_2$ *we have* $q_1 \simeq q'$*.*

We denote by $\mathsf{maxNeu}^{\mathsf{s}}_{\mathcal{A}}(q, \mathcal{Q})$ and $\mathsf{minStr}^{\mathsf{s}}_{\mathcal{A}}(q, \mathcal{Q})$ the set of all maximal neutral and of all minimal strict specializations, respectively, and similarly for generalizations. Maximal neutral reformulations are useful, since they allow us to easily identify the minimal strict reformulations desirable for navigating the space.

Observation 1 *Let* $\mathcal{Q} = \langle Q, \preceq \rangle$ *be an exploratory query space for* $\mathcal{A}$ *mediated by* $\mathcal{O}$*, and let* $\preceq^c$ *extend* $\preceq$ *so that* $q_1 \preceq^c q_2$ *whenever* $q_1 \simeq q_2$*. For* $q \in Q$*, we have:*

- $q_1 \in \mathsf{minStr}^{\mathsf{s}}_{\mathcal{A}}(q, \mathcal{Q})$ *iff* $q_1 \not\simeq^c q$*, and* $q_1 \preceq^c q_2$ *for some* $q_2 \in \mathsf{maxNeu}^{\mathsf{s}}_{\mathcal{A}}(q, \mathcal{Q})$*.*
- $q_1 \in \mathsf{minStr}^{\mathsf{g}}_{\mathcal{A}}(q, \mathcal{Q})$ *iff* $q_1 \not\preceq^c q$*, and* $q_2 \preceq^c q_1$ *for some* $q_2 \in \mathsf{maxNeu}^{\mathsf{g}}_{\mathcal{A}}(q, \mathcal{Q})$*.*

Not surprisingly, identifying minimal strict reformulations is not feasible in polynomial time in general. It is at least as hard as the following following decision problem: given q and q' in $\mathcal{Q}$, verify if q' is a maximal neutral specialization (or generalization) of q in $\mathcal{Q}$. Specifically, coNP-hardness for maximal neutral specializations can be proved even for monadic tree-shaped CQs mediated by the empty ontology, by reducing the problem of verifying if a given $\mathcal{EL}$ concept is the most specific concept for individuals $a_1, \ldots, a_n$, given ABox $\mathcal{A}$ [13]. However, we show in the next section that using an offline-phase that compiles the answers to the queries in the space, computing such relevant reformulations can still be realized efficiently using Datalog with stratified negation (since the size of the compilation is comparably smaller than the size of the dataset).

4 Capturing Query Spaces in Datalog

In this section, we describe a way to realize our exploratory framework using Datalog. Before proceeding, we recall the syntax and semantics of *Datalog programs with stratified negation* [3].

Datalog with Equality and Stratified Negation. Let $\mathbf{P}$, $\mathbf{V}$ and $\mathbf{K}$ be countable infinite sets of predicates, variables and constants. Variables and constants are *terms*. An atom β is either (i) $\mathsf{p}(\boldsymbol{v})$ with $\boldsymbol{v}$ a tuple of terms of the same arity as p, or (ii) $v = v'$ for terms v, v'. A Datalog rule ρ has the form:

$$\mathsf{p}(\boldsymbol{v}) \leftarrow \beta_1, \ldots, \beta_k, \neg\beta_{k+1}, \ldots, \neg\beta_m$$

where $m \geq k$, $\mathsf{p}(\boldsymbol{v})$ is called the *head of* ρ and denoted by $head(\rho)$, while the set of atoms $\{\beta_1, \ldots, \beta_m\}$ is the *body of* ρ; $body^+(\rho)$ denotes the positive atoms and

$body^-(\rho)$ the negative ones. As usual, all variables in $head(\rho)$ and in $body^-(\rho)$ must also occur in $body^+(\rho)$. A rule with empty body is called *fact* and maybe written $\mathsf{p}(\boldsymbol{v})$.

A *Datalog program* Π is a set of Datalog rules. Π is *stratified* if it can be partitioned as $\Pi_1, \ldots, \Pi_n$ such that for each p, all rules with p in the head are in the same partition Π_i, and for all atoms in the bodies of those rules, the definitions of such predicates are in some Π_j, with $j \leq i$ if the atom is positive, or $j < i$ for negative atoms.

A set of facts I *satisfies a rule* ρ if there exists a mapping $h : vars(\rho) \mapsto term(I)$ such that whenever $h(body^+(\rho)) \subseteq I$ and $h(body^-(\rho)) \not\subseteq I$, then h ($head(\rho)$) $\subseteq I$. A *model of* Π is any set of facts that satisfies each $\rho \in \Pi$. For a stratified program Π and finite set of facts D, $\Pi(D)$ denotes the minimal model of Π (which is unique and always exists) consisting of $\bigcup_{i=0}^{n} I_i$, where $I_0 = D$ and for $1 \leq i \leq n$, each I_i minimally extends I_{i-1} such that I_i is a model of Π_i.

