Lightweight Spatial Conjunctive Query Answering using Keywords - Extended Version*

Thomas Eiter, Thomas Krennwallner, and Patrik Schneider

Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,tkren,patrik}@kr.tuwien.ac.at

Abstract. With the advent of publicly available geospatial data, ontology-based data access (OBDA) over spatial data has gained increasing interest. Spatio-relational DBMSs are used to implement geographic information systems (GIS) and are fit to manage large amounts of data and geographic objects such as points, lines, polygons, etc. In this paper, we extend the Description Logic DL-Lite with spatial objects and show how to answer spatial conjunctive queries (SCQs) over ontologies—that is, conjunctive queries with point-set topological relations such as *next* and *within*—expressed in this language. The goal of this extension is to enable an off-the-shelf use of spatio-relational DBMSs to answer SCQs using rewriting techniques, where data sources and geographic objects are stored in a database and spatial conjunctive queries are rewritten to SQL statements with spatial functions. Furthermore, we consider keyword-based querying over spatial OBDA data sources, and show how to map queries expressed as simple keyword lists describing objects of interest to SCQs, using a meta-model for completing the SCQs with spatial aspects. We have implemented our lightweight approach to spatial OBDA in a prototype and show initial experimental results using data sources such as Open Street Maps and Open Government Data Vienna from an associated project. We show that for real-world scenarios, practical queries are expressible under meta-model completion, and that query answering is computationally feasible.

1 Introduction

By the ever increasing availability of mobile devices, *location-aware* search providers are becoming increasingly commonplace. Search providers (e.g., Google Maps `http://maps.google.com/` or Nokia Maps `http://here.net`) offer the possibility to explore their surroundings for desired locations, also called *points-of-interest (POIs)*, but usually miss the possibility to express spatial relations (e.g., *next* and *within*). For an expressive location-aware search, the combination of Semantic Web techniques and spatial data processing (with spatial relations) is appropriate, given they provide a data backbone for spatial and taxonomic information to query semantically-enriched POIs.

To realize location-aware semantic search support, one needs to capture *categories* of POIs (e.g., Italian restaurant), their relations to additional *qualitative attributes* (e.g., having a guest garden). Further, one needs to capture the *spatial relations between*

* Supported by the Austrian Research Promotion Agency (FFG) project P828897, and the Austrian Science Fund (FWF) project P20840.

$$
\begin{array}{rlrlrl}
Shop & \sqsubseteq SpatialFeat & hasOp & \sqsubseteq hasQVal & Op & \sqsubseteq QVal \\
\exists hasQVal^{-} & \sqsubseteq SpatialFeat & Shop & \sqsubseteq \exists hasOp & Wlan & \sqsubseteq QVal \\
Park & \sqsubseteq SpatialFeat & \exists hasOp^{-} & \sqsubseteq Op & GuestGarden & \sqsubseteq QVal \\
Supermarket & \sqsubseteq Shop & QVal & \sqsubseteq \exists hasQVal & SpatialFeat & \sqsubseteq \neg Geometry \\
Walmart & \sqsubseteq Op & & & &
\end{array}
$$

Fig. 1: Ontology with integrated meta-model (TBox excerpt; role names start lowercase)

POIs (e.g., located inside a park). For modeling and interpreting a user's intention, it seems suggestive to use ontology languages and associated reasoning services. However, for spatial aspects we need to extend or combine them with spatial data reasoning. Furthermore, we must respect that ordinary users of location-aware search need a plain query interface; they are not experts in query languages, and an interface to express search intentions by lists of *keywords* in a Google-like manner would be desirable.

However, we face several obstacles for a seamless keyword-based querying and integration of geospatial data sources and ontologies. First, for a meaningful search result, we need to consider data obtained by integrating multiple data sources, which may be provided by autonomous vendors in heterogeneous formats (e.g., OpenStreetMap or Open Government Data data, a restaurant guide, etc). Using various data sources of substantial size gives the opportunity to find intended POIs, which may fall into multiple concepts ranging from rather generic to more detailed ones such as "restaurant" vs. "pizzeria." Moreover, we can exploit the structure of the taxonomic information that is implicit in the data sources by making it concrete in an ontology. Such *ontology-based data access* can be used to answer broad queries like "restaurants with Italian Cuisine," that should return pizzerias, trattorias, and osterias.

Second, from keyword-based input, we must generate meaningful formal queries to an ontology. In that, we must respect that the users may have no prior knowledge of the domain. Thus, we must be able to recognize and generate *relevant* combinations of possible keywords according to the ontology that represents the domain.

Third, as we query mainly spatial data, we need to capture spatial relations between different spatial objects and give users the possibility to use a fixed set of keywords to express them. For spatial querying answering, we must define an appropriate semantics and provide techniques that combine spatial with ontological query answering.

Fourth, a lot of research has been put into efficient query answering techniques over *lightweight ontology languages*, such as the DL-Lite family [7]. Conjunctive query (CQ) evaluation over DL-Lite ontologies can be delegated, by *first-order query rewriting*, to a Relational Database Management System (RDBMS), which facilitates scalable query processing. To secure this for an extension with spatial reasoning, the first-order rewritability of the latter is desirable. Furthermore, as first-order rewritings of queries might get prohibitively large in general (a known feature), also issues of manageable query generation from keywords must be respected.

We address the above issues with the following approach outlined in a nutshell.

- Various data sources are integrated via a global schema represented by an DL-Lite$_R$ ontology that is enriched with spatial information. The ontology-based knowledge base (KB) is separated into a TBox, an ABox with *normal* individuals and a spatio-relational database with *spatial objects*. We apply a mild extension to DL-Lite$_R$ by associating individuals to spatial objects by a predefined binding. A preprocessor creates this binding using a domain-specific heuristic (which is not considered here).

• The enriched ontology can be accessed, at the system level, by *spatial conjunctive queries (SCQ)*, which extend conjunctive queries with spatial predicates (e.g. intersects). In such queries, individuals can be located with spatial objects whose relationships are determined. By rewriting techniques, and in exploiting the $PerfectRef$ algorithm [7], SCQs can be rewritten to a union of conjunctive queries (UCQ). Under certain syntactic conditions, a 2-stage evaluation—evaluation of the ontology part of the query (over the ABox, which is stored in an RDBMS) followed by filtering via spatial conditions—is possible, which makes this approach attractive for practical realization.

• For keyword-based query answering, concepts of the ontology are labeled with keywords. On query evaluation, the keywords which the user enters are mapped to concepts and roles from the ontology; an *auto-completion* service aids the user to compensate lack of domain knowledge. Based on the keyword structure, a feasible CQ is generated and extended with spatial predicates to SCQs; in that, we use a specific *meta-model* that is stored in the ontology. Fig. 1 shows an excerpt of the ontology; the concept $SpatialFeat$ intuitively says that the individual has spatial features, which is extended by the subroles of $hasQVal$ with qualitative values, which are asserted to subconcepts of $QVal$. Furthermore, the individual is represented by a geometry, asserted to subconcepts of $Geometry$. However, also normal role assertions for qualitative attributes are considered (e.g., a restaurant with a guest garden).

We have implemented this approach in an experimental prototype, which is part of a more extensive system for smart, semantically enriched route planning system (MyITS, `http://myits.at/`) over real world data sources such as OpenStreetMap (OSM), Open Government Data (OGD) of Vienna, and the *Falter* restaurant guide for Vienna. The data sources are integrated by a global schema represented by an ontology expressed in DL-Lite$_R$. It turns out that naively generated UCQs may be too large for execution on conventional RDBMS. We thus improved our approach by exploiting the structure of the TBox in an *optimized* generation of queries from keyword, to eventually obtain smaller UCQs. First experiments show that this approach is feasible in a real-world scenario. Furthermore, we show that the optimizations described are important for feasibility. An extended version of this paper provides more details that are omitted for space reasons.[1]

2 Preliminaries

We adopt DL-Lite$_R$ [7] as the underlying ontological language and introduce an approach in which the FO-rewriting of $PerfectRef$ (see [7] and [6] for details) is strictly separated from spatial querying. As a result of this separation, we only allow spatial predicates (e.g., $Contains$) on the top level of the query structure. Regarding the semantics, we following partly the ideas of [15], but focus primarily on query answering (not solely satisfiability). Furthermore, we use a different notion for spatial relations.

