Query Rewriting for Ontology-Mediated Conditional Answers

Medina Andreşel, Magdalena Ortiz, Mantas Šimkus
{andresel, ortiz}@kr.tuwien.ac.at, simkus@dbai.tuwien.ac.at
Institute of Logic and Computation, TU Wien, Austria

Abstract

Among many solutions for extracting useful answers from incomplete data, *ontology-mediated queries (OMQs)* use domain knowledge to infer missing facts. We propose an extension of OMQs that allows us to make certain assumptions—for example, about parts of the data that may be unavailable at query time, or costly to query—and retrieve *conditional answers*, that is, tuples that become certain query answers when the assumptions hold. We show that querying in this powerful formalism often has no higher worst-case complexity than in plain OMQs, and that these queries are first-order rewritable for *DL-Lite*$_\mathcal{R}$. Rewritability is preserved even if we allow some use of *closed predicates* to combine the (partial) closed- and open-world assumptions. This is remarkable, as closed predicates are a very useful extension of OMQs, but they usually make query answering intractable in data complexity, even in very restricted settings.

Introduction

Querying incomplete data and extracting useful answers from it is a long-standing challenge, and many approaches have been proposed over the years. *Ontology-mediated queries (OMQs)* are a promising way to exploit domain knowledge for this purpose (Xiao et al. 2018; Bienvenu and Ortiz 2015). In an OMQ a regular database query is paired with ontological knowledge expressed as a theory in a suitable logical formalism; frequently, *Description Logics (DLs)* are used to write ontologies (Baader et al. 2003). When the OMQ is evaluated over an input dataset, it can access not only the explicit facts, but also implicit facts that can be inferred from the data and the ontology. OMQs are useful, but in many scenarios they are insufficient. In particular, they provide limited means to return *relaxed* query answers, i.e. answers that are not certain (in terms of logical entailment), but that are still related to the initial query.

To obtain more and better answers from incomplete data, we extend OMQs in a novel way, building on a classic idea: rather than obtaining only *certain answers*, we retrieve *conditional answers* which, in a nutshell, pair tuples (which need not be certain answers) with sets of facts that would make them true certain answers. Traditional certain answers are just assumptive answers where the assumption component is empty. Conditional answers are a well known tool for acccessing data that need not be current and complete (Demolombe 1992; 1998; Christiansen and Andreasen 1998), and have been explored in databases for several purposes, including temporal data (Arenas and Bertossi 2002), data streams (Cruz-Filipe, Gaspar, and Nunes 2019), and other forms of incomplete data (Zhang et al. 2007). In the setting of OMQs, they are very natural. For example, when querying diverse data sources—a key application of OMQs—parts of the data may be missing completely or be very costly to obtain.With the above in mind, we enrich OMQs with *assumption patterns*, which define the form of complex assertions that may become true about some individuals. This leads to a powerful formalism, but does not impact significantly the worst-case complexity compared to plain OMQs. Importantly, our OMQs with assumptions are *first-order (FO) rewritable* for *DL-Lite*$_\mathcal{R}$ (Artale et al. 2009). An OMQ is said to be FO-rewritable if it can be converted into FO queries that return all the answers to the OMQ when evaluated over the data alone (with no ontology). FO-rewritability is highly desired: it implies very low data complexity, and that OMQ answering can be realized using existing database technologies (Bienvenu et al. 2018; Gottlob et al. 2014).

Our extension of OMQs turns out to be particularly powerful when complete and incomplete information coexist. Based on classical logic, standard OMQs make the *open-world assumption (OWA)*, where a fact that cannot be inferred may be either true or false. Traditional databases, in contrast, make the *closed-world assumption (CWA)*: facts that cannot be inferred are assumed false. The OWA makes OMQs useful for incomplete data, but at the same time, too weak to yield useful answers when the data is known to be partially complete. For example, consider an OMQ where a query asking for stations accessible from airport Heathrow: $q(x) = conn(\mathsf{Heathrow}, x) \wedge Station(x)$ is paired with a TBox stating that each airport is connected to some station: ($Airport \sqsubseteq \exists conn.Station$ in DL notation). Under the classical semantics, this OMQ does not have answers over the dataset $\{Airport(\mathsf{Heathrow}), Station(\mathsf{HydePark})\}$: we

know that Heathrow is connected to station but we cannot be certain that it is HydePark (could be another station). However, if we *know* that the list of stations is complete in the data, and thus that s is the only station, then we expect Heathrow to be an answer. A simple yet powerful way to achieve this is to explicitly declare the predicates that should be assumed complete, e.g., marking $Station$ as closed. Unfortunately, doing so is costly: even for the most restricted OMQs languages, like CQs paired with *DL-Lite* ontologies, closed predicates make OMQ answering intractable in data complexity, and destroy FO-rewritability (Lutz, Seylan, and Wolter 2013). Remarkably, in our OMQs, we can use closed predicates in our assumptions, for instance assuming that $\exists conn.Station(\mathsf{Heathrow})$, will allow us, to obtain HydePark as part of the answer, while preserving FO-rewritability.

Proofs that are ommitted due to space restrictions can be found in the extended version.

Preliminaries

Let $\mathsf{N_C}$, $\mathsf{N_R}$, $\mathsf{N_I}$, and $\mathsf{N_V}$ be countably infinite, disjoint sets of *concept names*, *role names*, *individuals*, and *variables*, respectively. Elements of $\mathsf{N_R^\pm} := \mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$ are called *roles*, and r^- is the *inverse* of r. ($\mathcal{ELI}$) *concepts* are defined inductively: (a) $\top$ and each $A \in \mathsf{N_C}$ are concepts, and (b) if C, D are concepts and r is a role, then $C \sqcap D$ and $\exists r.C$ are concepts. A *basic concept* is any $A \in \mathsf{N_C}$, or a concept $\exists r.\top$ (written also $\exists r$). A (*DL-Lite*$_\mathcal{R}$) *concept inclusion* has the form $B_1 \sqsubseteq B_2$, a *role inclusion* the form $r_1 \sqsubseteq r_2$, and a *disjointness assertion* the form $disj(B_1, B_2)$ or $disj(r_1, r_2)$, where B_1, B_2 are basic concepts and r_1, r_2 are roles. A (*DL-Lite*$_\mathcal{R}$) *TBox* $\mathcal{T}$ is a finite set of concept inclusions, role inclusions, and disjointness assertions.

Elements of $\mathsf{N_I} \cup \mathsf{N_V}$ are *terms*. *Atoms* are of the form $r(t, t')$ and $C(t)$, where t, t' are terms, $r \in \mathsf{N_R^\pm}$, and C is a concept. Atoms without variables are called *ground atoms* or *assertions*. An *ABox* $\mathcal{A}$ is a finite set of assertions. We use the term *database*, and the symbol $\mathcal{D}$, for ABoxes with assertions only of the form $A(t)$ or $r(t, t')$, where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$.[1]

We employ the standard notation for *interpretations*. They have the form $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is the *domain* and $\cdot^\mathcal{I}$ is the *interpretation function*. We make the *standard name assumption (SNA)*, since some of our results consider closed predicates.[2] In particular, we assume that $\mathsf{N_I} \subseteq \Delta^\mathcal{I}$ and $c^\mathcal{I} = c$ for all $c \in \mathsf{N_I}$. The notions of *modelhood*, *consistency* and *entailment* are standard. Note that each database $\mathcal{D}$ can be seen as an interpretation $\mathcal{I}_\mathcal{D}$ with domain $\mathsf{N_I}$ and with $A^\mathcal{I}_\mathcal{D} = \{c \mid A(c) \in \mathcal{D}\}$ for all $A \in \mathsf{N_C}$ and $r^\mathcal{I}_\mathcal{D} = \{(c, c') \mid r(c, c') \in \mathcal{D}\}$ for all $r \in \mathsf{N_R}$.