4.1 Datalog Encoding of Template-generated Query Spaces

We now build a Datalog program that derives and evaluates all the queries in a space. In this section we fix a template $\Psi = \exists \boldsymbol{y}.\tau_1 \wedge \cdots \wedge \tau_n$ with answer variables $\boldsymbol{x} = (x_1, \ldots, x_\ell)$, as well as a set of reformulation axioms $\mathcal{R}$. Further, we assume that all non-answer variables occuring only once in Ψ have been substituted by '_', and for convenience, assume $\boldsymbol{x} \cup \boldsymbol{y} \subseteq \mathbf{V}$ (these are the only Datalog variables written in lower-case). Since we denote concepts, roles and individuals by constants, we assume for simplicity $\mathbf{C} \cup \mathbf{R} \cup \mathbf{I} \subseteq \mathbf{K}$. We also assume a constant $v_x \in \mathbf{K}$ for every variable x and a constant c_τ for each atom τ in Ψ. A special constant null acts as placeholder for '_' and as a 'filler' for irrelevant predicate positions. For each atom τ_i, we use $\boldsymbol{x_i}$ to denote (x, null) if τ_i is either $P^{\mathsf{x}}(x)$, $P^{\mathsf{x}}(x, \_)$ or $x = a^{\mathsf{x}}$; (null, x) if $\tau_i = P^{\mathsf{x}}(\_, x)$; and (x, y) if $\tau_i = P^{\mathsf{x}}(x, y)$. By convention, $\boldsymbol{U} = (U_1, \ldots, U_n)$ and $\boldsymbol{V} = (V_1, \ldots, V_n)$ are n-ary tuples of variables, and $\boldsymbol{X}$ is an ℓ-ary tuple of variables.

In Table 2, we show how Ψ and $\mathcal{R}$ are encoded: each atom τ in Ψ is mapped ($\mapsto$) to a fact in D_Ψ, and axioms in $\mathcal{R}$ to facts in $D_\mathcal{R}$. A fixed program Π_{rules} simulates the reformulation rules from Table 1. These rules derive a $\mathsf{atm_s}$ atom for each reformulation of a specializing atom, and a $\mathsf{atm_g}$ atom for each reformulation of a generalizing atom in Ψ. The final group of rules $\Pi_\mathcal{Q}$ build up the set of OMQs in the space. Each tuple in $\mathsf{refAtms}$ represents a reformulated version of each atom in the template, and allows us to build a query by putting the corresponding variables in the right positions.

We can now make precise how the query space is captured. We denote by $tr(q)$ the translation of a given CQ $q(\boldsymbol{x})$ into the signature of our Datalog program, obtained by applying to each atom in q the function tr with $tr(x = a) = \mathsf{uAtm}(a, t(x))$, $tr(A(x)) = \mathsf{uAtm}(A, t(x))$, and $tr(r(x, y)) = \mathsf{bAtm}(r, t(x), t(y))$, where for each x_i, $t(x_i) = \mathsf{null}$ if $x = \_$ and $t(x_i) = X_i$ otherwise.

Definition 5. *We let* $\Pi_\Psi = \Pi_{rules} \cup \Pi_\mathcal{Q}$ *and* $D_{\Psi,\mathcal{R}} = D_\Psi \cup D_\mathcal{R}$*, and call the pair* $\langle \Pi_\Psi, D_{\Psi,\mathcal{R}} \rangle$ *the* Datalog encoding of $(\Psi, \mathcal{R})$.

$\boldsymbol{D_{\mathcal{R}}}$:	$A \mapsto \mathsf{conc}(A)$,	$A \sqsubseteq A' \mapsto \mathsf{cISA}(A, A')$	$A(A) \mapsto \mathsf{uAssrt}(A, a)$
	$r \mapsto \mathsf{role}(r)$,	$\exists r \sqsubseteq A \mapsto \mathsf{cISA}(r, A)$	$r(a, b) \mapsto \mathsf{bAssrt}(r, a, b)$
	$a \mapsto \mathsf{const}(a)$,	$A \sqsubseteq \exists r \mapsto \mathsf{cISA}(A, r)$	$\{s(a, b), r \cdot s \sqsubseteq r\} \mapsto \mathsf{ddn}(b, a, r), \mathsf{rup}(a, b, r)$
		$s \sqsubseteq r \mapsto \mathsf{rISA}(s, r)$	$\{A \sqsubseteq \exists s.A', r \cdot s \sqsubseteq r\} \mapsto \mathsf{ddn}(A', A, r)$
		$p^- \sqsubseteq r \mapsto \mathsf{rISAinv}(p, r)$	$\{A \sqsubseteq \exists s.A', r \cdot s \sqsubseteq r\} \mapsto \mathsf{rup}(A, A', r)$

$\boldsymbol{D_{\Psi}}$:	$x \notin \boldsymbol{x} \mapsto \mathsf{nAns}(v_x)$	$A^{\mathsf{x}}(x) \mapsto \mathsf{atm_s}(c_\tau, A, v_x, \mathsf{null})$	$A(x) \mapsto \mathsf{refAt}(c_\tau, A, v_x, \mathsf{null})$
	$A \mapsto \mathsf{conc}(A)$	$r^{\mathsf{x}}(x, y) \mapsto \mathsf{atm_s}(c_\tau, r, v_x, v_y)$	$r(x, y) \mapsto \mathsf{refAt}(c_\tau, r, v_x, v_y)$
	$r \mapsto \mathsf{role}(r)$	$x = a^{\mathsf{x}} \mapsto \mathsf{atm_s}(c_\tau, a, v_x, \mathsf{null})$	$x = a \mapsto \mathsf{refAt}(c_\tau, a, v_x, \mathsf{null})$
	$a \mapsto \mathsf{const}(a)$	with $\mathsf{x} \in \{\mathsf{s}, \mathsf{g}\}$	