Point-Set Topological Relations. We follow the point-set topological relation model in [13], where spatial relations are defined in terms of pure set theoretic operations. The

[1] `http://www.kr.tuwien.ac.at/staff/patrik/ESWC2013Ext.pdf`

realization of spatial objects is based on a set $P_E \subseteq \mathbb{R}^2$ of points in the plane; the (names of) spatial objects themselves are in a set Γ_S. While the set of points for a spatial object s is infinite (unless it is a point), it can be finitely defined by an associated *admissible geometry* $g(s)$. The geometries are defined by sequences $p = (p_1, \dots, p_n)$ of points that induce a point ($n = 1$), a line segment ($n = 2$), a line ($n > 2$), or a polygon. All points used for admissible geometries are from a finite set $P_F \subseteq P_E$ of points.

Spatio-relational Database. Thus, we define a *spatio-relational database* over Γ_S as a pair $\mathcal{S} = (P_F, g)$ of a point set $P_F \subseteq \mathbb{R}^2$ and a mapping $g : \Gamma_S \to \bigcup_{i \geq 1} {P_F}^i$.

The extent of a geometry p (full point set) is given by the function $points(p)$ and is a (possibly infinite) subset of P_E. For a spatial object s, we let $points(s) = points(g(s))$. We need $points$ to evaluate the spatial relations of two spatial objects via their respective geometries. For our spatio-thematic KBs, we consider the following types of admissible geometries p over P_F (with their representation), and let $P_E = \bigcup_{s \in \Gamma_S} points(s)$: a

- *point* is a sequence $p = (p_1)$, where $points(p_1) = \{p_1\}$;
- *line segment* is a sequence $p = (p_1, p_2)$, and $points(p) = \{\alpha p_1 + (1 - \alpha)p_2 \mid \alpha \in \mathbb{R}, 0 \leq \alpha \leq 1\}$;
- *line* is a sequence $p = (p_1, \dots, p_n)$ of line segments (p_i, p_{i+1}), $1 \leq i < n$, the first (p_1, p_2) and last (p_{n-1}, p_n) segments do not share an end-point, and $points(p) = \bigcup_{i=1}^{n-1} points(p_i)$;
- *polygon* is like a line but (p_1, p_2) and (p_{n-1}, p_n) share an end point; we have $points(a) = \bigcup_{i=1}^{n-1} points(p_i) \cup int(l_c)$, where $int(l_c)$ is the interior built from the separation of P_E by p into two disjoint regions.

Some $s \in \Gamma_S$ may serve to define via g a bounding box. We omit more complex geometries like areas or polygons with holes. Based on $points(x)$, we can define the spatial relation of point-sets in terms of *pure set operations*:

- $Equals(x, y)$: $points(x){=}points(y)$ and $NotEquals(x, y)$: $points(x){\neq}points(y)$;
- $Inside(x, y)$: $points(x){\subseteq}points(y)$ and $Outside(x, y)$: $points(x){\cap}points(y){=}\emptyset$;
- $Intersect(x, y)$: $points(x) \cap points(y) \neq \emptyset$ and $NextTo(x, y)$: $b(x) \cap\ b(y) \neq \emptyset$, where $b(z) = \{a \in P_E \mid dist(a, points(z)) \leq d_B\}$ for a distance function $dist\colon \mathbb{R}^2 \to \mathbb{R}_0^+$ and a distance value $d_B \in \mathbb{R}$.

Now for any spatial relation $S(s, s')$ and $s, s' \in \Gamma_S$, holds on a spatio-relational DB $\mathcal{S}$, written $\mathcal{S} \models S(s, s')$, if $S(g(s), g(s')$ evaluates to true. Relative to $points$ and $dist$ (and d_B), this is easily captured by a first-order formula over $(\mathbb{R}^2, \leq)$, and with regard to geo-spatial RDBMS trivially first-order expressible.

Note that the space model of [13] differs from the more detailed *9-Intersection model* (DE-9IM) of [10], which considers strict separation of the interior and object boundary; this leads to 9 instead of 5 spatial relations. We also omit spatial predicates in the signature, assuming a "standard" point-set interpretation of the spatial-relations [13]. Our approach is modular and flexible enough to allow further relations (e.g., $connects$) or use other interpretations such as DE-9IM.

Syntax and Semantics of DL-Lite$_R$. We recall the definitions from [7]. Consider a vocabulary of individual names Γ_I, *atomic concepts* Γ_C, and *atomic roles* Γ_R. Given atomic concepts A and atomic roles P, we define *basic* concepts B and *basic* roles R, *complex* concepts C and *complex* role expressions E, and P^- be the *inverse* of P as

$$B ::= A \mid \exists R \qquad C ::= B \mid \neg B \qquad R ::= P \mid P^- \qquad E ::= R \mid \neg R \ .$$

A DL-Lite$_R$ *knowledge base* is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where the TBox $\mathcal{T}$ consists of a finite set of *inclusion assertions* of the form $B \sqsubseteq C$ and $R \sqsubseteq E$, and the ABox $\mathcal{A}$ is a finite set of *membership assertions* on atomic concepts and on atomic roles of the form $A(a)$ and $P(a, b)$, where a and b are individual names.

The semantics of DL-Lite$_R$ is given in terms of FO interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a nonempty domain and $\cdot^{\mathcal{I}}$ an *interpretation function* such that $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for all $a \in \Gamma_I$, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for all $A \in \Gamma_C$, $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all $P \in \Gamma_R$, and $(P^-)^{\mathcal{I}} = \{(a_2, a_1) \mid (a_1, a_2) \in P^{\mathcal{I}}\}$; $(\exists R)^{\mathcal{I}} = \{a_1 \mid \exists a_2 \in \Delta^{\mathcal{I}} \text{ s.t. } (a_1, a_2) \in R^{\mathcal{I}}\}$; $(\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$; and $(\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$.

The notions of satisfaction of inclusion axioms and assertions, TBox and ABox resp. knowledge base is as usual, as well as logical implication; both are denoted with $\models$. We assume the *unique name assumption* holds for different individuals and values.

Checking satisfiability of DL-Lite$_R$ ontologies is *first-order (FO) rewritable* [7], i.e., for all $\mathcal{T}$, there is a Boolean FO query $Q_{\mathcal{T}}$ (constructible from $\mathcal{T}$) s.t. for every $\mathcal{A}$, $\mathcal{T} \cup \mathcal{A}$ is satisfiable iff $DB(\mathcal{A}) \not\models Q_{\mathcal{T}}$, where $DB(\mathcal{A})$ is the *least Herbrand model* of $\mathcal{A}$.

3 DL-Lite$_R(S)$

In this section, we extend DL-Lite$_R$ with spatial objects to DL-Lite$_R(S)$. We present its syntax and semantics, a transformation of to DL-Lite, and show that satisfiability and conjunctive query answering over DL-Lite$_R(S)$ KBs are FO-rewritable.

Syntax. Let Γ_S and Γ_I be pairwise disjoint sets as defined above. A *spatio-thematic knowledge base* (KB) is defined as $\mathcal{L}_{\mathcal{S}} = \langle \mathcal{T}, \mathcal{A}, \mathcal{S}, \mathcal{B} \rangle$, where $\mathcal{T}$ (resp. $\mathcal{A}$) is defined as a DL-Lite$_R$ TBox (resp. ABox), $\mathcal{S}$ is a spatio-relational database, and $\mathcal{B} \subseteq \Gamma_I \times \Gamma_S$ is a partial function called the *binding from $\mathcal{A}$ to $\mathcal{S}$*, similar to [15]; we apply a mild extension to DL-Lite$_R$ by associating individuals from $\mathcal{A}$ to spatial objects from $\mathcal{S}$.

We assume $\mathcal{B}$ to be already given. Furthermore, we extend DL-Lite$_R$ with the ability to specify the *localization* of a concept. For this purpose, we extend the syntax with

$$C ::= B \mid \neg B \mid (loc\ A) \mid (loc_s\ A), \quad s \in \Gamma_S,$$

where A is an atomic concept in $\mathcal{T}$; intuitively, $(loc\ A)$ is the set of individuals in A that can have a spatial extension, and $(loc_s\ A)$ is the subset which have extension s.