A *first-order (FO) query* φ is any formula of function-free FO logic, built using atoms over concept and role names only, as well as equality atoms $t_1 = t_2$ for terms t_1, t_2. We use $terms(\varphi)$, $vars(\varphi)$ and $free(\varphi)$ to denote the terms, the variables, and the free (or *answer*) variables of φ, respectively. The *arity* of φ is $|free(\varphi)|$. We may write $\varphi(\vec{x})$ to indicate that $free(\varphi) = \vec{x}$ (here we assume an arbitrary but fixed ordering of $free(\varphi)$). If $\varphi(\vec{x})$ has the form $\exists \vec{y}.(\ell_1 \wedge \cdots \wedge \ell_k)$, where $\ell_1, \ldots, \ell_k$ are atoms, then it is a *conjunctive query (CQ)*. We may sometimes treat such a $\varphi(\vec{x})$ as the set $\{\ell_1, \ldots, \ell_k\}$. If $\varphi(\vec{x})$ has the form $q_1(\vec{x}) \vee \cdots \vee q_k(\vec{x})$, where each $q_i(\vec{x})$ is a CQ, we call it a *union of conjunctive queries (UCQ)*. An *answer* to an FO query φ of arity k over an interpretation $\mathcal{I}$ is a k-tuple $\vec{c}$ of individuals such that $\mathcal{I} \models \varphi(\vec{c})$. We say that a 0-ary (aka *Boolean*) query evaluates to $true$ in $\mathcal{I}$ if $\mathcal{I} \models \varphi$ (i.e., the empty tuple is an answer), and to $false$ otherwise.

A (k-ary) *ontology-mediated query (OMQ)* is a pair $(q(\vec{x}), \mathcal{T})$ where $q(\vec{x})$ is a k-ary CQ and $\mathcal{T}$ is a TBox. Its *(certain) answers over a database* $\mathcal{D}$ are defined as $ans(\mathcal{Q}, \mathcal{D}) = \{\vec{c} \in (\mathsf{N_I})^k \mid \mathcal{I} \models q(\vec{c}) \text{ for all } \mathcal{I} \models (\mathcal{T}, \mathcal{D})\}$. The *query answering problem* is to decide, given an OMQ $\mathcal{Q}$, a tuple $\vec{a}$ and a database $\mathcal{D}$, whether $\vec{a} \in ans(\mathcal{Q}, \mathcal{D})$.

We use standard notations for *substitutions*, which map terms to terms, and call a substitution *grounding* if all terms are mapped to individuals. A substitution θ *unifies* two sets of atoms Γ_1 and Γ_2 if $\Gamma_1\theta = \Gamma_2\theta$.

Assumptive Ontology-mediated Queries

We extend standard OMQs with *assumption patterns*, which prescribe the shapes of additional assertions that we may want to assume true for query answering. In this paper, we choose to define them as $\mathcal{ELI}$ concepts and roles: this appears to be a relatively simple language for the assumption patters, and it naturally captures interesting examples.

Definition 1. *An* assumptive ontology-mediated query *is a triple $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H})$, where $(q(\vec{x}), \mathcal{T})$ is an OMQ, and $\mathcal{H}$ is a set of atoms called* assumption patterns.

Intuitively, $\mathcal{Q}$ gives us *conditional answers* over $\mathcal{D}$: pairs $(\vec{c}, \mathcal{E})$ where $\vec{c}$ is an answer to $(q(\vec{x}), \mathcal{T})$ over $\mathcal{D} \cup \mathcal{E}$.

Definition 2. *A pair $(\vec{a}, \mathcal{E})$ of a tuple of constants $\vec{a}$ and an ABox $\mathcal{E}$ is a* conditional answer *to $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H})$ over a database $\mathcal{D}$ if*

1. *$\mathcal{D} \cup \mathcal{E}$ is consistent with $\mathcal{T}$, and*
2. *there is substitution π such that: (a) $\pi(\vec{x}) = \vec{a}$, (b) $\mathcal{E} \subseteq \mathcal{H}\pi$, and (c) $\mathcal{T}, \mathcal{D} \cup \mathcal{E} \models q\pi$.*

We denote by $cans(\mathcal{Q}, \mathcal{D})$ the set of conditional answers to $\mathcal{Q}$ over $\mathcal{D}$. The conditional answering problem *is defined as follows: Given database $\mathcal{D}$, AOMQ $\mathcal{Q}$ and a pair $(\vec{a}, \mathcal{E})$, test if $(\vec{a}, \mathcal{E})$ is a conditional answer of $\mathcal{Q}$ w.r.t. $\mathcal{D}$.*

We call $(\vec{a}, \mathcal{E}) \in cans(\mathcal{Q}, \mathcal{D})$ a minimal conditional answer *if there is no $\mathcal{E}' \subsetneq \mathcal{E}$ such that $(\vec{a}, \mathcal{E}') \in cans(\mathcal{Q}, \mathcal{D})$, and denote the set thereof by $cans_{min}(\mathcal{Q}, \mathcal{D})$.*

The set of (minimal) conditional answers can be in general infinite, e.g., if $\mathcal{Q} = (A(x), \emptyset, \{A(x)\})$ and $\mathcal{D} = \emptyset$ then $cans_{min}(\mathcal{Q}, \mathcal{D}) = \{(c, A(c)) \mid c \in \mathsf{N_I}\}$. We often consider the conditional answers where π ranges over *the active domain of* $\mathcal{D}$, $adom(\mathcal{D}) = \{a \mid A(a) \in \mathcal{D}\} \cup \{a, b \mid r(a, b) \in \mathcal{D}\}$. We denote this subset of conditional answers by $cans_{adom}(\mathcal{Q}, \mathcal{D})$.

[1] This use of *databases* is not related to the open- or closed-world assumption, and is not to be confused with DBoxes (Franconi, Ibáñez-García, and Seylan 2011).

[2] With no closed predicates, the SNA doesn't impact our results.

Note that $\mathcal{H}$ and q may share variables, and this restricts the query positions for which some assumption patterns may be applicable (as the same substitution π is applied to both).

Example 1. *Ann wants to find vegan restaurants in a central district that are easily accessible by bus. She can use a query*

$$q(x) = \exists yz\ VeganRest(x) \wedge locNext(x, y) \wedge BusStop(y) \wedge partOf(y, z) \wedge CntDst(z).$$

But she knows that, in the database, some spatial relations like locNext are incomplete, and to get their complete extensions queries are sent to a remote and sometimes slow geospatial database. Therefore, she chooses to enable assumptions $\mathcal{H} = \{locNext(x, y),\ BusStop(y)\}$ to postpone verifying whether points are next to a bus stop.

The query has no certain answers over the database $\mathcal{A} = \{\, VeganRest(\mathsf{r}_1), BusStop(\mathsf{s}_1), BusStop(\mathsf{s}_2)$ $partOf(\mathsf{s}_1, \mathsf{a}), CntDst(\mathsf{a})\}$, *but the AOMQ $(q(x), \emptyset, \mathcal{H})$ produces $(\mathsf{r}_1, \{locNext(\mathsf{r}_1, \mathsf{s}_1)\})$ as (minimal) conditional answer, and she can later verify whether the candidate r_1 indeed has the bus stop s_1 nearby. More complex assumptions like $\mathcal{H}' = \{locNext(x, y), BusStop \sqcap partOf.CntDst(y)\}$ give $(\mathsf{r}_1, \{locNext(r_1, s_2),\ BusStop \sqcap \exists partOf.CntDst\ (\mathsf{s}_2)\})$ as a conditional answer, indicating that that r_1 is also an answer to $q(x)$ if it is known to be located next to s_2 and additionally s_2 is situated in a central district.*

The example illustrates that conditional answers for AOMQs can retrieve additional information that we would not obtain with standard OMQs. Indeed, AOMQs are a generalization of standard OMQs: all certain answers are conditional answers, provided that the database is consistent.