Encoding of the reformulation rules $\boldsymbol{\Pi_{rules}}$

(R1–3)

$$\mathsf{atm_s}(V_\tau, U, Y, Z) \leftarrow \mathsf{atm_s}(V_\tau, U', Y, Z), \mathsf{cISA}(U, U'), \mathsf{conc}(U').$$
$$\mathsf{atm_g}(V_\tau, U', Y, Z) \leftarrow \mathsf{atm_g}(V_\tau, U, Y, Z), \mathsf{cISA}(U, U'), \mathsf{conc}(U).$$
$$\mathsf{atm_s}(V_\tau, U, Y, \mathsf{null}) \leftarrow \mathsf{atm_s}(V_\tau, U', Y, \mathsf{null}), \mathsf{cISA}(U, U'), \mathsf{conc}(U), \mathsf{role}(U').$$
$$\mathsf{atm_g}(V_\tau, U', Y, \mathsf{null}) \leftarrow \mathsf{atm_g}(V_\tau, U, Y, \mathsf{null}), \mathsf{cISA}(U, U'), \mathsf{conc}(U'), \mathsf{role}(U).$$

(R4), **(R5)**

$$\mathsf{atm_s}(V_\tau, U, Y, Z) \leftarrow \mathsf{atm_s}(V_\tau, U', Y, Z), \mathsf{rISA}(U, U').$$
$$\mathsf{atm_g}(V_\tau, U', Y, Z) \leftarrow \mathsf{atm_g}(V_\tau, U, Y, Z), \mathsf{rISA}(U, U').$$
$$\mathsf{atm_s}(V_\tau, U, Y, Z) \leftarrow \mathsf{atm_s}(V_\tau, U', Z, Y), \mathsf{rISAinv}(U, U').$$
$$\mathsf{atm_g}(V_\tau, U', Y, Z) \leftarrow \mathsf{atm_g}(V_\tau, U, Z, Y), \mathsf{rISAinv}(U, U').$$

(R6), **(R9)**

$$\mathsf{atm_s}(V_\tau, U', W, \mathsf{null}) \leftarrow \mathsf{ddn}(U, U', Z), \mathsf{atm_s}(V_\tau, U, W, \mathsf{null}), W \neq \mathsf{null}, \mathsf{refAt}(V_{\tau'}, Z, Y, W), V_\tau \neq V_{\tau'}, \mathsf{nAns}(W).$$
$$\mathsf{atm_g}(V_\tau, U', W, \mathsf{null}) \leftarrow \mathsf{rup}(U, U', Z), \mathsf{atm_g}(V_\tau, U, W, \mathsf{null}), W \neq \mathsf{null}, \mathsf{refAt}(V'_\tau, Z, Y, W), V_\tau \neq V_{\tau'}, \mathsf{nAns}(W).$$

(R7), **(R8)**

$$\mathsf{atm_s}(V_\tau, V, Y, \mathsf{null}) \leftarrow \mathsf{atm_s}(V_\tau, U, Y, \mathsf{null}), \mathsf{uAssrt}(U, V).$$
$$\mathsf{atm_g}(V_\tau, U, Y, \mathsf{null}) \leftarrow \mathsf{atm_g}(V_\tau, V, Y, \mathsf{null}), \mathsf{uAssrt}(U, V).$$
$$\mathsf{atm_g}(V_\tau, U, \mathsf{null}, Y) \leftarrow \mathsf{atm_g}(V_\tau, V', Y, \mathsf{null}), \mathsf{bAssrt}(U, V, V').$$
$$\mathsf{atm_g}(V_\tau, U, Y, \mathsf{null}) \leftarrow \mathsf{atm_g}(V_\tau, V, Y, \mathsf{null}), \mathsf{bAssrt}(U, V, V').$$
$$\mathsf{atm_s}(V_\tau, V, Y, \mathsf{null}) \leftarrow \mathsf{atm_s}(V_\tau, U, Y, \mathsf{null}), \mathsf{bAssrt}(U, V, V'), Y \neq \mathsf{null}.$$
$$\mathsf{atm_s}(V_\tau, V', \mathsf{null}, Z) \leftarrow \mathsf{atm_s}(V_\tau, U, \mathsf{null}, Z), \mathsf{bAssrt}(U, V, V'), Z \neq \mathsf{null}.$$

$$\mathsf{refAx}(U', U) \leftarrow \mathsf{cISA}(U, U'). \quad \mathsf{refAx}(U', U) \leftarrow \mathsf{rISA}(U, U').$$
$$\mathsf{refAx}(U, X) \leftarrow \mathsf{uAtm}(U, X). \quad \mathsf{refAx}(U, X) \leftarrow \mathsf{bAtm}(U, X, Y).$$
$$\mathsf{refAx}(U, Y) \leftarrow \mathsf{bAtm}(U, X, Y). \quad \mathsf{refAx}(U', U) \leftarrow \mathsf{rISAinv}(U, U').$$
$$\mathsf{refAx}(U', U) \leftarrow \mathsf{rup}(U, U', Z). \quad \mathsf{refAx}(U, U') \leftarrow \mathsf{ddn}(U, U', Z)$$
$$\mathsf{refAt}(V_\tau, U, X, Y) \leftarrow \mathsf{atm_s}(V_\tau, U, X, Y). \quad \mathsf{refAt}(V_\tau, U, X, Y) \leftarrow \mathsf{atm_g}(V_\tau, U, X, Y).$$