Semantics. Our aim is to give a semantics to the localization concepts $(loc\ A)$ and $(loc_s\ A)$ such that a KB $\mathcal{L}_{\mathcal{S}} = \langle \mathcal{T}, \mathcal{A}, \mathcal{S}, \mathcal{B} \rangle$ can be readily transformed into an ordinary DL-Lite$_R$ KB $\mathcal{K}_{\mathcal{S}} = \langle \mathcal{T}', \mathcal{A}' \rangle$, using concepts $C_{\mathcal{S}_{\mathcal{T}}}$ and C_s for individuals with some spatial extension resp. located at s. Note that $C_{\mathcal{S}_{\mathcal{T}}}$ cannot be forced to be the union of all C_s, as this would introduce disjunction (this hinders the passing from the open to the closed world assumption, which is important for the FO-rewriting of DL-Lite).

An (DL-Lite$_R$) *interpretation of $\mathcal{L}_{\mathcal{S}}$* is a structure $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, b^{\mathcal{I}} \rangle$, where $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ is an interpretation of $\langle \mathcal{T}, \mathcal{A} \rangle$, and $b^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Gamma_S$ is a partial function that assigns some individuals a location, such that for every $a \in \Gamma_I$, $(a, s) \in \mathcal{B}$ implies $b^{\mathcal{I}}(a^{\mathcal{I}}) = s$.

We extend the semantics of the previous section with $(loc\ A)$, $(loc_s\ A)$, where A is an atomic concept in $\mathcal{T}$:

$$(loc\ A)^{\mathcal{I}} \supseteq \{e \in \Delta^{\mathcal{I}} \mid e \in A^{\mathcal{I}} \land \exists s \in \Gamma_S : (e, s) \in b^{\mathcal{I}}\},$$
$$(loc_s\ A)^{\mathcal{I}} = \{e \in \Delta^{\mathcal{I}} \mid e \in A^{\mathcal{I}} \land (e, s) \in b^{\mathcal{I}}\}.$$

The interpretation of complex concepts, satisfaction, etc. is then as usual. For example, $A \sqsubseteq (loc_s\ A)$ expresses that all individuals in A are located at s; $B \sqsubseteq (loc\ A)$ states that individuals in B can have a location if they are in A.

Transformation to DL-Lite$_R$. Let $C_{\mathcal{S}_\mathcal{T}}$ and C_s, for every $s \in \Gamma_S$, be fresh concepts. We transform $\mathcal{L}_\mathcal{S}$ to $\mathcal{K}_\mathcal{S} = \langle \mathcal{T}', \mathcal{A}' \rangle$, where $\mathcal{T}' = \tau(\mathcal{T}) \cup \mathcal{T}_\mathcal{S}$ and $\mathcal{A}' = \tau(\mathcal{A}) \cup \mathcal{A}_\mathcal{B}$, and

- $\tau(X)$ replaces each occurrence of $(loc\ A)$ and $(loc_s\ A)$ in X with $C_{\mathcal{S}_\mathcal{T}} \sqcap A$ and $C_s \sqcap A$, respectively, and splits $\sqcap$ up;
- $\mathcal{T}_\mathcal{S}$ represents generic localization information via concepts, and contains the axiom $C_s \sqsubseteq C_{\mathcal{S}_\mathcal{T}}$, and the constraints $C_s \sqsubseteq \neg C_{s'}$ for all $s \neq s' \in \Gamma_S$;
- $\mathcal{A}_\mathcal{B}$ represents the concrete bindings between $\mathcal{A}$ and Γ_S, and for every $(a, s) \in \mathcal{B}$, we add $C_s(a)$ in $\mathcal{A}_\mathcal{B}$. Note that we do not assert $\neg C_s(a)$ for $(a, s) \notin \mathcal{B}$, keeping the open world assumption for bindings.

For example, let A (resp. $C_{\mathcal{S}_\mathcal{T}}$) be the concept $Park$ (resp. $SpatialFeat$), cp be the spatial object of "City Park," and the polygon $poly\_cp$ representing cp's spatial extend. The KB has the assertions $Park \sqsubseteq (loc\ Park)$, $CityParkCafe \sqsubseteq (loc_{cp}\ Park)$, and $CityParkCafe(c)$. Then, the transformation produces $Park \sqsubseteq (SpatialFeat \sqcap Park)$, $CityParkCafe \sqsubseteq (C_{poly\_cp} \sqcap Park)$, $C_{poly\_cp} \sqsubseteq SpatialFeat$, and $C_{poly\_cp}(cp)$.

Note that $\mathcal{K}_\mathcal{S}$ is indeed a DL-Lite$_R$ ontology, by the syntactic restrictions on localization concepts. It is not hard to verify that the models of $\mathcal{L}_\mathcal{S}$ and $\mathcal{K}_\mathcal{S}$ with the same domain ($\Delta^\mathcal{I} = \Delta^{\mathcal{I}'}$) coincide on common concepts and roles as follows: (i) if $\mathcal{I} \models \mathcal{L}_\mathcal{S}$, then $\mathcal{I}' \models \mathcal{K}_\mathcal{S}$ where $C_s^{\mathcal{I}'} = \{e \in \Delta^\mathcal{I} \mid (e, s) \in b^\mathcal{I}\}$, $C_{\mathcal{S}_\mathcal{T}}^{\mathcal{I}'} = \bigcup_{s \in \Gamma_S} C^{\mathcal{I}'}$ $(= dom(b^\mathcal{I}))$; conversely, (ii) if $\mathcal{I}' \models \mathcal{K}_\mathcal{S}$, then $\mathcal{I} \models \mathcal{L}_\mathcal{S}$ where $b^\mathcal{I} = \{(e, s) \mid e \in C_s^{\mathcal{I}'}\}$ and $(loc\ A)^\mathcal{I} = C_{\mathcal{S}_\mathcal{T}}^{\mathcal{I}'} \cap A^{\mathcal{I}'}$. As an easy consequence of this correspondence, we obtain:

Proposition 1. *Satisfiability checking and CQ answering for ontologies in DL-Lite$_R(S)$ is FO-rewritable.*

As the models of $\mathcal{L}_\mathcal{S}$ and $\mathcal{K}_\mathcal{S}$ correspond, we can check satisfiability on $\mathcal{K}_\mathcal{S}$, i.e., a standard DL-Lite$_R$ KB. An ontology CQ q over $\mathcal{L}_\mathcal{S}$ is easily rewritten to a CQ over $\mathcal{K}_\mathcal{S}$.

4 Query Answering in DL-Lite$_R(S)$

We next define spatial conjunctive queries (SCQ) over $\mathcal{L}_\mathcal{S} = \langle \mathcal{T}, \mathcal{A}, \mathcal{S}, \mathcal{B} \rangle$. Such queries may contain ontology and spatial predicates. Formally, an SCQ $q(\mathbf{x})$ is a formula

$$Q_{O_1}(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge Q_{O_n}(\mathbf{x}, \mathbf{y}) \wedge Q_{S_1}(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge Q_{S_m}(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $\mathbf{x}$ are the *distinguished* variables and $\mathbf{y}$ are either *non-distinguished* (*bound*) variables or individuals from Γ_I. Each $Q_{O_i}(\mathbf{x}, \mathbf{y})$ is an atom for $\mathcal{T}$ and of the form $A(z)$ or $P(z, z')$, with z, z' from $\mathbf{x} \cup \mathbf{y}$; the atoms $Q_{S_j}(\mathbf{x}, \mathbf{y})$ are over the vocabulary for the spatial relations in Sec. 2 and of the form $S(z, z')$, with z, z' from $\mathbf{x} \cup \mathbf{y}$.

For example, $q(x) = Playground(x) \wedge Within(x, y) \wedge Park(y)$ is a SCQ which intuitively returns the playgrounds located in parks.