Proposition 1. *Let $\mathcal{Q} = (q(\vec{x}), \mathcal{T})$ be an OMQ. Then, for every database $\mathcal{D}$ consistent with $\mathcal{T}$ we have*

$$\vec{a} \in ans(\mathcal{Q}, \mathcal{D})\ \textit{iff}\ (\vec{a}, \emptyset) \in cans((q(\vec{x}), \mathcal{T}, \emptyset), \mathcal{D})$$

For OMQ languages where we can rely on exisiting algorithms for entailment of regular OMQs and consistency testing, it is not hard to obtain an algorithm for answering AOMQs. Given a candidate conditional answer $(\vec{a}, \mathcal{E})$, a database $\mathcal{D}$, and an AOMQ $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H})$, we can decide whether $(\vec{a}, \mathcal{E}) \in cans(\mathcal{Q}, \mathcal{D})$

in three steps:

1. Guess $\mathcal{H}' \subseteq \mathcal{H}$ and a substitution π from $vars(\mathcal{H}') \cup free(q)$ to individuals in $\mathcal{E}$.
2. Check if $\mathcal{H}'\pi = \mathcal{E}$, if $\pi(\vec{x}) = \vec{A}$, and if $\mathcal{D} \cup \mathcal{E}$ is consistent with $\mathcal{T}$.
3. Test whether $\mathcal{T}, \mathcal{D} \cup \mathcal{E} \models q\pi$.

For typical OMQ languages that combine well known DLs with CQs, the combined complexity of OMQ answering falls into the classes NP, ExpTime, or 2-ExpTime. In such cases, this simple algorithm yields tight complexity bounds, as the cost of the additional steps is subsumed by the cost of answering OMQs.

Theorem 1. *Given $(\vec{a}, \mathcal{E})$, a database $\mathcal{D}$, and an AOMQ $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H})$, the following results hold for the combined complexity of deciding $(\vec{a}, \mathcal{E}) \in cans(\mathcal{Q}, \mathcal{D})$:*

- *It is* NP*-complete for DL-Lite$_{core}$, DL-Lite$_{\mathcal{R}}$ and $\mathcal{EL}^{\bot}$.*
- *It is* ExpTime*-complete for $\mathcal{ELI}$, Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SHOIQ}$.*
- *It is* ExpTime*-complete for $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$.*

Rewriting *DL-Lite*$_{\mathcal{R}}$ AOMQs

In this section we provide a *rewriting algorithm* for AOMQs $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H})$ where $\mathcal{T}$ is a *DL-Lite*$_{\mathcal{R}}$ TBox. The algorithm takes such a $\mathcal{Q}$ as input, and it outputs a set of FO-queries $Rew(\mathcal{Q})$, whose answers over any database $\mathcal{D}$ are in one-to-one correspondence with the conditional answers of $\mathcal{Q}$ over $\mathcal{D}$.

Given a TBox $\mathcal{T}$, we denote by $neg(\mathcal{T})$ the set of all disjointness axioms in $\mathcal{T}$, and by $pos(\mathcal{T})$ the set $\mathcal{T} \setminus neg(\mathcal{T})$. Our procedure to obtain the full FO-rewriting for AOMQs has three steps:

(1) We rewrite $\mathcal{Q}$ w.r.t. $pos(\mathcal{T})$ using the standard *DL-Lite*$_{\mathcal{R}}$ rewriting rules.
(2) Each resulting query is rewritten w.r.t. $\mathcal{H}$.
(3) Finally, we take into account $neg(\mathcal{T})$.

Rewriting w.r.t. Positive TBox It is a well-known result due to (Calvanese et al. 2007) that an OMQ with a *DL-Lite*$_{\mathcal{R}}$ TBox can be reformulated as a UCQ that can simply be evaluated over an input dataset (without taking the input TBox into account).

Proposition 2 ((Calvanese et al. 2007))**.** *For an OMQ $\mathcal{Q} = (q(\vec{x}), \mathcal{T})$, where $\mathcal{T}$ is a DL-Lite$_{\mathcal{R}}$ TBox, one can compute a UCQ $q_{\mathcal{T}}(\vec{x})$ such that $ans(\mathcal{Q}, \mathcal{D}) = \{\vec{c} \,|\, \mathcal{I}_{\mathcal{D}} \models q_{\mathcal{T}}(\vec{c})\}$ for every database $\mathcal{D}$ consistent with $\mathcal{T}$.*

The above rewriting from an ontology-mediated CQ into a plain UCQ preserves conditional answers.

Lemma 1. *Let $(q(\vec{x}), \mathcal{T})$ be an OMQ where $\mathcal{T}$ is a DL-Lite$_{\mathcal{R}}$ TBox. Then, for every database $\mathcal{D}$ consistent with $\mathcal{T}$ and every set of assumption patterns $\mathcal{H}$,*

$$cans((q(\vec{x}), \mathcal{T}, \mathcal{H}), \mathcal{D}) = cans((q_{\mathcal{T}}(\vec{x}), \emptyset, \mathcal{H}), \mathcal{D}).$$

Rewriting w.r.t. Assumption Patterns In the next step, we rewrite the query with respect to the assumption patterns. The core idea is to identify subqueries which are made true by the assumptions, and drop them. Since $\mathcal{H}$ can contain complex atoms, it is convenient to expand it as follows.

Definition 3. *An* expansion *of a set $\mathcal{H}$ of assumption patterns is obtained by adding atoms using the following rules:*

- *If $(C_1 \sqcap C_2)(t) \in \mathcal{H}$, add $C_1(t)$ and $C_2(t)$ to $\mathcal{H}$.*
- *If $(\exists p.C)(t) \in \mathcal{H}$ and there are no $\{p(t, y), C(y)\} \subseteq \mathcal{H}$, add $p(t, y)$ and $C(y)$ to $\mathcal{H}$ for some fresh variable y.*
- *If $r^-(t, t') \in \mathcal{H}$, then add $r(t', t)$ to $\mathcal{H}$.*
- *If $r(t, t') \in \mathcal{H}$, then add $r^-(t', t)$ to $\mathcal{H}$.*

We use $\exp(\mathcal{H})$ to denote a fixed arbitrary expansion of $\mathcal{H}$.

In AOMQs, we can assume any grounding of the atoms in the hypothesis $\mathcal{H}$, and this may make some atoms of the query true. We reflect this in the rewriting by simply dropping such atoms. In general, we may drop atoms that contain

answer variables, or that share variables with other atoms that are not removed. These variables need some care: we must keep track of the terms in $\mathcal{H}$ that will give us their query match in the conditional answers.

Definition 4 (Rewriting w.r.t. $\mathcal{H}$)**.** *Let $q(\vec{t})$ be a CQ and Γ a subset of atoms in q. We denote by $keep(q, \Gamma)$ the set of variables in $vars(\Gamma)$ that are in $free(q) \cup vars(q \setminus \Gamma)$.*

Let $\mathcal{H}$ be a set of assumption patterns and $q(\vec{t})$ a CQ. We write $q(\vec{x}) \rightsquigarrow_{\mathcal{H}} q'(\vec{x'})$ and call q' a rewriting of q w.r.t. $\mathcal{H}$ *if it is obtained by choosing*

(i) a subset Γ of the atoms of q and a subset $\mathcal{H}'$ of $\mathcal{H}$,
(ii) a substitution h that unifies Γ and $\exp(\mathcal{H}')$, and which has $h(x) \in terms(\mathcal{H}')$ for each $x \in keep(q, \Gamma)$,

and then doing the following:
1. Replace by $\top$ every atom in Γ.
2. Add $\bigwedge\{x = h(x) \mid x \in keep(q, \Gamma) \cup vars(\mathcal{H}')\}$.
3. Drop existential quantification for each $x \in vars(\mathcal{H}')$.
The set $hrew(q, \mathcal{H})$ contains all q' such that $q \rightsquigarrow_{\mathcal{H}} q'$.