Query space encoding $\boldsymbol{\Pi_{\mathcal{Q}}}$:

$$\mathsf{refAtms}(U_1, \ldots, U_n) \leftarrow \mathsf{refAt}(c_{\tau_1}, U_1, \boldsymbol{x_1}), \ldots, \mathsf{refAt}(c_{\tau_n}, U_n, \boldsymbol{x_n}).$$
$$\mathsf{query}_\Psi(\boldsymbol{U}, \boldsymbol{x}) \leftarrow \mathsf{refAtms}(\boldsymbol{U}), \mathsf{qAtm}_{\tau_1}(U_1, \boldsymbol{x_1}), \ldots, \mathsf{qAtm}_{\tau_n}(U_n, \boldsymbol{x_n}).$$
$$\mathsf{qAtm}_\tau(U, X, \mathsf{null}) \leftarrow \mathsf{uAtm}(U, X), \mathsf{conc}(U).$$
$$\mathsf{qAtm}_\tau(U, X, \mathsf{null}) \leftarrow U = X, \mathsf{const}(U).$$
$$\mathsf{qAtm}_\tau(U, X, Y) \leftarrow \mathsf{bAtm}(U, X, Y), \mathsf{role}(U).$$

Table 2. Datalog encoding of $\Psi[\boldsymbol{x}]$, $\mathcal{R}$ and reformulation rules

.

Π_{ref}:

$$\mathsf{str_s}(\boldsymbol{U}[U_i \mapsto V], c_{\tau_i}, V') \leftarrow \mathsf{refAx}(V, V'), \mathsf{refAt}(c_{\tau_i}, V, X, Y), \mathsf{refAt}(c_{\tau_i}, V', X', Y'), \mathsf{query}_\Psi(\boldsymbol{U}[U_i \mapsto V], \boldsymbol{X}),\ \neg\mathsf{query}_\Psi(\boldsymbol{U}[U_i \mapsto V'], \boldsymbol{X}).$$

$$\mathsf{str_g}(\boldsymbol{U}[U_i \mapsto V'], c_{\tau_i}, V) \leftarrow \mathsf{refAx}(V, V'), \mathsf{refAt}(c_{\tau_i}, V, X, Y), \mathsf{refAt}(c_{\tau_i}, V', X', Y'), \mathsf{query}_\Psi(\boldsymbol{U}[U_i \mapsto V], \boldsymbol{X}),\ \neg\mathsf{query}_\Psi(\boldsymbol{U}[U_i \mapsto V'], \boldsymbol{X}).$$

$$\mathsf{nrt_s}(\boldsymbol{U}[U_i \mapsto V], c_{\tau_i}, V') \leftarrow \mathsf{refAx}(V, V'), \mathsf{refAtms}(\boldsymbol{U}[U_i \mapsto V]), \neg\mathsf{str_s}(\boldsymbol{U}[U_i \mapsto V], c_{\tau_i}, V').$$

$$\mathsf{ntr_g}(\boldsymbol{U}[U_i \mapsto V], c_{\tau_i}, V') \leftarrow \mathsf{refAx}(V', V), \mathsf{refAtms}(\boldsymbol{U}[U_i \mapsto V]), \neg\mathsf{str_g}(\boldsymbol{U}[U_i \mapsto V], c_{\tau_i}, V').$$

$$\mathsf{nrt_x}(\boldsymbol{U}[U_i \mapsto V], c_{\tau_i}, V'') \leftarrow \mathsf{nrt_x}(\boldsymbol{U}[U_i \mapsto V], c_{\tau_i}, V'), \mathsf{nrt_x}(\boldsymbol{U}[U_i \mapsto V'], c_{\tau_i}, V'').$$

$$\mathsf{maxNtr_x}(\boldsymbol{U}[U_i \mapsto V], c_{\tau_i}, V') \leftarrow \mathsf{nrt_x}(\boldsymbol{U}, c_{\tau_i}, V'),\ \neg\,\mathsf{nrt_x}(\boldsymbol{U}, c_{\tau_i}[U_i \mapsto V], V''), V' \neq V''.$$

$$\mathsf{max_x}(\boldsymbol{U}, \boldsymbol{V}) \leftarrow \mathsf{refAtms}(\boldsymbol{U}), \mathsf{maxNtr_x}(\boldsymbol{U_1}, c_{\tau_1}, V_1), \ldots, \mathsf{maxNtr_x}(\boldsymbol{U_n}, c_{\tau_n}, V_n). \quad \text{with } \mathsf{x} \in \mathsf{s}, \mathsf{g}$$

$$\mathsf{min_s}(\boldsymbol{U}, V_1, \ldots, V_{i-1}, V', V_{i+1}, \ldots, V_n) \leftarrow \mathsf{max_s}(\boldsymbol{U}, \boldsymbol{V}), \mathsf{strict}(\boldsymbol{U}, c_{\tau_i}, V_i, V').$$

$$\mathsf{min_g}(\boldsymbol{U}, V_1, \ldots, V_{i-1}, V', V_{i+1}, \ldots, V_n) \leftarrow \mathsf{max_g}(\boldsymbol{U}, \boldsymbol{V}), \mathsf{strict}(\boldsymbol{U}, c_{\tau_i}, V', V_i).$$