Semantics. Let $\mathcal{I} = \left\langle \Delta^\mathcal{I}, \cdot^\mathcal{I}, b^\mathcal{I} \right\rangle$ be an interpretation of $\mathcal{L}_\mathcal{S}$. A *match* for $q(\mathbf{x})$ in $\mathcal{I}$ is a function $\pi : \mathbf{x} \cup \mathbf{y} \to \Delta^\mathcal{I}$ such that $\pi(c) = c^\mathcal{I}$, for each constant c in $\mathbf{x} \cup \mathbf{y}$, and for each $i = 1, \ldots n$ and $j = 1, \ldots, m$, (i) $\pi(z) \in A^\mathcal{I}$, if $Q_{O_i}(\mathbf{x}, \mathbf{y}) = A(z)$; (ii) $(\pi(z), \pi(z')) \in P^\mathcal{I}$, if $Q_{O_i}(\mathbf{x}, \mathbf{y}) = P(z, z')$; and (iii) $\exists s, s' \in \Gamma_S : (\pi(z), s) \in$

$b^{\mathcal{I}} \wedge (\pi(z'), s') \in b^{\mathcal{I}} \wedge \mathcal{S} \models S(s, s')$, if $Q_{S_j}(\mathbf{x}, \mathbf{y}) = S(z, z')$. That is, for spatial predicates individuals must have (unique) spatial extensions and the relationship between them must hold.

Then, a tuple $\mathbf{c} = c_1, \ldots, c_k$ over Γ_I is an *answer* for $q(\mathbf{x})$ in $\mathcal{I}$, $\mathbf{x} = x_1, \ldots$, if $q(\mathbf{x})$ has some match π in $\mathcal{I}$ such that $\pi(x_i) = c_i$, $i = 1, \ldots, k$; furthermore, $\mathbf{c}$ is an answer for $q(\mathbf{x})$ over $\mathcal{L}_{\mathcal{S}}$, if it is an answer in every model $\mathcal{I}$ of $\mathcal{L}_{\mathcal{S}}$. The *result* of $q(\mathbf{x})$ over $\mathcal{L}_{\mathcal{S}}$, denoted $res(q(\mathbf{x}), \mathcal{L}_{\mathcal{S}})$, is the set of all its answers.

The semantic correspondence between $\mathcal{L}_{\mathcal{S}}$ and $\mathcal{K}_{\mathcal{S}} = \langle \mathcal{T}', \mathcal{A}' \rangle$ guarantees that we can transform $q(\mathbf{x})$ into an equivalent query over $\mathcal{L}_{\mathcal{S}}{}' = \langle \mathcal{T}', \mathcal{A}', \mathcal{S}, \mathcal{B} \rangle$ by replacing each spatial atom $S(z, z')$ in $q(\mathbf{x})$ with

$$\bigvee\nolimits_{s,s' \in \Gamma_S} (C_s(z) \wedge C_s(z') \wedge S(s, s')). \qquad (2)$$

The resulting formula is easily cast into form $uq(\mathbf{x}) = q_1(\mathbf{x}) \vee \cdots \vee q_l(\mathbf{x})$, i.e., a union of CQs $q_i(\mathbf{x})$. The answers of $uq(\mathbf{x})$ in an interpretation $\mathcal{I}'$ of $\mathcal{L}_{\mathcal{S}}{}'$ are the answers of all $q_i(\mathbf{x})$ in $\mathcal{I}'$, and $res(uq(\mathbf{x}), \mathcal{L}_{\mathcal{S}}{}')$ is defined in the obvious way. We then can show:

Proposition 2. *For every SQC $q(\mathbf{x})$ over $\mathcal{L}_{\mathcal{S}}$, $res(q(\mathbf{x}), \mathcal{L}_{\mathcal{S}}) = res(uq(\mathbf{x}), \mathcal{L}_{\mathcal{S}}{}')$.*

Hence, answering SCQs in DL-Lite$_R(S)$ ontologies is FOL-rewritable. In particular, for fixed $\mathcal{S}$, we can eliminate $S(s, s')$ from (2), which yields a pure ontology query. Alternatively, we can replace it with $S_{s,s'}(z)$, where $S_{s,s'}$ is a fresh concept, and add $C_s \sqsubseteq S_{s,s'}$ to the TBox $\mathcal{T}'$ iff $\mathcal{S} \models S(s, s')$, thus changing $\mathcal{S}$ more flexibly.

Spatial Conjunctive Query Evaluation. The above SCQ rewriting is exponential in the number of spatial atoms, which incurs limitations. However, *if no bounded variables occur in spatial atoms*, we can separate query answering into an ontology part and a spatial query part, which can be efficiently evaluated in two stages:

(1) evaluate the ontology part of the query $q(\mathbf{x})$ (i.e., drop all spatial atoms) over $\mathcal{L}_{\mathcal{S}}{}'$. For that we can apply the *standard* DL-Lite$_R$ query rewriting of $PerfectRef$ and evaluate the result over the ABox, stored in an RDBMS.

(2) filter the result of Step (1), by evaluating the formulas (2) on the bindings π for the distinguished variables $\mathbf{x}$ (which are mapped to individuals). For that, retrieve in Step (1) also all instances of C_s, for all $s \in \Gamma_S$.

Step (2) amounts to computing a *spatial join* $\bowtie_S$, for which (at least) different evaluation strategies exist. One strategy, denoted as O_D, relies entirely on the functions of a spatial-extended RDBMS. The other, denoted as O_I, relies on an internal evaluation of $\bowtie_S$, i.e., spatial relations, where the intermediate results are kept in-memory.

We have considered both strategies, restricting to *acyclic queries* (i.e., the query hypergraph is tree-shaped; see e.g. [12] for a definition). For such queries, join trees can be built, which can be processed in a bottom up manner. In doing so, we distinguish between ontology and spatial nodes, and actually interleave the DL-Lite$_R$ query rewriting (Step (1)) and spatial join checking (Step (2)). In the following, we describe the main steps of our query evaluation:

First, a join tree J_T is build from the SCQ. We refer to [12] for a discussion of efficient methods to do so.

Second, each node n_T in J_T is visited in a bottom-up left-to-right order. We distinguish two cases depending on the type of the node n_T:

(1) n_T is an ontological node with an atom of Q_O:
We keep track of all atoms, adding them to a set S_{Sub}. If the parent of n_T is a Q_S, we apply the rewriting of $PerfectRef$ on the conjunction of S_{Sub} and keep the result in a temporary relation called R_Sub (which is RDBMS view);
(2) n_T is a spatial node with an atoms of Q_S:
We process the *spatial join* $\bowtie_S$ of all children $n_{T_1}, \cdots, n_{T_n}$ in n_T using either strategy O_D or O_I:
   (a) For O_D, we use a classical join $\bowtie$ with the spatial relation as the selection condition. In this case, we utilize existing spatial functions of the RDBMS, where the optimization is left to the RDBMS. This leads to the case, that the whole join tree N_T is evaluated as a single large query (rewritten to a SQL expression) over $\mathcal{A}$ and $\mathcal{S}$.
   (b) For O_I, we evaluate every $n_{T_1}, \cdots, n_{T_n}$ in n_T separately and calculate the spatial relations in-memory. This strategy implies that no spatial functions of the RDBMS is used. However, the intermediate results have to be stored in-memory, as these results will be used in the higher level spatial joins of J_T.

Note that for strategy O_D, we rewrite the spatial atoms ($Contains$, $Within$, etc.) directly to corresponding functions (cf. [8] for details) of the spatial-extension of the RDBMS. The different strategies noticeably affect the performance (see Sec. 8).

5 From Keywords to Spatial Conjunctive Queries

In this section, we provide the details for the generation of SCQ from a *valid* sequence of keywords; We shall consider in the next section how such sequences are obtained in a controlled way, by *automatic completion* and checking *keyword combinations*.

We assume an ontology O_U, which has an associated *meta-model* for structuring the query generation (described below). The generation is realized in three steps. First, the keywords are mapped to concepts from O_U and to spatial predicates. Then, a set of completion rules (which regard the meta-model) is applied to the resulting sequence of atomic formulas. Finally the completed sequence is converted into a SCQ.