Please note that the rewritten queries may have more free variables than the original q. After each rule application the resulting q' has $vars(\mathcal{H}')$ as free variables, additionally to the original $free(q)$. These additional free variables and the added equality atoms will allow us to read from each (ordinary) answer to some rewritten OMQ, the assignment for $\mathcal{H}$ that gives the corresponding conditional answer.

Example 2. *Recall $q(x)$ from Example 1, and consider $\mathcal{H} = \{locNext(x', y'), BusStop(y')\}$. Take $\Gamma = \{locNext(x, y), BusStop(y)\}$ and $\mathcal{H}' = \mathcal{H}$. For $h = \{x \mapsto x', y \mapsto y'\}$, since $keep(q, \Gamma) = \{x, y\}$ and $h(x), h(y) \in vars(\mathcal{H})$, we obtain the following query $q'(x, x', y')$ as a rewriting of q w.r.t. $\mathcal{H}$:*

$$\exists yz\ VeganRest(x) \wedge partOf(y, z) \wedge CntDst(z) \wedge x = x' \wedge y = y'.$$

We obtain $ans(q'(x, x', y'), \mathcal{D}) = \{(\mathsf{r}_1, \mathsf{r}_1, \mathsf{s}_1)\}$ over our example database $\mathcal{D}$, and the substitution $\pi = \{x \mapsto \mathsf{r}_1, x' \mapsto \mathsf{r}_1, y' \mapsto \mathsf{s}_1\}$ yields the conditional answer $(\mathsf{r}_1, \{locNext(\mathsf{r}_1, \mathsf{s}_1), BusStop(\mathsf{s}_1)\})$.

Note that if $\mathcal{H} = \{\exists locNext.BusStop(x)\}$ the rewriting is not applicable to $q(x)$, since there is no unifier h such that $h(y) = x$, for $y \in keep(q, \Gamma)$. Intuitively, this is because $\mathcal{H}$ just ensures that some point is next to a bus stop, but one cannot ensure that the bus stop is in a central district. In contrast, $\mathcal{H}' = \{\exists locNext.(BusStop \sqcap partOf.CntDst)(x)\}$ yields $q'(x) = VeganRest(x) \wedge x = x$.

The key to the correctness is that the unifier h for Γ and $\mathcal{H}'$ exists iff the atoms of Γ are made true in a grounding of $\mathcal{H}$, so each rewriting step drops precisely atoms that would be made true by assuming some grounding of the hypothesis. We show that the rewriting is complete.

Lemma 2. *Let $\mathcal{Q} = (q(\vec{x}), \emptyset, \mathcal{H})$ be an AOMQ where $q(\vec{x})$ is a CQ. For every substitution π, every database $\mathcal{D}$, and $\mathcal{H}' \subseteq \mathcal{H}$, if $(\pi(\vec{x}), \mathcal{H}'\pi) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$, then there is some $q \rightsquigarrow_{\mathcal{H}} q'$ such that $\pi(\vec{x}, \vec{y}) \in ans(q'(\vec{x}, \vec{y}), \mathcal{D})$.*

Proof(Sketch). Take $\mathcal{D}$, $\mathcal{H}' \subseteq \mathcal{H}'$ and $(\pi(\vec{t}), \mathcal{H}'\pi) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$. By definition, $\mathcal{D} \cup \mathcal{H}'\pi \models q\pi$. Consider maximal subquery q'' such that $\mathcal{D} \models q''\pi$ and choose Γ the remainder subquery. Making some case distinctions according to the size and shape of q'', the rewriting is shown to be applicable to Γ, thus obtaining q' which consists of q'' plus some term equalities, which is matched by π into $\mathcal{D}$. □

The following lemma shows that it is also sound.

Lemma 3. *Let $\mathcal{Q} = (q(\vec{x}), \emptyset, \mathcal{H})$ be an AOMQ and where q is a CQ, and let $\mathcal{D}$ be a database. If $q(\vec{x}) \rightsquigarrow_{\mathcal{H}} q'(\vec{x} \cup \vec{y})$ and π is a substitution with $\pi(\vec{x}, \vec{y}) \in ans(q', \mathcal{D})$, then there is some $\mathcal{H}' \subseteq \mathcal{H}$ such that $(\pi(\vec{x}), \mathcal{H}'\pi) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$.*

Proof(Sketch). Arbitrarily fix $\mathcal{D}$, let $q'(\vec{x} \cup \vec{y})$ be a rewriting of q w.r.t. $\mathcal{H}$ and π an arbitrary substitution such that $\pi(\vec{x}, \vec{y}) \in ans(q'(\vec{x} \cup \vec{y}), \mathcal{D})$. By definition, for some $\Gamma \subseteq q$ and $\mathcal{H}' \subseteq \mathcal{H}$ such that $\vec{y} = vars(\mathcal{H}')$ there is a unifier h such that $h(x) \in \vec{y}$, for each $x \in keep(q, \Gamma)$. Since $\Gamma h = \Gamma' h$ and $x = h(x) \in q'$, for each $x \in keep(q, \Gamma) \cup \vec{y}$, and since $\vec{y} \subseteq free(q'')$, we obtain that $\Gamma'\pi \models (\Gamma \wedge \{x = h(x) \mid x \in keep(q, \Gamma) \cup \vec{y}\})\pi$. It follows that, $\mathcal{D} \cup \mathcal{H}'\pi \models q\pi$ hence $(\pi(\vec{x}), \mathcal{H}'\pi) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$. □

To rewrite an AOMQ, we apply first the standard OMQ rewriting and then use the assumption patterns. It is then convenient to identify UCQ $q_{\mathcal{T}}(\vec{x})$ as the set of its CQs, denoted by $trew(q(\vec{x}), \mathcal{T})$.

Definition 5 (Perfect Rewriting of AOMQ over $pos(\mathcal{T})$)**.** *Let $\mathcal{Q} = (q(\vec{x}), pos(\mathcal{T}), \mathcal{H})$ be an AOMQ where $\mathcal{T}$ is a DL-Lite$_{\mathcal{R}}$ TBox. Then its* perfect rewriting *is the following set of CQs $Rew(Q)$*

$$\{q''(\vec{y}) \mid q''(\vec{y}) \in hrew(q', \mathcal{H}) \text{ and } q'(\vec{x}) \in trew(q(\vec{x}), \mathcal{T})\}.$$

Note that the queries in $Rew(Q)$ may not all share the same answer variables, hence we write it as a set of CQs rather than a UCQ.

By Lemmas 1, 2, and 3, $Rew(Q)$ is sound and complete for AOMQs over TBoxes with no disjointness assertions. The proof can be found in the online version, and it is a special case of Theorem 2 below.

Incorporating the Disjointness Assertions Now we consider disjointness assertions. In standard OMQs, to evaluate an OMQ $(\varphi, \mathcal{T})$ one can usually check first if the given database $\mathcal{D}$ is consistent with $\mathcal{T}$, and then answer φ assuming consistency of $(\mathcal{T}, \mathcal{D})$. Unfortunately, this approach is not sufficient for AOMQs, since we also need to verify whether a candidate $\mathcal{E}$ causes the inconsistency of an otherwise consistent $(\mathcal{T}, \mathcal{D})$. For that reason, we incorporate the inconsistency check as part of the rewritten query.

Definition 6 (Inconsistency CQs)**.** *For any given DL-Lite$_{\mathcal{R}}$ TBox $\mathcal{T}$, let $\alpha \in neg(\mathcal{T})$. We define the (Boolean) CQ $q_\alpha()$ as follows:*

- *if $\alpha = disj(A, A')$, then q_α is $\exists x\ A(x) \wedge A'(x)$,*
- *if $\alpha = disj(A, \exists r)$, then q_α is $\exists xy\ A(x) \wedge r(x, y)$,*
- *if $\alpha = disj(r_1, r_2)$, then q_α is $\exists xy\ r_1(x, y) \wedge r_2(x, y)$.*

For a given set of assumption patterns $\mathcal{H}$, the set of inconsistent CQs for $(\mathcal{T}, \mathcal{H})$*, denoted by Rew$_\perp(\mathcal{H}, \mathcal{T})$ is*

$$\bigcup_{\alpha \in neg(\mathcal{T})} \{q'' \mid q'' \in hrew(q', \mathcal{H}),\ \text{and } q' \in trew(q_\alpha, \mathcal{T})\}.$$

Note that, again, we obtain a set of queries with possibly different subsets of $vars(\mathcal{H})$ as free variables. We show the following using Lemmas 2 and 3.