Table 3. Datalog program to compute query reformulations

Let $\boldsymbol{c} = (c_1, \ldots, c_n)$ *be a tuple of constants from* $\mathbf{C} \cup \mathbf{R} \cup \mathbf{I}$. *The* unfolding of Π_Ψ for $\boldsymbol{c}$ *is the rule obtained from*

$$\mathsf{query}_\Psi(\boldsymbol{c}, \boldsymbol{x}) \leftarrow \mathsf{refAtms}(\boldsymbol{c}), \mathsf{qAtm}_{\tau_1}(c_1, \boldsymbol{x_1}), \ldots, \mathsf{qAtm}_{\tau_n}(c_n, \boldsymbol{x_n})$$

by choosing some $\rho \in \Pi_\mathcal{Q}$ *for each* $\mathsf{qAtm}_{\tau_i}(c_i, \boldsymbol{x_i})$, *and a substitution* σ *such that*

- $head(\rho)\sigma = \mathsf{qAtm}_{\tau_i}(c_i, \boldsymbol{x_i})$, *and*
- *each atom* $\mathsf{conc}(c_i)$, $\mathsf{role}(c_i)$ *and* $\mathsf{const}(c_i)$ *is contained in* $D_{\Psi,\mathcal{R}}$

and then: (i) replacing $\mathsf{qAtm}_{\tau_i}(c_i, \boldsymbol{x_i})$ *with* $body(\rho)\sigma$, *and (ii) removing each atom from* $body(\mathsf{query}_\Psi(\boldsymbol{c}, \boldsymbol{x}))$ *that is contained in* $\Pi_\Psi(D_{\Psi,\mathcal{R}})$. *If the body of the unfolding* Π_Ψ *for* $\boldsymbol{c}$ *is* $tr(q)$ *for some CQ* q, *we call* $\boldsymbol{c}$ *the* Π_Ψ-encoding *of* q.

Example 4. Let $\boldsymbol{c} = (Conference, hasLoc, \mathsf{Rhodes})$. The unfolding of Π_Ψ for $\boldsymbol{c}$ is

$$\mathsf{query}_\Psi(\boldsymbol{c}, x) \leftarrow \mathsf{uAtm}(Conference, x), \mathsf{bAtm}(hasLoc, x, z), z = \mathsf{Rhodes}.$$

The body matches $tr(q_2)$, so $\boldsymbol{c}$ is a Π_Ψ-encoding for q_2.

If q has a Π_Ψ-encoding, then it is derivable from Ψ w.r.t $\mathcal{R}$. Conversely, for each derivable q we can find a Π_Ψ-encoding.

Lemma 2. *A CQ* q *is derivable from* Ψ *using* $\mathcal{R}$ *iff there exists a* Π_Ψ*-encoding of* q.

4.2 Evaluation of the Datalog Translation for *DL-Lite*$_\mathcal{R}$ OMQs

We now fix an ABox $\mathcal{A}$ and a *DL-Lite*$_\mathcal{R}$ ontology $\mathcal{O}$, and consider the evaluation of the queries $(q, \mathcal{O})$ in $\mathcal{Q}^\mathcal{R}_\Psi$ over $\mathcal{A}$. The relevant q are encoded in $\langle \Pi_\Psi, D_{\Psi,\mathcal{R}} \rangle$, but we still need an *OMQ answering algorithm* for the DL of $\mathcal{O}$. In the case of *DL-Lite*$_\mathcal{R}$, one could call an external query rewriting engine for the encoded queries. However, we chose to partially complete the data w.r.t. $\mathcal{O}$, and then evaluate the Datalog encoding over the extended dataset.[2]

[2] Such a procedure is very easy to realize if an *existential rule* engine [10] is chosen instead of a plain Datalog, as we do in the next section.

We say that a CQ template is *rooted* if each variable is either an answer variable or occurs in a join sequence of the form $r_1(x_0, x_1), \ldots, r_k(x_{k-1}, x_k)$, where x_1 is an answer variable and for $1 < i \leq k$, x_i occurs existentially in Ψ. We define the *join length* of a CQ template as the length of its longest join sequence, and let k be the join length of Ψ. In the following, B denotes either a concept name or $\exists r$. Then, we apply the following procedure:

1. For Σ denoting the signature of $(\Psi, \mathcal{R})$, we build a *k-bounded Σ-expansion* $\mathcal{A}_\Sigma^{\mathcal{O},k}$ *of* $\mathcal{A}$ *w.r.t.* $\mathcal{O}$ constructed by taking for each $A \in \Sigma$ and each $r \in \Sigma$:
   - If $\mathcal{T} \models B \sqsubseteq A$ and $\mathcal{A} \models B(a)$, then $A(a) \in \mathcal{A}_\Sigma^{\mathcal{O},k}$.
   - If $\mathcal{T} \models B \sqsubseteq \exists r_1 \sqsubseteq \ldots \sqsubseteq \exists r_n$, $\mathcal{T} \models \exists r_n^- \sqsubseteq A$ and $\mathcal{A} \models B(a)$, then $A(a_{r_1 \ldots r_n}) \in \mathcal{A}_\Sigma^{\mathcal{O},k}$, where $a_{r_1 \ldots r_n}$ is a fresh individual not in $\mathcal{A}$ and $n \leq k$.
   - If $\mathcal{T} \models s \sqsubseteq r$ (with s possibly inverse) and $\mathcal{A} \models s(a, b)$, then $r(a, b) \in \mathcal{A}_\Sigma^{\mathcal{O},k}$.
   - If $\mathcal{T} \models B \sqsubseteq \exists r_1 \sqsubseteq \ldots \sqsubseteq \exists r_n \sqsubseteq \exists r$ and $\mathcal{A} \models B(a)$ then $r(a_{r_1 \ldots r_n}, a_{r_1 \ldots r_n r}) \in \mathcal{A}_\Sigma^{\mathcal{O},k}$, where $k \geq n \geq 0$ and $a_{r_1 \ldots r_n}$ and $a_{r_1 \ldots r_n r}$ are fresh individuals.
2. Lastly, we translate $\mathcal{A}_\Sigma^{\mathcal{O},k}$ into the signature of Π_Ψ, similarly as before. For that, we assume that each individual in $\mathcal{A}_\Sigma^{\mathcal{O},k}$ is a constant in $\mathbf{K}$. We define $D_\mathcal{A} = tr(\mathcal{A}_\Sigma^{\mathcal{O},k})$, where tr translates each assertion in $\mathcal{A}_\Sigma^{\mathcal{O},k}$ as above.

We formulate a minor adaptation of a well known result in the OMQ literature [14, 6].

Lemma 3. *Let $(q, \mathcal{O})$ be a rooted DL-Lite$_\mathcal{R}$ OMQ over the signature Σ. Then, $ans(q, \mathcal{O}, \mathcal{A}) = ans(q, \emptyset, \mathcal{A}_\Sigma^{\mathcal{O},k})$.*

If the template is rooted, then we can answer all $(q, \mathcal{O})$ in the space by evaluating Π_Ψ over $D_{\Psi,\mathcal{R}} \cup D_\mathcal{A}$. From Lemmas 2 and 3 we obtain:

Theorem 1. *Let Ψ be rooted. Let $q \in \mathcal{Q}_\Psi^\mathcal{R}$, and let $\boldsymbol{c_q}$ be the Π_Ψ-encoding of q. Then*

$$\boldsymbol{a} \in ans(q, \mathcal{O}, \mathcal{A}) \quad \textit{iff} \quad \mathsf{query}_\Psi(\boldsymbol{c_q}, \boldsymbol{a}) \in \Pi_\Psi(D_{\Psi,\mathcal{R}} \cup D_\mathcal{A})$$

4.3 Datalog Program to Compute Query Reformulations

Finally, we also define a set Π_{ref} of Datalog rules that compute the minimal strict reformulations of each query in the space, which are the most relevant for exploration. The program Π_{ref} is presented in Table 3. In a nutshell, it derives atoms $\mathsf{min_g}(\boldsymbol{c_q}, \boldsymbol{c_{q'}})$ for tuples of Π_Ψ-encodings of queries q, q' such that $q' \in \mathsf{minStr}_\mathcal{A}^\mathsf{g}(q, \mathcal{Q})$, and analogously for the specializations. For this, it relies on finding pairs $\boldsymbol{c_q}$ and $\boldsymbol{c_{q'}}$ of Π_Ψ-encodings that are the result of applying one reformulation axiom such that $\mathsf{query}_\Psi(\boldsymbol{c_q}, \boldsymbol{a})$ and $\neg\mathsf{query}_\Psi(\boldsymbol{c_{q'}}, \boldsymbol{a})$. Using pairs of strict reformulations and stratified negation, we can identify the reformuations that are neutral, and again use negation to find the maximal neutral ones.

Relying on Observation 1, it is not hard to prove that Π_{ref} computes the relevant reformulations that we use to explore the dataset:

Template Ψ_i [# atoms in Ψ_i]	$\Psi_1[2]$	$\Psi_2[3]$	$\Psi_3[5]$	$\Psi_4[5]$	$\Psi_5[6]$
Size of $\Pi_i = (\mathrm{D}_i)$ **(MB)**	0.45	5	0.2	18.8	0.5
# Queries in $Q^{\mathcal{R}}_{\Psi_i}$ with answers	4	82	56	316	119
Sum of answers over all queries	1774	3431	16	2758	113
$\Delta_s(\Psi_i)$ / $\Delta_g(\Psi_i)$	350.7 / 175.2	236 / 1005	1.32 / 1.28	139 / 510	3.9 / 40
			Time		
Computation of $\Pi_i(\mathrm{D}_i)$ **(s)**	2.4	4.04	2	10.7	5.6
Avg. retrieval of answers $q \in Q^{\mathcal{R}}_{\Psi_i}$ **(ms)**	0.5	0.17	0.3	0.09	0.16
Avg. retrieval $q' \in \mathsf{minStrG/S}(q)$ **(ms)**	0.3 / 0.4	0.10 / 0.09	0.14 / 0.1	0.04 / 0.03	0.10 / 0.09

Table 4. Experiment results over DBpedia. Here $\Pi_i = \Pi_{\Psi_i} \cup \Pi_{ref}$ and $\mathrm{D}_i = D_{\Psi_i,\mathcal{R}} \cup D_{\mathcal{A}}$.