We assume that spatio-thematic KBs are labeled, i.e., they are of the form $\mathcal{L}_\mathcal{S} = \langle \mathcal{T}, \mathcal{A}, \mathcal{S}, \mathcal{B}, \mathcal{N} \rangle$, where $\mathcal{N}$ is a set of textual labels representing keywords. The labels of $\mathcal{N}$ are assigned by `rdfs:label` to the concepts of $\mathcal{T}$. Multiple labels can be assigned to a single element, which allows to have synonyms. Further, translations for keywords in different languages can be enabled by the assignments.

Meta-Model for Structured Query Generation. We require on the top level of the ontology in use a strict separation of the concepts for spatial features *SpatialFeat* (e.g., *Park*, *Restaurant*, etc.), qualitative values $QVal$ (e.g., operator *Op*, *Cuisine*, etc.), and *Geometry* (e.g., *Point*, *Polygon*, etc.). Since our approach is designed to query spatial objects, every query has to be related to some *SpatialFeat*, which is extended by the subroles of $hasQVal$ with qualitative values (asserted to $QVal$) and is represented by the role *hasGeometry* as a geometry (asserted to *Geometry*). By this separation on the top level (which also exists in GeoOWL `http://www.w3.org/2005/Incubator/geo/XGR-geo/`), we have a *meta-model*, which is then used for the generation of "meaningful"

(R1) If $C_1 \sqsubseteq$ *SpatialFeat* and $C_2 \sqsubseteq$ *QualAttribute* rewrite to $(C_1 \ \ hasQVal \ \ C_2)$;
(R2) If $C_1 \sqsubseteq$ *SpatialFeat*, $C_2 \sqsubseteq$ *QualAttribute*, $C_3 \sqsubseteq$ *QualAttribute* rewrite to $((C_1 \ \ hasQVal \ \ C_2) \ \ hasQVal \ \ C_3)$;
(R3) If $C_1 \sqsubseteq$ *QualAttribute* rewrite to $(SpatialFeat \ \ hasQVal \ \ C_1)$;
(R4) If $C_1 \sqsubseteq$ *QualAttribute* and $C_2 \sqsubseteq$ *QualAttribute* rewrite to $((SpatialFeat \ \ hasQVal \ \ C_1) \ \ hasQVal \ \ C_2)$;
(R5) If $E_1 \sqsubseteq$ *SpatialFeat* or E_1 is a SQ, $E_2 \sqsubseteq$ *SpatialFeat* or E_2 is SQ, and S is a spatial predicate rewrite to $((E_1) \ \ S \ \ E_2)$;
(R6) If $E_1 \sqsubseteq$ *SpatialFeat* or E_1 is a SQ, and $E_2 \sqsubseteq$ *SpatialFeat* or E_2 is SQ rewrite to $((E_1) \ \ NextTo \ \ E_2)$;

Table 1: Completion rules; the result of rules (R1)–(R4) is denoted as subquery (SQ)

queries. Any ontology used with our approach has to be structured according to the meta-model. Fig. 1 shows some axioms of the meta-model for a specific ontology.

Generation of SCQs from Keywords. The automatic completion and combination step produces a set of *valid* keyword sequences, from which one sequence $K = (k_1, k_2, \dots, k_n)$ is chosen (unless the user determines one). Each keyword k_i represents either a concept or a spatial predicate. We must connect all k_i according to the meta-model to obtain SCQs, which then evaluate to spatial objects.

The rewriting of K to a SCQ Q is based on three steps that resemble a transducer with a context-free (left-recursive) grammar and a set of completion rules. The latter are important, because even if the transducer generates syntactically correct queries, their results might not consist of spatial objects. E.g., we could have a query $ItalianCuisine(x)$, but the results $R =$ (*pizza, pasta, ...*) could not be located on a map. Therefore, we have to extend the query as follows: $Restaurant(x) \wedge hasCuisine(x, y) \wedge ItalianCuisine(y)$.

In the following, we describe the three steps in the rewriting of K in detail:

(1) We obtain a new sequence K' from the sequence K by replacing every keyword with either a concept from $\mathcal{T}$ or a predefined spatial predicate. We check whether the keywords are associated to a concept in $\mathcal{N}$, otherwise we ignore it.
(2) We apply the completion rules in Table 1 on K' in a left-to-right order until no rules are applicable, resulting in a sequence K''.
(3) We generate the query $q(\mathbf{x})$ from K'' according to the function
$$f(K'') = (\cdots((C_1(x_1) \wedge E_{1,1}(x_1, y_1) \wedge E_{1,2}(y_1)) \wedge \chi_2) \wedge \cdots) \wedge \chi_n$$
where $\chi_i = E_{i,1}(\vartheta(E_{i-1,1}), y_i) \wedge E_{i,2}(y_i)$ and C_1 is a concept atom; each $E_{i,1}$ is either empty, a role atom, or a spatial atom, and each $E_{i,2}$ is either empty or a concept atom; $\vartheta(E_{i,1})$ is x_i if $E_{i,1}$ is a spatial atom, and x_{i-1} if $E_{i,1}$ is a role atom. These assignments ensure that the spatial atoms are always related to the top concept, while role atoms are related to the next level in the query tree.

After these steps, we obtain a valid SPQ $q(\mathbf{x})$ for query evaluation (Sec. 4). For rules (R2)–(R4), Table 1 shows in fact a simplified version, as they could be extended to arbitrary sequences of $QualAttributes$. Furthermore, rule (R6) defines a *default* relationship, if two spatial features are queried. Rewriting them to a simple conjunction between $C_1(x)$ and $C_2(x)$ would often lead to empty results, as two identical objects assigned to different concepts do not often exist within geospatial data sources.

Example 1. Given the keywords $K = (\textit{italian cuisine, non-smoking, in, park})$, we apply the first step, where we replace every k_i with an associated concept C_i from $\mathcal{N}$: $K' = (\mathit{ItalianCuisine}, \mathit{NonSmoking}, \mathit{Within}, \mathit{Park})$. In the second step we apply the completion rules to obtain $K'' = (((\mathit{SpatialFeat}\ \mathit{hasQVal}\ \ \mathit{ItalianCuisine})\ \mathit{hasQVal}\ \mathit{NonSmoking})\ \ \mathit{Within}\ \mathit{Park})$. Finally we get a SCQ $q(x_1, x_2){=}f(K'')$ with

$$\mathit{SpatialFeat}(x_1) \wedge \mathit{hasQVal}(x_1, y_1) \wedge \mathit{ItalianCuisine}(y_1) \wedge \mathit{hasQVal}(x_1, y_2) \wedge \\ \mathit{NonSmoking}(y_2) \wedge \mathit{Within}(x_1, x_2) \wedge \mathit{Park}(x_2)\ \ .$$

6 Generating Keyword Sequences

Since our approach is designed to have a single text-field for the keyword entries, we aim to provide fast automatic completion, keywords detection, and keyword combination functions. If a user enters keywords on a user interface (UI), we guide her by automatic completion and by showing possible combinations compliant with the ontology. For that, we must take the structure of the KB into account. Furthermore, as many combinations may be compliant, a selection of "relevant" combinations must be provided.

As the need for very low response time (e.g., below 100ms) makes on-demand calculation from the KB infeasible, a *prefix index* is created offline that stores all possible prefixes for a label of $\mathcal{N}$. It amounts to a function $f_P(e)$ which maps a string e to all possible labels of $\mathcal{N}$, such that $\bigcup_{n \in \mathcal{N}}(\mathit{Pref}(e) \subseteq \mathit{Pref}(n))$.

For example, the labels $\mathcal{N} = \{pub, public, park\}$, f_P map *p*, *pub*, and *park* as follows: $\{p\} \to \{park, pub, public\}$, $\{pu\} \to \{pub, public\}$, $\{park\} \to \{park\}$.

Syntactic Connectivity of Concepts. As multiple keywords are entered, we need to determine which concepts are connectable. We use a notion of syntactic connectivity $\mathbf{C}$ based on the syntactic structure of the KB, which captures the connection between two concepts through subconcepts and roles, but also through a common subsumer. For two concepts, we follow the inclusion assertion and check whether they share a common subsumer denoted as C_S, excluding the top concept. As the KB is based on DL-Lite$_R$, we can capture the following inclusion assertions: (i) concept inclusion $M_C : C_1 \sqsubseteq C_2$, role hierarchies $M_H : R_1 \sqsubseteq R_2$; (ii) role membership which covers the range (resp. domain) of a role as $M_R : \exists R^- \sqsubseteq C$ (resp. $M_D : \exists R \sqsubseteq C$); and (iii) mandatory participation $M_P : C \sqsubseteq \exists R$. We deliberately do not consider disjoint concepts as $C_1 \sqsubseteq \neg C_2$ in the inclusions, and distinguish *direct* and *indirect* connections between two concepts.