Lemma 4. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox and $\mathcal{H}$ a set of assumption patterns. For any database $\mathcal{D}$ consistent with $\mathcal{T}$, and π a substitution that ranges over adom($\mathcal{D}$), we have that $\pi(\vec{y}) \in ans(q_\bot, \mathcal{D})$ for some $q_\bot \in Rew_\bot(\mathcal{H}, \mathcal{T})$ iff there is some $\mathcal{H}' \subseteq \mathcal{H}$ such that $\mathcal{D} \cup \pi(\mathcal{H}')$ is inconsistent with $\mathcal{T}$.*

To prevent inconsistency we can negate $Rew_\bot(\mathcal{H}, \mathcal{T})$ and add it to the previous rewriting. The only minor issue to take care of is that the queries have different free variables. For a tuple $\vec{y}$ from $vars(\mathcal{H})$, we denote by $Rew(\mathcal{Q}^{\vec{y}}_{pos})$ the UCQ whose disjuncts are the queries $q \in Rew(\mathcal{Q}_{pos})$ with $free(q) = \vec{y}$, and similarly, $Rew_\bot{}^{\vec{y}}(\mathcal{H}, \mathcal{T})$ is a UCQ with all queries $q \in Rew_\bot{}^{\vec{y}}(\mathcal{H}, \mathcal{T})$ with $free(q) = \vec{y}$. Then the rewriting w.r.t. a general TBox is obtained as follows.

Definition 7 (Perfect Rewriting of general AOMQs)**.** *Let $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H})$ be an AOMQ such that $\mathcal{T}$ is a DL-Lite$_\mathcal{R}$ TBox, Then the perfect rewriting of $\mathcal{Q}$, $Rew(\mathcal{Q})$ is defined as*

$$\bigcup_{\vec{y} \subseteq vars(\mathcal{H})} \{Rew(\mathcal{Q}^{(\vec{x},\vec{y})}_{pos}) \wedge \neg Rew_\bot{}^{\vec{y}}(\mathcal{H}, \mathcal{T})\}.$$

The following theorem is proved using Lemma 4 and the fact that perfect rewriting for $\mathcal{Q}_{pos}$ is correct.

Theorem 2. *Let $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H})$ be a DL-Lite$_\mathcal{R}$ AOMQ. For every database $\mathcal{D}$, the following are equivalent:*

- $(\vec{a}, \mathcal{E}) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$.
- *There exists $\mathcal{H}' \subseteq \mathcal{H}$, $\varphi(\vec{x}, \vec{y}) \in Rew(\mathcal{Q})$, a substitution π such that $\pi(\vec{x}) = \vec{a}$, $\mathcal{H}'\pi = \mathcal{E}$ and $\pi(\vec{x}, \vec{y})$ is an answer to the FO query φ over $\mathcal{I}_\mathcal{D}$.*

AOMQs with Closed Predicates

It has been argued often in the literature that strengthening the usual certain answer semantics of OMQs, where all models under the usual open-world semantics of ontologies are taken into account, is highly desirable. This can be done by declaring some predicates as *closed*.

Definition 8. *Let $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$ be a set of* closed predicates*. An interpretation $\mathcal{I}$ is a* Σ-model *of $(\mathcal{T}, \mathcal{D})$, written $\mathcal{I} \models_\Sigma (\mathcal{T}, \mathcal{D})$, if $\mathcal{I} \models (\mathcal{T}, \mathcal{D})$ and additionally $\vec{a} \in p^\mathcal{I}$ implies $p(\vec{a}) \in \mathcal{D}$ for all $p \in \Sigma$.*

An OMQ with closed predicates (OMQC) *is a tuple $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \Sigma)$ where $(q, \mathcal{T})$ is an OMQ.*

The notion of certain answers for OMQCs is lifted in the usual way from OMQs. Unfortunately, checking if a given tuple is an answer over its Σ models is intractable even if $\mathcal{T}$ is in the most basic DLs. For *DL-Lite$_{core}$*, for instance, it is coNP-hard, and thus not FO-rewritable (Lutz, Seylan, and Wolter 2013).

In this section, we show that AOMQs allow us to add closed predicates in a restricted, yet non-trivial way, while preserving FO-rewritability.

Definition 9 (AOMQ with closed predicates (AOMQC))**.** *An* AOMQ with closed predicates (AOMQC) *is a tuple $\mathcal{Q} = (q, \mathcal{H}, \mathcal{T}, \Sigma)$, where $(q, \mathcal{H}, \mathcal{T})$ is an AOMQ and Σ is a set of concept and role names that do not occur in $\mathcal{T}$.*

We generalize the definition of conditional answers to AOMQCs. Note the first condition: now we cannot take arbitrary groundings $\mathcal{E}$ of $\mathcal{H}$ in conditional answers, but only groundings that respect the extensions of Σ in the input $\mathcal{D}$.

Definition 10. *A pair $(\vec{a}, \mathcal{E})$ of a tuple of constants $\vec{a}$ and an ABox $\mathcal{E}$, is called a* conditional answer *to $\mathcal{Q}$ over $\mathcal{D}$ if*

1. *there is a Σ-model of $\mathcal{T}, \mathcal{D} \cup \mathcal{E}$,*
2. *$p(\vec{t}) \in \mathcal{E}$ implies $p(\vec{t}) \in \mathcal{D}$ for all $p \in \Sigma$,*
3. *there is substitution π such that: (a) $\pi(free(\varphi)) = \vec{a}$, (b) $\mathcal{E} \subseteq \mathcal{H}\pi$, and (c) $\mathcal{T}, \mathcal{D} \cup \mathcal{E} \models_\Sigma q\pi$.*

Conditional answers of AOMQCs capture typical scenarios where closed predicates are desirable for OMQs.

Example 3. *Let $\mathcal{Q} = (q(x), \mathcal{T}, \mathcal{H}, \Sigma)$ be an AOMQC with $q(x) = \exists y.(SkodaModel(x) \wedge hasEngine(x, y) \wedge ICEng(y))$, assumption patterns $\mathcal{H} = \{\exists hasEngine.SkodaEng(x)\}$, closed predicates $\Sigma = \{SkodaModel, SkodaEng\}$, and TBox $\mathcal{T} = \{DieselEng \sqsubseteq ICEng,\ PetrolEng \sqsubseteq ICEng\}$. Consider also a database:*
$\mathcal{D} = \{SkodaModel(\mathsf{sm17}), SkodaEng(\mathsf{se1}), SkodaEng(\mathsf{se2}), \mathcal{D} = \{\ DieselEng(\mathsf{se1}), PetrolEng(\mathsf{se2})\}$.

Evaluating $\mathcal{Q}$ over $\mathcal{D}$ produces as conditional answer $(\mathsf{sm17}, \{\exists hasEngine.SkodaEng(\mathsf{sm17})\})$. Intuitively, if we know that the only two Skoda engines are se1 *and* se2*, and both are internal combustion (IC) engines, then it suffices to assume that a Skoda model $x =$* sm17 *has a Skoda engine to infer that it has a IC engine.*

This example is a a minor adaptation of Example 1 in (Lutz, Seylan, and Wolter 2015). They consider a plain OMQ, and $\mathcal{T}$ contains also the axiom $SkodaModel \sqsubseteq \exists hasEngine.SkodaEng$. They argue that sm17 *is an intended answer, but their OMQ falls in a fragment that is not FO-rewritable, unlike the FO-rewritable AOMQC above.*

The remarkable feature of this use of closed predicates is that it does not cause the usual complexity increase: any AOMQC is FO-rewritable when the TBox is in *DL-Lite$_\mathcal{R}$*.