Theorem 2. *Let* $\mathcal{Q} = (Q^{\mathcal{R}}_{\Psi}, \preceq^{\mathcal{R}}_{\Psi})$ *and let* $D_{all} = D_{\Psi,\mathcal{R}} \cup D_{\mathcal{A}}$. *For* $q, q' \in \mathcal{Q}$, *let* $\boldsymbol{c_q}$ *be* Π_Ψ*-encoding of* q *and* $\boldsymbol{c_{q'}}$ *the* Π_Ψ*-encoding of* q'. *Then, for* $\mathsf{x} \in \{\mathsf{s}, \mathsf{g}\}$*:*
(a) $q' \in \mathsf{maxNeu}^{\mathsf{x}}_{\mathcal{A}}(q, \mathcal{Q})$ *iff* $\mathsf{max}_{\mathsf{x}}(\boldsymbol{c_q}, \boldsymbol{c_{q'}}) \in \Pi_{\Psi,\mathcal{R},ref}(D_{all})$,
(b) $q' \in \mathsf{minStr}^{\mathsf{x}}_{\mathcal{A}}(q, \mathcal{Q})$ *iff* $\mathsf{min}_{\mathsf{x}}(\boldsymbol{c_q}, \boldsymbol{c_{q'}}) \in \Pi_{\Psi,\mathcal{R},ref}(D_{all})$.

5 Implementation and Evaluation

We implemented our exploratory framework using Rulewerk Java API for VLog Datalog reasoner [10]. The implementation consists of a template parser and a translator of the template and rules of the encoding into Rulewerk syntax. All experiments have been performed on a MacBook Pro (2.7 GHz i5 8GB) using JavaSE 14.0.1 and Rulewerk version 0.5.0[3]. We used the DBpedia ontology and dataset, accessed via the endpoint[4]. We used one set $\mathcal{R}$ of reformulation axioms with 336 axioms and assertions extracted from DBpedia. With its signature and the DBpedia ontology, we computed the k-bounded Σ-extension $D_{\mathcal{A}}$ using existential rules in VLog. The resulting dataset was computed relatively fast, given that the data was remotely accessed, and it took in total about 3 minutes to materialize. The size of the extended dataset is around 700 MB. We have designed templates of various sizes and shapes over the ontology vocabulary containing classes such as: $\mathsf{Event}^{\mathsf{s}}$, $\mathsf{Museum}^{\mathsf{s}}$, $\mathsf{ArtWork}^{\mathsf{s}}$, $\mathsf{Organization}^{\mathsf{s}}$ etc., properties $\mathsf{startDate}^{\mathsf{g}}$, $\mathsf{museum}^{\mathsf{s}}$, $\mathsf{hasLocation}^{\mathsf{s}}$, $\mathsf{headquarter}^{\mathsf{g}}$ etc., and resources e.g., $\mathsf{Paris}^{\mathsf{g}}$. The main goals of our evaluation were (a) to test the feasibility of our framework in practice, in particular the tradeoff between the time to evaluate the Datalog program and the time to answer and compute query reformulations from the pre-computed model, and (b) to test if the template-generated query spaces ensure a gradual navigation of the answers. For each input Ψ_i we have measured: $|Q^{\mathcal{R}}_{\Psi_i}|$ - number of generated queries with answers, total number of answers captured by the query space, and as well as the computational time: to evaluate the Datalog program, average time to read the answers to queries and average time to compute reformulations from the compiled model. Then, for each query q, we

[3] https://github.com/knowsys/rulewerk
[4] SPARQL endpoint: https://dbpedia.org/sparql

measure $\Delta_s(q)$ – the average number of answers that are droped by $\mathsf{minStr}^{\mathsf{s}}_{\mathcal{A}}(q)$, respectively $\Delta_g(q)$ the average number of answers that $\mathsf{minStr}^{\mathsf{g}}_{\mathcal{A}}(q)$ ensures. Then, for the entire query space, $\Delta_s(\Psi_i) = (\sum_{q \in Q_{\Psi_i}} \Delta_s(q))/|Q_{\Psi_i}|$ measures the average discarded answers when navigating the query space and similarly for $\Delta_g(\Psi_i)$. Table 4 summarizes our evaluation. Evaluating the Datalog program over the materialized data is done within seconds for all the query spaces and depends on the number of answers captured by the query space (i.e., the larger the number of answers the longer it took to compile). Reading the answers to queries and navigating the query space is done in less than 1 ms in all cases. The selectivity and inclusivity of the minimal strict reformulations leads to a reasonably gradual exploration of the data, relative to the number of answers captured by each space.

6 Related Work and Conclusions

In recent years, several exploratory search engines have been proposed to support data access for different exploratory purposes. The basic idea at the core of many of them is to guide the query formulation process, in a step-by-step fashion. The many proposed techniques for exploring ontology-mediated data include similarity-based methods such as [17, 21], and visual query languages [4, 18, 19], which include ontology reasoning with query language expresivity ranging between tree-shaped CQs and monadic positive existential queries. More recently, [20] is able to cover formulation of SPARQL queries, however without ontological reasoning. To abstract away the ontology reasoning step needed to obtain complete answers, in [7] a schema-agnostic approach to rewrite *DL-Lite*$_{\mathcal{R}}$ OMQs into SPARQL 1.1. is proposed.