A *direct connection* between concepts C_A and C_B exists, denoted $\phi_D(C_A, C_B)$, if a sequence $C_A \to_M \exists R_1 \to_M C_1 \to_M \exists R_1 \ldots C_n \to_M \exists R_n \to_M C_B$ exists, where $M = M_D \cup M_H \cup M_C \cup M_R \cup M_P$. Furthermore, an *indirect connection* between C_A and C_B exists, denoted $\phi_I(C_A, C_B)$, if $\phi_D(C_A, C_S) \wedge \phi_D(C_B, C_S)$ holds for some C_S.

Example 2. In the example Fig. 1, the concepts $Supermarket$ and Op are directly connected: $Supermarket \to_{M_C} \mathit{Shop} \to_{M_P} \exists \mathit{hasOp} \to_{M_R} \mathit{Op}$. On the other hand, *GuestGarden* and $Wlan$ are indirectly connected: $\mathit{GuestGarden} \to_{M_C} QVal \to_{M_P} \exists hasQVal \to_{M_R} \mathit{SpatialFeat} \leftarrow_{M_R} \exists hasQVal \leftarrow_{M_P} QVal \leftarrow_{M_C} Wlan$.

In general, several sequences that witness $\phi_D(C_A, C_B)$ resp. $\phi_I(C_A, C_B)$ exist.

Automatic Completion, Detection, and Combination of Keywords. As we get a sequence of entered strings $E = (e_1, e_2, \dots, e_n)$, we need several steps to create the completed keywords, as the strings could be prefixes or misspelled.

First, we obtain the set of labels $L \subseteq \mathcal{N}$ by applying the prefix function $f_P(e_i)$ for every $e_i \in E$. Second, we build several levels of labels $L_1, \cdots, L_m$ based on the size of the subsets of L. As every L_i has the subsets $L_{i,1}, \cdots, L_{i,o}$ of the same size, we check for every $L_{i,j}$, if every pair of concepts (assigned to the labels of $L_{i,j}$) is syntactically connected at least in one direction. If we have found a $L_{i,j}$ with connected concepts, we add all sets of L_i (which are connectable) to the results. This is done by concatenating the labels of every set of L_i and add them to the result strings. We refer to Algorithm 1 for a detailed description.

By introducing an iterative algorithm, we return the largest possible combinations of keywords, thus excluding misspelled strings. However, we have in the worst-case exponentially many connectivity checks in the lengths of E.

Algorithm 1: Create Keywords Combinations

Input: A sequence of words $E = (e_1, e_2, \dots, e_n)$
Output: Set of keyword combinations $K = \{k_1, k_2, \dots, k_n\}$

```
K ← ∅;
L ← get all possible keywords for E applying the prefix function ;
P ← build the power set from keywords L ;
while K = ∅ and P ≠ ∅ do
    P_i ← get the largest subset of P, if some have same size, take one by one;
    O_i ← ∅;
    foreach keyword k_j in set P_i with len(k_j) > 1 do
        O_i ← add the set of concepts assigned to k_j ;
    T_i ← add a pair ⟨C_A, C_B⟩ for each possible combinations of concepts in O_i ;
    combine ← True ;
    foreach element ⟨C_A, C_B⟩ in T_i do
        if ⟨C_A, C_B⟩ are not syntactic connected then
            combine ← False
    if combine = True then
        concatenate P_i and add to R ;
    else
        remove P_i from P
```

Example 3. Given $E = ($*rest, in, non-smok*$)$, we obtain the labels $L = \{$*restaurant, indian food, intl food, non-smoking*$\}$. The first level of L contains the sets $L_{1,1} = \{$*restaurant, indian food, non-smoking*$\}$ and $L_{1,2} = \{$*restaurant, intl food, non-smoking*$\}$. The concepts assigned to them are $C_{1,1} = \{\mathit{Restaurant}, \mathit{IndianCuisine}, \mathit{NonSmoking}\}$ and $C_{1,2} = \{\mathit{Restaurant}, \mathit{IntlCuisine}, \mathit{NonSmoking}\}$. Then, we check for $C_{1,1}$, if every pair $(C, C'), C \neq C' \in C_{1,1}$, is syntactically connected, and likewise for $C_{1,2}$. The first two pairs are directly connected and the last pair is indirectly connected by the common subsumer $\mathit{SpatialFeat}$. Hence, the concepts in $C_{1,1}$ (and in $C_{1,2}$) are connectable. Then, we concatenate $L_{1,1}$ (resp. $L_{1,2}$) and add the strings to the results.

7 Refinement of Conjunctive Query Generation

While FO-rewritability of CQ answering over DL-Lite$_R$ KBs implies tractable data complexity, the size of the rewriting can increase exponentially with the number of atoms in the input CQ. Empirical findings [20] are that queries with more than 5-7 atoms can lead to large UCQs (e.g., unions of thousands of CQs) which cannot be handled by current RDBMS. Similar problems emerge with our generated SCQ (Sec. 8). One reason is the completion step in the SCQ generation. The generated SCQ can be too *general*, as we complete the intermediate sequence K' (Sec. 5) with the concept $\mathit{SpatialFeat}$ and role $\mathit{hasQVal}$, which are at the top-level (by our meta-model) of an ontology.

The *refinement* O_Q of the completion step is applied on every ontological subquery of a SCQ of the form $S(x_1) \wedge R_1(x_1, y_1) \wedge C_1(y_1) \wedge \ldots \wedge R_n(x_{n-1}, y_n) \wedge C_n(y_n)$, where $S \sqsubseteq \mathit{SpatialFeat}$, $\{R_1 \ldots, R_n\} \sqsubseteq \mathit{hasQVal}$, and $\{C_1, \ldots, C_n\} \sqsubseteq \mathit{QualAttribute}$ holds. It is based on the following ideas:

- Reduce the concept and role hierarchies: every edge in a path of ϕ_D or ϕ_I is an inclusion assertion, which increases the size of the rewritten UCQ; in particular, role inclusions can cause an exponential blow up [7];
- keep connectivity: by choosing paths according to ϕ_I, we keep the domain, range, mandatory participation, regarding the roles connecting S to $\{C_1, \ldots, C_n\}$.

Before applying O_Q, note that so far, S is a *most common subsumer* different from the top concept with respect to ϕ_I; i.e., for every pairs $(S, C_1), \ldots, (S, C_i)$, $\phi_I(S, C_j)$ holds for all j and the sum of path lengths for $\phi_I(S, C_j)$ is maximal. Thus, we try to minimize the path lengths under the constraint that ϕ_I is fulfilled for all pairs $\phi_I(S, C_j)$.

Briefly, it works as follows. We start the refinement O_Q by taking every subconcept S_i of S. We choose a shortest path, say p_j, according to ϕ_I for every pair (S_i, C_j), $1 \leq j \leq n$, and we add up all path lengths $|p_j|$ to len_{S_i}. Finally, we choose the S_i with the lowest len_{S_i} as a replacement of S and $R_1 \ldots, R_n$, where the latter are replaced with the roles appearing on the shortest paths p_j for S_i. We refer to Algorithm 2 to give a more detailed view.

Example 4. Let $q(x_1)$ be $\mathit{SpatialFeat}(x_1) \wedge$ $\mathit{hasQVal}(x_1, y_1) \wedge$ $\mathit{ItalianCuisine}(y_1) \wedge$ $\mathit{hasQVal}(x_1, y_2) \wedge \mathit{NonSmoking}(y_2)$. For the pairs $(\mathit{Rest}, \mathit{ItalianCuisine})$ and $(\mathit{Rest}, \mathit{NonSmoking})$, we have a path p_1 of length 2 ($\mathit{Rest} \rightarrow \exists \mathit{hasCuisine} \rightarrow \mathit{ItalianCuisine}$) and another path p_2 of length 2 ($\mathit{Rest} \rightarrow \exists \mathit{provides} \rightarrow \mathit{NonSmoking}$). Hence, the refinement O_Q produces the optimized query $q'(x_1)$, as the original paths are both of length 3 and Rest is a subconcept of $\mathit{SpatialFeat}$: $\mathit{Rest}(x_1) \wedge \mathit{hasCusine}(x_1, y_1) \wedge \mathit{ItalianCuisine}(y_1) \wedge \mathit{provides}(x_1, y_2) \wedge \mathit{NonSmoking}(y_2)$.