Rewriting with Closed Predicates

We present now an FO-rewriting for AOMQCs $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H}, \Sigma)$ with $\mathcal{T}$ a *DL-Lite$_\mathcal{R}$* TBox in normal form. Similarly as for AOMQs, we first apply the TBox rewriting in Proposition 2, and then rewrite w.r.t. assumption patterns. This step, however, is different if $\mathcal{H}$ has closed predicates.

Intuitively, now we cannot assume an arbitrary grounding of $\mathcal{H}$, but only groundings that respect the extensions of Σ in the input database. There is no canonical way to ground $\mathcal{H}$, and instead we need to iterate over all groundings and exclude those that are not valid w.r.t. Σ and the input data. However, they are determined by the finite number of possible groundings of the closed predicates, so we use an FO-query with universal quantification to iterate over them.

We denote by $terms_\Sigma(q)$ all terms $\vec{t} \in terms(q)$ such that $p(\vec{t}) \in q$ and $p \in \Sigma$.

Definition 11 (Rewriting w.r.t. $(\mathcal{H}, \Sigma)$)**.** *Let $\mathcal{H}$ be a set of assumption patterns, Σ a set of closed predicates and q a*

CQ. A rewriting of q w.r.t. $(\mathcal{H}, \Sigma)$, *written* $q \rightsquigarrow_{\mathcal{H},\Sigma} \varphi$, *has the following form:*

$$\forall \vec{y_1}, \ldots, \vec{y_k}.(p_1(\vec{y_1}) \wedge \cdots \wedge p_k(\vec{y_k}) \rightarrow q')$$

where $k \geq 0$, $\vec{y_i}$ *are fresh variables, and each* $p_i(\vec{y_i})$ *is such that* $p_i \in \Sigma$. *Such a rewriting is obtained by choosing:*

(i) a subset Γ *of the atoms of* q, *and a subset* $\mathcal{H}'$ *of* $\mathcal{H}$

(ii) a substitution h *that unifies* Γ *and* $\exp(\mathcal{H}')$, *and which has* $h(x) \in terms(\mathcal{H}') \cup terms_\Sigma(\mathcal{H}')$, *for each* $x \in keep(q, \Gamma)$,

and then doing the following. First, we add an atom $p_i(\vec{y_i})$ *for each* $p_i \in \Sigma$ *such that* $p_i(h(\vec{z})) \in \mathcal{H}'$ *for some* $\vec{z} \in vars(\Gamma)$, *and then to obtain* q', *we proceed as follows:*

1. *Replace by* $\top$ *every atom in* Γ.
2. *Add* $\bigwedge\{\vec{z} = \vec{y_i} \mid \vec{z} \in vars(\Gamma), p_i(h(\vec{z})) \in \exp(\mathcal{H}')\}$.
3. *Add* $\bigwedge\{x = h(x) \mid x \in keep(q, \Gamma) \cup vars(\mathcal{H}')\}$.
4. *Drop from the existential quantification all* $x \in vars(\mathcal{H}')$.

The set of φ *such that* $q \rightsquigarrow_{\mathcal{H},\Sigma} \varphi$ *is denoted* $hrew(q, \mathcal{H}, \Sigma)$.

The rewriting generalizes Definition 4, but the main difference is that we add to the universally quantified precondition each closed $p(x)$ in $\exp(\mathcal{H})$ that participates in the assumptions that make Γ true.

Example 4. *Let* $\Sigma = \{BusStop\}$ *be the set of closed predicates, and let* $\mathcal{Q} = (q(x), \emptyset, \{\exists locNext.BusStop(x)\}, \Sigma)$, *with* $q(x)$ *as in Example 1. In this case, since we can test for all bus stops if they are part of a central district,* $\mathcal{H}$ *is applicable. Our rewriting captures this as follows:*

- *choose* $\Gamma = \{locNext(x, y), BusStop(y)\}$, *let* $\exp(\mathcal{H}) = \{locNext(x, y'), BusStop(y')\}$ *and*
- *let* $h = \{x \mapsto x, y \mapsto y'\}$;

h *satisfies condition (ii), hence we obtain* φ*:*

$$\forall y'.(BusStop(y') \rightarrow \exists yz.VeganRest(x) \wedge partOf(y, z) \wedge CntDst(z) \wedge y = y').$$

We show next that the rewriting w.r.t. $(\mathcal{H}, \Sigma)$ is correct, analogously to the previous case.

Lemma 5. *Let* $\mathcal{Q} = (q(\vec{x}), \emptyset, \mathcal{H}, \Sigma)$ *be a DL-Lite*$_\mathcal{R}$ *AOMQC. For any database* $\mathcal{D}$, *any* $\mathcal{H}' \subseteq \mathcal{H}$ *and any substitution* π, *if* $(\pi(\vec{x}), \mathcal{H}'\pi) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$, *then there is some* φ *such that* $q(\vec{x}) \rightsquigarrow_{\mathcal{H},\Sigma} \varphi(\vec{x}, \vec{y})$ *and* $\pi(\vec{x}, \vec{y}) \in ans((\varphi, \emptyset, \Sigma), \mathcal{D})$.

Proof(Sketch). For an arbitrary $\mathcal{D}$ and $\mathcal{H}' \subseteq \mathcal{H}$, take any π such that $(\pi(\vec{x}), \mathcal{H}'\pi) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$. By definition we have that $\mathcal{D} \cup \mathcal{H}'\pi \models_\Sigma q\pi$. Intuitively, we can rely on a set of representative ground expansions of $\exp(\mathcal{H}'\pi)$ w.r.t. $(\Sigma, \mathcal{D})$ to homomorphically map q into each Σ-model of $\mathcal{D} \cup \mathcal{H}'\pi$. Such representatives disagree only on $vars_\Sigma(\exp(\mathcal{H}'\pi))$. Then, the part of q that is mapped by π over $\mathcal{D}$, is the same over all Σ-models. Therefore we obtain that there exist Γ, $\mathcal{H}'$ and h as in Definition 11. Let q' be obtained following steps 1-4 and let φ be of the form $\forall \vec{z_1}, \ldots, \vec{z_n}.(p_1(\vec{z_1}) \wedge \cdots \wedge p_k(\vec{z_k}) \rightarrow q')$, where $0 \leq k \leq n$ and $p_i \in \Sigma$ such that $p_i(h(\vec{z})) \in \exp(\mathcal{H}')$ for $\vec{z} \in vars(\Gamma)$. It follows then that $\mathcal{D} \models_\Sigma \pi(\varphi)$, hence $\pi(\vec{x}, \vec{y}) \in ans((\varphi, \emptyset, \Sigma), \mathcal{D})$. □

We show next that the rewriting w.r.t $(\mathcal{H}, \Sigma)$ is sound.

Lemma 6. *Let* $\mathcal{Q} = (q(\vec{x}), \emptyset, \mathcal{H}, \Sigma)$ *be a DL-Lite*$_\mathcal{R}$ *AOMQC and* $\varphi(\vec{x}, \vec{y})$ *a rewriting w.r.t.* $(\mathcal{H}, \Sigma)$. *Let* $\mathcal{D}$ *be a database consistent with* $\mathcal{T}$. *If there is a substitution* π *with* $\pi(\vec{x}, \vec{y}) \in ans((\varphi, \emptyset, \Sigma), \mathcal{D})$, *then for some* $\mathcal{H}' \subseteq \mathcal{H}$ $(\pi(\vec{x}), \mathcal{H}'\pi) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$.