Query generalizations have been proposed as a technique for interpreting null answers (i.e., empty answers) in cooperative database systems [16]. The considered generalizations are similar to the ones we propose, however they also consider numeric comparisons, while we additionally consider the roll-up and drill-down operations. In line with, [15], we propose a query template language also designed to ease the process of constructing queries that allows to describe a set of CQs that are semantically related via two types of taxonomies: concept and role hierarchies, and dimensions. Then, navigating the query space is done by moving from one query to another that minimally changes the answers. One advantage of our approach is that it is implementable in Datalog. Evaluating the obtained Datalog program is related to the problem of answering queries using views, which has been intensively studied for relational data [12, 11]. As shown in [9], to answer OMQs using views, a different semantics to materialize the views is needed. However, for *DL-Lite*$_{\mathcal{R}}$ this can be done using existing techniques.

From our preliminary evaluation on DBpedia using VLog reasoner, our approach seems feasible in practice, however an extended evaluation is part of future work. It would also be interesting to exploit the compilation for analytical purposes and to extend the reformulation rules to relate queries that have a different structure.

Acknowledgements This work was supported by the Austrian Science Fund (FWF) projects P30360, P30873 and W1255.

References

1. Andresel, M., Ibáñez-García, Y., Ortiz, M., Simkus, M.: Relaxing and restraining queries for OBDA. In: AAAI. pp. 2654–2661. AAAI Press (2019)
2. Andresel, M., Ibáñez-García, Y.A., Ortiz, M., Simkus, M.: Taming complex role inclusions for DL-Lite. In: Description Logics. CEUR Workshop Proceedings, vol. 2211. CEUR-WS.org (2018)
3. Apt, K.R., Blair, H.A., Walker, A.: Towards a Theory of Declarative Knowledge, pp. 89–148. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988)
4. Arenas, M., Cuenca Grau, B., Kharlamov, E., Marciuska, S., Zheleznyakov, D.: Faceted search over ontology-enhanced RDF data. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. pp. 939–948. CIKM '14 (2014)
5. 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, New York, NY, USA (2003)
6. Bienvenu, M., Ortiz, M., Simkus, M., Xiao, G.: Tractable queries for lightweight description logics. In: IJCAI. pp. 768–774. IJCAI/AAAI (2013)
7. Bischof, S., Krötzsch, M., Polleres, A., Rudolph, S.: Schema-agnostic query rewriting in SPARQL 1.1. In: International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 8796, pp. 584–600. Springer (2014)
8. Calvanese, D., Giacomo, G.D., 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)
9. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: View-based query answering in description logics: Semantics and complexity. J. Comput. Syst. Sci. **78**(1), 26–46 (2012)
10. Carral, D., Dragoste, I., González, L., Jacobs, C.J.H., Krötzsch, M., Urbani, J.: Vlog: A rule engine for knowledge graphs. In: ISWC (2). Lecture Notes in Computer Science, vol. 11779, pp. 19–35. Springer (2019)
11. Halevy, A.Y.: Theory of answering queries using views. SIGMOD Rec. **29**(4), 40–47 (2000)
12. Halevy, A.Y.: Answering queries using views: A survey. VLDB J. **10**(4), 270–294 (2001)
13. Jung, J., Lutz, C., Wolter, F.: Least general generalizations in description logic: Verification and existence. In: AAAI 2020, Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, Februrary 7 -12, 2020, New York, New York, US (2020)
14. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: KR. AAAI Press (2010)
15. Martinenghi, D., Torlone, R.: Taxonomy-based relaxation of query answering in relational databases. VLDB J. **23**(5), 747–769 (2014)
16. Motro, A.: Query generalization: A method for interpreting null answers. In: Expert Database Workshop. pp. 597–616. Benjamin/Cummings (1984)

17. Sabou, M., Ekaputra, F.J., Ionescu, T.B., Musil, J., Schall, D., Haller, K., Friedl, A., Biffl, S.: Exploring enterprise knowledge graphs: A use case in software engineering. In: ESWC. Lecture Notes in Computer Science, vol. 10843, pp. 560–575. Springer (2018)
18. Sherkhonov, E., Grau, B.C., Kharlamov, E., Kostylev, E.V.: Semantic faceted search with aggregation and recursion. In: International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 10587, pp. 594–610. Springer (2017)
19. Soylu, A., Kharlamov, E., Zheleznyakov, D., Jimenez-Ruiz, E., Giese, M., Horrocks, I.: OptiqueVQS: Visual query formulation for OBDA. In: Proceedings of the 27th International Workshop on Description Logics (DL 2014). vol. 1193, pp. 725–728. Vienna, Austria (2014)
20. Vargas, H., Aranda, C.B., Hogan, A., López, C.: RDF explorer: A visual SPARQL query builder. In: ISWC (1). Lecture Notes in Computer Science, vol. 11778, pp. 647–663. Springer (2019)
21. Yahya, M., Berberich, K., Ramanath, M., Weikum, G.: Exploratory querying of extended knowledge graphs. Proc. VLDB Endow. **9**(13), 1521–1524 (2016)