We point out that after applying O_Q, we may lose *completeness* with respect to the original SCQ, as shown by the following example. Given a spatio-thematic KB containing ABox assertions $\mathit{Rest}(i_1)$, $\mathit{hasCuisine}(i_1, i_2)$, $\mathit{ItalianCuisine}(i_2)$, $\mathit{SpatialFeat}(i_3)$, $\mathit{hasQVal}(i_3, i_2)$, and $\mathit{ItalianCuisine}(i_2)$, such that $\mathit{hasCuisine}$ has defined domain Rest and range $\mathit{Cuisine}$. The query $q(x_1) = \mathit{SpatialFeat}(x_1) \wedge$ $\mathit{hasQVal}(x_1, y_1) \wedge$ $\mathit{ItalianCuisine}(y_1)$ evaluates to $\{i_1, i_3\}$. If we refine $q(x_1)$ to the SCQ $q'(x_1) = \mathit{Rest}(x_1) \wedge \mathit{hasCuisine}(x_1, y_1) \wedge \mathit{ItalianCuisine}(y_1)$, we just get $\{i_1\}$ as a result. Informally, completeness can be lost if the ABox assertions are very general. One way

Algorithm 2: Optimization O_Q

Input: S subconcept of $SpatialFeat$, $(R_1, \ldots, R_n)$ subroles of $hasQVal$, $(C_1, \ldots, C_n)$ subconcepts of $QualAttribute$, and TBox $\mathcal{T}$
Output: Concept S' and roles $(R'_1, \ldots, R'_n)$

```
len_T ← 0;
foreach S_i ⊑ S of T do
    sequence T ← ∅ and len_{S_i} ← 0 ;
    foreach C_j of (C_1, ..., C_n) do
        if φ_I(S_i, C_j) then
            path p_j ← get the shortest path between S_i and C_j keeping φ_I ;
            len_{S_i} ← len_{S_i} + length of p_j ;
            get (R_{j_1}, ..., R_{j_n}) from p_j with every role which is a subrole of hasQVal ;
            add (R_{j_1}, ..., R_{j_n}) to T ;
    if len_{S_i} < len_T then
        len_T ← len_{S_i}, S' ← S_i, and (R'_1, ..., R'_n) ← T ;
```

to keep completeness is thus to impose conditions on the ABox, which ensure that ABox assertions have to fulfill certain conditions.

8 Implementation and Experimental Results

We have implemented a prototype of our keyword-based query answering approach. It is developed in Java 1.6 and uses PostGIS 1.5.1 (for PostgreSQL 9.0) as spatial-extended RDBMS. For the FO-rewriting of DL-Lite$_R$, we adapted OWLGRES 0.1 [22] to obtain the *perfect rewriting* (with *PerfectRef*) of a CQ and the TBox. We evaluate spatial atoms in two different ways (Sec. 4), namely as O_D by using the query evaluation of PostGIS or as O_I as a built-in component of our query evaluation algorithm. For O_D, we use the PostGIS functions for evaluation, e.g., `ST_Contains(x, y)`, and for O_I, we apply the functions of the JTS Topology Suite (`http://tsusiatsoftware.net/jts`).

As part of a consortium with AIT Mobility Department (routing), Fluidtime (UI), ITS Vienna Region (data and routing), we have integrated our prototype for the keyword-based query answering in the MyITS system for intention-oriented route planning (`http://myits.at/`). Currently, the following services are available:

1. *Neighborhood routing*, where a user desires to explore the neighborhood for a keyword-based query; and
2. *Via routing*, where a route is calculated between a given origin-destination pair via some POI, which is dynamically determined by a keyword-based query.

Scenario. Our benchmarks are based on the usage scenarios of MyITS, which has a DL-Lite$_R$ geo-ontology with the following metrics: 324 concepts (with 327 inclusion assertions); 30 roles (with 19 inclusion assertions); 2 inverse roles; 23 (resp. 25) domains (resp. ranges) of roles; 124 *normal* individuals; a maximal depth of 7 (4) in the concept (role) hierarchy (`http://www.kr.tuwien.ac.at/staff/patrik/GeoConceptsMyITS-v0.9-Lite.owl`). For the *spatial* objects, we added and mapped the POIs of greater Vienna contained in OSM ($\approx$ 70k instances), in the Falter database ($\approx$ 3700 instances), and parts of the OGD Vienna data ($\approx$ 7200 instances). The annotation step created $\approx$18700 individuals, which lead to $\approx$18700 concepts and $\approx$26000 role assertions. The low annotation rate of 23% is related to the exclusion of some OSM POIs (e.g., benches, etc.) and the ongoing extensions of the mapping framework.

Experiments. We conducted our experiments on a Mac OS X 10.6.8 system with an Intel Core i7 2.66GHz and 4 GB of RAM. We increased `shared_buffers` and `work_mem` of PostgreSQL 9.0 to utilize available RAM. For each benchmark, the average of five runs for the query rewriting and evaluation time was calculated, having a timeout of 10 minutes, and a memout of 750 MB for each run. The results shown in Table 2 present runtime in seconds and query size (number of atoms in the CQ), and use —^{s} to denote DB errors (e.g., the stack depth limit of Postgres 9.0 is reached), —^{m} for Java heap space limit has been reached (750 MB), and —^{t} for timeout.

Benchmarks. We designed the first benchmark B_1 based on keywords to measure the refinement O_Q on CQ without spatial predicates. The queries used in B_1 are

Q_1: $(spar)$ matches individuals run by "Spar";
Q_2: $(guest\ garden)$ returns the individuals with a guest garden;
Q_3: $(italian\ cuisine, guest\ garden)$ retrieves individuals that serve italian cuisine (including Pizzerias, etc.) and have a guest garden;
Q_4: $(italian\ cuisine, guest\ garden, wlan)$ gives individuals of Q_3 that in addition provide WLAN; and
Q_5: $(italian\ cuisine, guest\ garden, wlan, child\ friendly)$ returns individuals of Q_4 that in addition are child-friendly.

As described above, the keywords are completed to SCQ prior to evaluation as described.

The benchmark B_2 aims at comparing the database (denoted O_D) and internal evaluation of spatial predicates (denoted O_I) under the refinement O_Q. Its queries are

Q_6: $(playground, within, park)$ returns playgrounds in a park;
Q_7: $(supermarket, next\ to, pharmacy)$ matches supermarkets next to a pharmacy;
Q_8: $(italian\ cuisine, guest\ garden, next\ to, atm, next\ to, metro\ station)$ gives individuals with Italian food and a guest garden, whereby these individuals are next to an ATM and a metro station. The nesting of the query is as previously defined $(((italian\ cuisine, guest\ garden), next\ to), \ldots, metro\ station)$; and
Q_9: $(playground, disjoint, park)$ retrieves playgrounds outside a park.

As the results in Table 2 show, the refinement O_Q is essential for feasibility. Without it, Java exceeds heap space limitation during *perfect rewriting* in most cases, and SQL queries become too large for the RDBMS. The ontology of our scenario is big, yet captures only a domain for cities using OSM, OGD Vienna, and Falter.

As ground truth we assume the unrefined query. We lost completeness only in Q_1; this is due to three objects, which were tagged in OSM as shops but not supermarkets. With respect to the benchmark queries, the OSM tagging and our (heuristic) mapping has a minor effect on the completeness. Further, the results for Q_2 to Q_5 reflect the fact that adding keywords extends the selectivity of the query (smaller results), but enlarges the UCQ considerably.