Proof (Sketch). Fix $\mathcal{D}$, a rewriting $\varphi(\vec{x}, \vec{y})$ w.r.t. $(\mathcal{H}, \Sigma)$, and substitution π such that $\pi(\vec{x}, \vec{y}) \in ans((\varphi, \emptyset, \Sigma), \mathcal{D})$. By definition there is some $\mathcal{H}' \subseteq \mathcal{H}$ and $\Gamma \subseteq q$ that unify. Let $\exp_\sigma(\mathcal{H}'\pi)$ be the result of applying a grounding σ to $\exp(\mathcal{H}'\pi)$, so that $p(\vec{a}) \in \mathcal{D}$, for each $p(\vec{a}) \in \exp_\sigma(\mathcal{H}'\pi)$ with $p \in \Sigma$. Similarly to previous case, since the subsets of atoms chosen in step (i) are unifiable and due to equality atoms, we obtain that $\mathcal{D} \cup \exp_\sigma(\mathcal{H}'\pi) \models_\Sigma q\pi$, for each such σ. Lastly, for each Σ-model $\mathcal{I}$ of $\mathcal{D} \cup \mathcal{H}'\pi$ there is some σ such that $\mathcal{D} \cup \exp_\sigma(\mathcal{H}'\pi)$ is homomorphically mapped into $\mathcal{I}$. Hence, $(\pi(\vec{x}), \mathcal{H}'\pi) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$. □

The perfect rewriting is similar to the previous case, but using the rewriting w.r.t. $(\mathcal{H}, \Sigma)$ instead of w.r.t. $\mathcal{H}$ alone.

Definition 12 (Perfect Rewriting of AOMQC over $pos(\mathcal{T})$)**.** *Let* $\mathcal{Q} = (q(\vec{x}), pos(\mathcal{T}), \Sigma, \mathcal{H})$ *be a DL-Lite*$_\mathcal{R}$ *AOMQ with closed predicates. The* perfect rewriting of $\mathcal{Q}$, $Rew(\mathcal{Q})$ *is*

$$\{\varphi'(\vec{x'}) \mid \varphi'(\vec{x'}) \in hrew(q', \mathcal{H}, \Sigma) \wedge q'(\vec{x}) \in trew(q, \mathcal{T})\}.$$

Example 5. *For* $\mathcal{Q}$ *of Example 3,* $trew(\mathcal{Q})$ *equivalent to* $\exists y.(SkodaModel(x) \wedge hasEngine(x, y) \wedge (ICEngine(y)) \vee PetrolEngine(y)) \vee DieselEngine(y))$.
Then, $Rew(\mathcal{Q})$ *is equivalent to:*
$SkodaModel(x) \wedge \forall y'.(SkodaEngine(y') \rightarrow \exists y.y = y' \wedge (ICEngine(y) \vee PetrolEngine(y) \vee DieselEngine(y)))$.
Evaluating such query over the data, will allow us to easily construct the expected conditional answer.

Soundness of this rewriting follows from Lemmas 1, 5 and 6.

Disjointness assertions As we did above, we embed the consistency check into the perfect rewriting of an AOMQC.

Definition 13 (Perfect rewriting for general AOMQCs)**.** *Let* $\mathcal{T}$ *be a DL-Lite*$_\mathcal{R}$ *TBox. For a given set of assumption patterns* $\mathcal{H}$ *and a set of closed predicates* Σ, *the* set of inconsistent queries for $(\mathcal{H}, \mathcal{T}, \Sigma)$, *denoted by* $Rew_\perp(\mathcal{H}, \mathcal{T}, \Sigma)$ *is*

$$\bigcup_{\alpha \in neg(\mathcal{T})} \{q'' \mid \varphi \in hrew(q', \mathcal{H}, \Sigma),\ and\ q' \in trew(q_\alpha, \mathcal{T})\}.$$

Let $\mathcal{Q} = (q, \mathcal{T}, \mathcal{H}, \Sigma)$ *be a DL-Lite*$_\mathcal{R}$ *AOMQC and let* $\mathcal{Q}_{pos} = (q, pos(\mathcal{T}), \mathcal{T}, \mathcal{H}, \Sigma)$. *The* perfect rewriting of $\mathcal{Q}$, $Rew(\mathcal{Q})$ *is as follows:*

$$\bigcup_{\vec{y} \subseteq vars(\mathcal{H})} \{\neg Rew_\perp{}^{\vec{y}}(\mathcal{H}, \mathcal{T}, \Sigma) \wedge (Rew(Q_{pos}^{(\vec{x}, \vec{y})}) \vee Rew_\perp{}^{\vec{y}}(\mathcal{H}, \mathcal{T}, \Sigma)\}.$$

The perfect rewriting is similar to Definition 7, except that in this case the first conjunct checks if there are any groundings of $\mathcal{H}' \subseteq \mathcal{H}$ that are consistent w.r.t. $\mathcal{T}$ and Σ, and then the second one checks if $Rew(Q_{pos}^{(\vec{x}, vars(\mathcal{H}'))})$ holds for each such grounding. Analogously as before, we have the following lemma.

$\mathcal{Q}$ [$\#\mathcal{H}$]	$\#trew(\mathcal{Q})$	$\#Rew(\mathcal{Q})$	$\#cans_{min}(\mathcal{Q})$	Time (sec)			
				$ans(Rew(\mathcal{Q}))$	$cans_{min}(\mathcal{Q})$	**Test** $gr(\mathcal{H})$	**FedSPARQL**
q_1[3]	12	54	14415	0,6	1,1	40,4	42,7
q_2[4]	44	47	1603	0,1	0,2	4,8	6,6
q_3[4]	65	69	980	0,3	0,5	3,5	5,4
q_4[3]	14	64	60501	2,1	4,6	145,6	163,9
q_5[1]	3	6	8742	0,08	0,2	22	32,2

Table 1: Evaluation results for AOMQs over MyITS dataset.

Lemma 7. *Take a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$, a set of assumption patterns $\mathcal{H}$, and a set Σ of predicates. For any database $\mathcal{D}$ consistent with $\mathcal{T}$ and any substitution π ranging over $adom(\mathcal{D})$, we have $\pi(\vec{y}) \in ans(q_\bot, \mathcal{D})$ for some $q_\bot \in Rew_\bot(\mathcal{H}, \mathcal{T}, \Sigma)$ iff there is some $\mathcal{H}' \subseteq \mathcal{H}$ such that $\mathcal{D} \cup \pi(\mathcal{H}')$ is Σ-inconsistent with $\mathcal{T}$.*

We can now prove the correctness of our rewriting.

Theorem 3. *Let $\mathcal{Q} = (q(\vec{x}), \mathcal{T}, \mathcal{H}, \Sigma)$ be a DL-Lite$_\mathcal{R}$ AOMQC. For every database $\mathcal{D}$, the following are equivalent:*

- $(\vec{a}, \mathcal{E}) \in cans_{adom}(\mathcal{Q}, \mathcal{D})$.
- *There exists $\mathcal{H}' \subseteq \mathcal{H}$, $\varphi(\vec{x}, \vec{y}) \in Rew(\mathcal{Q})$, and a substitution π such that $\pi(\vec{x}) = \vec{a}$, $\mathcal{H}'\pi = \mathcal{E}$ and $\pi(\vec{x}, \vec{y})$ is an answer to the FO query φ over $\mathcal{I}_\mathcal{D}$.*

Empirical Evaluation

To demonstrate the potential usefulness of our approach, we developed a prototype implementation of the AOMQ rewriting. It was done in Java using Apache Jena 2.11 and Jena ARQ as SPARQL query engine, and tested on a MacBook Pro i5 2.7, Sierra OS.

We used the ontology, data, and a data generation tool from the MyITS project (Eiter, Krennwallner, and Schneider 2013; Eiter et al. 2015). The tool creates ABoxes with assertions for spatial relations like $locNext$ out of *OpenStreetMap* data, using parameters such as distance to create large sets of facts, in addition to other 'local' data (e.g., crowd-sourced restaurant data). Then one can pose queries that need both parts of data, as well as ontological reasoning, to get answers (e.g., hotels in residential areas close to a subway station).

Given the geospatial querying example in the introduction, MyITS is an relevant test case. Indeed, instead of creating large ABoxes, we may want to keep the access to spatial data remote. To simulate this scenario, we extracted some spatial relations and out-sourced their access via a SPARQL endpoint (using Jena Fuseki). This resulted in two sources: the local datasets with 227634 RDF triples, and a remote one with more than 2 million triples. We created 5 AOMQs based on test queries of Eiter et al. (2015), and treated spatial atoms as assumption patterns. In this way, we can query the local datasets and verify in the remote access point whether the spatial relations hold, only for the relevant candidates.

Table 1 shows for these queries the sizes of rewritings w.r.t. $\mathcal{T}$, and $(\mathcal{T}, \mathcal{H})$, and the size of $cans_{min}$, which gives a bound on the number of remote tests. We evaluated the time needed to answer the full rewriting over the local dataset, the time to construct the set $cans_{min}$, and the time to test remotely the spatial atoms (using SPARQL 'ask' queries). The results show that evaluating $Rew(\mathcal{Q})$ and constructing $cans_{min}(\mathcal{Q})$ is very efficient, while testing the assumptions remotely was more expensive, as expected. In practice, this delay may be amortized in many cases, e.g., if many queries share remote tests. As a sanity check, we compared the total time needed by our approach to posing a federated SPARQL query (W3C 2013) using both data sources. The latter approach was slower, even despite the fact that we disregarded ontological reasoning; naively posing the result of TBox rewriting as a federated query seems infeasible.

Related Work

Conditional answers were studied since the early nineties, e.g., to cope with incompleteness in disjunctive deductive databases (Demolombe 1992). They are related to the problem of evaluating queries in hypothetically updated states of databases (Gabbay et al. 1995; Christiansen and Andreasen 1998). Griffin and Hull (1997) rewrite hypothetical queries into equivalent usual queries which are evaluated using standard techniques. Recently, ten Cate et al. (2015) define so-called *why-not queries*, where an ontology is leveraged to obtain explanations of why tuples are not an answer, and study the complexity of obtaining most general explanations. We consider the shape of the explanations to be part of the input, while focusing on preserving worst-case (data) complexity. This work is most in line with the work of Calvanese et al. (2013), where *negative answers* are employed for describing a tuple of individuals and an associated explanation to why it is not an answer to the given query. Explanations there are ABoxes with assertions over concept and role names, while here this is generalized to sets of $\mathcal{ELI}$ atoms. Additionally, we allow for closed predicates.

Closed predicates are very useful to cope with partial incompleteness, but they can increase dramatically the complexity of reasoning. So far most results were negative, even for lightweight DLs (Lutz, Seylan, and Wolter 2013; 2015; Ngo, Ortiz, and Šimkus 2016).

Conclusions

We have introduced AOMQs, which are extensions of OMQs with assumption patterns, designed for leveraging information when querying incomplete databases. Answering AOMQs consists of computing conditional answers, which generally extend the answers of OMQs with tuples that are made true by the assumptions. In the case of *DL-Lite$_\mathcal{R}$*

AOMQs, they remain FO-rewritable even in the presence of closed predicates. A simple prototype for constructing and testing minimal conditional answers shows promising results, and suggests that this approach may be useful in scenarios when answering OMQs over all relevant data at once is costly or infeasible. For future work, it would be interesting to consider different shapes of assumption patterns, and to extend the evaluation with closed predicates.

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

References

Arenas, M., and Bertossi, L. 2002. Hypothetical temporal reasoning in databases. *Journal of Intelligent Information Systems* 19(2):231–259.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The dl-lite family and relations. *J. Artif. Int. Res.* 36(1):1–69.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press.

Bienvenu, M., and Ortiz, M. 2015. *Ontology-Mediated Query Answering with Data-Tractable Description Logics*. Cham: Springer International Publishing. 218–307.

Bienvenu, M.; Kikot, S.; Kontchakov, R.; Podolskii, V. V.; and Zakharyaschev, M. 2018. Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. *J. ACM* 65(5):28:1–28:51.

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

Calvanese, D.; Ortiz, M.; Šimkus, M.; and Stefanoni, G. 2013. Reasoning about explanations for negative query answers in dl-lite. *J. Artif. Intell. Res.* 48:635–669.

Christiansen, H., and Andreasen, T. 1998. A practical approach to hypothetical database queries. In Freitag, B.; Decker, H.; Kifer, M.; and Voronkov, A., eds., *Transactions and Change in Logic Databases*, 340–355. Berlin, Heidelberg: Springer Berlin Heidelberg.

Cruz-Filipe, L.; Gaspar, G.; and Nunes, I. 2019. Hypothetical answers to continuous queries over data streams. *CoRR* abs/1905.09610.

Demolombe, R. 1992. A strategy for the computation of conditional answers. In *ECAI*, 134–138.

Demolombe, R. 1998. Answers about validity and completeness of data: Formal definitions, usefulness and computation technique. In Andreasen, T.; Christiansen, H.; and Larsen, H. L., eds., *Flexible Query Answering Systems*, 138–147. Springer Berlin Heidelberg.

Eiter, T.; Pan, J. Z.; Schneider, P.; Šimkus, M.; and Xiao, G. 2015. A rule-based framework for creating instance data from openstreetmap. In *RR*, volume 9209 of *Lecture Notes in Computer Science*, 93–104. Springer.

Eiter, T.; Krennwallner, T.; and Schneider, P. 2013. Lightweight spatial conjunctive query answering using keywords. In *The Semantic Web: Semantics and Big Data*, 243–258. Springer Berlin Heidelberg.

Franconi, E.; Ibáñez-García, Y. A.; and Seylan, I. 2011. Query answering with dboxes is hard. *Electr. Notes Theor. Comput. Sci.* 278:71–84.

Gabbay, D.; Giordano, L.; Martelli, A.; and Olivetti, N. 1995. Hypothetical updates, priority and inconsistency in a logic programming language. In *Logic Programming and Nonmonotonic Reasoning*, 203–216. Springer Berlin Heidelberg.

Gottlob, G.; Kikot, S.; Kontchakov, R.; Podolskii, V. V.; Schwentick, T.; and Zakharyaschev, M. 2014. The price of query rewriting in ontology-based data access. *Artif. Intell.* 213:42–59.

Griffin, T., and Hull, R. 1997. A framework for implementing hypothetical queries. *SIGMOD Rec.* 26(2):231–242.

Lutz, C.; Seylan, I.; and Wolter, F. 2013. Ontology-based data access with closed predicates is inherently intractable(sometimes). In *IJCAI*, 1024–1030. IJCAI/AAAI.

Lutz, C.; Seylan, I.; and Wolter, F. 2015. Ontology-mediated queries with closed predicates. In *IJCAI*, 3120–3126. AAAI Press.

Ngo, N.; Ortiz, M.; and Šimkus, M. 2016. Closed predicates in description logics: Results on combined complexity. In *KR*, 237–246. AAAI Press.

ten Cate, B.; Civili, C.; Sherkhonov, E.; and Tan, W. 2015. High-level why-not explanations using ontologies. In *PODS*, 31–43. ACM.

W3C. 2013. Sparql 1.1 federated query. w3c recommendation 21 march 2013. https://www.w3.org/TR/sparql11-federated-query/.

Xiao, G.; Calvanese, D.; Kontchakov, R.; Lembo, D.; Poggi, A.; Rosati, R.; and Zakharyaschev, M. 2018. Ontology-based data access: A survey. In *IJCAI*, 5511–5519. ijcai.org.

Zhang, Y.; Chen, H.; Sheng, H.; and Wu, Z. 2007. Applying hypothetical queries to e-commerce systems to support reservation and personal preferences. In Desai, B. C., and Barker, K., eds., *Eleventh International Database Engineering and Applications Symposium (IDEAS 2007), September 6-8, 2007, Banff, Alberta, Canada*, 46–53. IEEE Computer Society.