We were surprised by the large difference between internal and external evaluation of the spatial relations. We would have expected the external evaluation by the RDBMS

Table 2: Benchmark Results (Evaluation time in secs), unrefined results in parentheses

(a) Benchmark B_1

	Instances	Query Size	Time
Q_1	106 (109)	438 (2256)	1.66 (4.96)
Q_2	1623 (1623)	51 (2256)	1.23 (5.59)
Q_3	204 (—^{s})	28 (71712)	1.14 (—^{s})
Q_4	32 (—^{m})	56 (—^{m})	1.48 (—^{m})
Q_5	3 (—^{m})	112 (—^{m})	4.11 (—^{m})

(b) Benchmark B_2, time only with O_Q

	Instances	Query Size	Time	
			O_I	O_D
Q_6	93 (93)	2 (2)	1.54	19.3
Q_7	378 (378)	4 (4)	2.22	—^{t}
Q_8	26 (—^{s})	30 (71714)	3.37	—^{t}
Q_9	151 (151)	2 (2)	2.02	—^{t}

is more efficient. Rewritten SQL queries have a three-leveled nesting, which consists of spatial joins ($\bowtie_S$) on the first, unions ($\cup$) on the second, and normal joins ($\bowtie$) on third level. It seems that standard query evaluation and optimization (in Postgres 9.0) are overwhelmed by such complex structures.

9 Related Work and Conclusion

Regarding SCQ, the closest to our work is [18], where crisp results for the combination of FO-rewritability of DL-Lite combined with the RCC-family (which offers qualitative models of abstract regions in topological space) are provided. For more expressive DLs, Lutz et al. [17] introduced the notion of ω-admissibility, which allows the combination of $\mathcal{ALC}$ and $RCC8$ [19], for subsumption testing. In PelletSpatial [21], the authors implemented a hybrid combination of $\mathcal{SHOIN}$ and $RCC8$. We follow a different approach in which the spatial regions are considered as point sets as in [14, 15]. However, we focus on scalable query answering (without distance primitives) and the related implementation issues. In this way, we face similar challenges as recent Geo-SPARQL engines did (e.g., Strabon [16] and Parliament [3]). However, we have a stronger focus on *ontology-based data access* than on linked open-data (with an RDF data model).

Keyword-based search on the Semantic Web is a well-covered field of research. A necessarily incomplete list of relevant approaches is SemSearch [24], XXploreKnow [23], and QUICK [25] which are general purpose search engines. The KOIOS [4], DO-ROAM [9], and the system of [2] support (text-based) spatial queries using ontologies. Our approach differs from these systems regarding the expressivity of DL-Lite, with the addition of spatial querying; the use of a meta-model for suitable query generation; and a focus on gradual extendibility with new data sources.

In this paper, we presented an extension of DL-Lite$_R$ with spatial objects using point-set topological relations for query answering. The extension preserves FO-rewritability, which allows us to evaluate a restricted class of conjunctive queries with spatial atoms over existing spatio-relational RDBMS. Second, we provided a technique for the generation of spatial conjunctive queries from a set of keywords. For this, we introduced a combination of a meta-model and completion rules to generate "meaningful" queries. Third, we implemented a prototype and performed experiments to evaluate the applicability in a real-world scenario. From our point of view, the first results are encouraging, as the evaluation time appeared to be moderate (always below 5 secs). Furthermore, our keyword-based approach is easy to extend, the text-based input is lightweight, and it has a reasonable precision through auto-completion and keyword combinations. However, precision could be improved by more advanced query expansion techniques (cf. [11]).

Future research is naturally directed to variants and extensions of the presented ontology and query language. E.g., one could investigate how spatial conjunctive queries work over $\mathcal{EL}$ [1] or $\mathsf{Datalog}^{\pm}$ [5]. For our motivating application, the point set model was sufficient, but extending our approach with the DE-9IM model [10] would be appealing and introduce further spatial relations. Then, one could work on query expansion techniques and on refinement of query generation, in a way such that completeness is ensured. Finally, regarding the implementation, one could investigate the reason for the unexpected performance on very large queries with spatial functions and conduct further experiments on larger geospatial DBs, possibly comparing our approach to the mentioned Geo-SPARQL engines.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: IJCAI 2005. pp. 364–369. Morgan-Kaufmann Publishers (2005)
2. Baglioni, M., Masserotti, M.V., Renso, C., Spinsanti, L.: Improving geodatabase semantic querying exploiting ontologies. In: GeoS 2011. LNCS, vol. 6631, pp. 16–33. Springer, Heidelberg (2011)
3. Battle, R., Kolas, D.: Enabling the geospatial Semantic Web with Parliament and GeoSPARQL. Semantic Web Journal 3(4), 355–370 (2012)
4. Bicer, V., Tran, T., Abecker, A., Nedkov, R.: Koios: Utilizing semantic search for easy-access and visualization of structured environmental data. In: ISWC 2011. LNCS, vol. 7032, pp. 1–16. Springer, Heidelberg (2011)
5. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. Artificial Intelligence 193, 87–128 (2012)
6. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and databases: The DL-Lite approach. In: Reasoning Web 2009. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009)
7. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
8. Clementini, E., Sharma, J., Egenhofer, M.J.: Modelling topological spatial relations: Strategies for query processing. Computers & Graphics 18(6), 815 – 822 (1994)
9. Codescu, M., Horsinka, G., Kutz, O., Mossakowski, T., Rau, R.: DO-ROAM: Activity-oriented search and navigation with OpenStreetMaps. In: GeoS 2011. LNCS, vol. 6631, pp. 88–107. Springer, Heidelberg (2011)
10. Egenhofer, M.J., Franzosa, R.D.: Point set topological relations. International Journal of Geographical Information Systems 5(2), 161–174 (1991)
11. Fu, G., Jones, C.B., Abdelmoty, A.I.: Ontology-based spatial query expansion in information retrieval. In: OTM Conferences 2005, Part II. LNCS, vol. 3761, pp. 1466–1482. Springer, Heidelberg (2005)
12. Gottlob, G., Leone, N., Scarcello, F.: The complexity of acyclic conjunctive queries. Journal of the ACM 48(3), 431–498 (2001)

13. Güting, R.H.: Geo-relational algebra: A model and query language for geometric database systems. In: EDBT 1988. LNCS, vol. 303, pp. 506–527. Springer, Heidelberg (1988)
14. Haarslev, V., Lutz, C., Möller, R.: A description logic with concrete domains and a role-forming predicate operator. Journal of Logic and Computation 9(3), 351–384 (1999)
15. Kutz, O., Wolter, F., Zakharyaschev, M.: A note on concepts and distances. In: DL 2001. CEUR-WS, vol. 49 (2001)
16. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Strabon: A semantic geospatial DBMS. In: ISWC 2012. LNCS, vol. 7649, pp. 295–311. Springer, Heidelberg (2012)
17. Lutz, C., Milicic, M.: A tableau algorithm for description logics with concrete domains and general TBoxes. Journal of Automated Reasoning 38(1-3), 227–259 (2007)
18. Özçep, Ö.L., Möller, R.: Scalable geo-thematic query answering. In: ISWC 2012. LNCS, vol. 7649, pp. 658–673. Springer, Heidelberg (2012)
19. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: KR 1992. pp. 165–176. Morgan Kaufmann (1992)
20. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR 2010. pp. 290–300. AAAI Press (2010)
21. Stocker, M., Sirin, E.: Pelletspatial: A hybrid RCC-8 and RDF/OWL reasoning and query engine. In: OWLED 2009. Springer, Heidelberg (2009)
22. Stocker, M., Smith, M.: Owlgres: A scalable OWL reasoner. In: OWLED 2008. Springer, Heidelberg (2008)
23. Tran, T., Cimiano, P., Rudolph, S., Studer, R.: Ontology-based interpretation of keywords for semantic search. In: ISWC 2007. LNCS, vol. 4825, pp. 523–536. Springer, Heidelberg (2007)
24. Uren, V.S., Lei, Y., Motta, E.: Semsearch: Refining semantic search. In: ESWC 2008. LNCS, vol. 5021, pp. 874–878. Springer, Heidelberg (2008)
25. Zenz, G., Zhou, X., Minack, E., Siberski, W., Nejdl, W.: From keywords to semantic queries - incremental query construction on the semantic web. J. Web Semant. 7(3), 166–176 (2009)