Chapter 5
On the Top-k Retrieval Problem for Ontology-Based Access to Databases

Umberto Straccia

Abstract The chapter is a succinct summary on the problem of evaluating ranked top-k queries in the context of ontology-based access over relational databases. An ontology layer is used to define the relevant abstract concepts and relations of the application domain, while facts with associated score are stored into a relational database. Queries are conjunctive queries with ranking aggregates and scoring functions. The results of a query may be ranked according to the score and the problem is to find efficiently the top-k ranked query answers.

1 Introduction

Managing uncertainty and fuzziness is starting to play an important role in Semantic Web (SW) research, and has been recognised by a large number of research efforts in this direction (see, e.g., [59, 62] for a concise overview).

We recall for the inexpert reader that there has been a long-lasting misunderstanding in the literature of artificial intelligence and uncertainty modelling, regarding the role of probability/possibility theory and vague/fuzzy theory. A clarifying chapter is [21]. Specifically, under *uncertainty theory* fall all those approaches in which statements rather than being either true or false, are true or false to some *probability* or *possibility* (for example, "it will rain tomorrow"). That is, a statement is true or false in any world/interpretation, but we are "uncertain" about which world to consider as the right one, and thus we speak about e.g. a probability distribution or a possibility distribution over the worlds. On the other hand, under *fuzzy theory* fall all those approaches in which statements (for example,"the hotel is cheap") are true to some *degree*, which is taken from a truth space (usually [0, 1]). That is, an interpretation maps a statement to a truth degree, since we are unable to establish whether a

U. Straccia (✉)
ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, PI, Italy
e-mail: straccia@isti.cnr.it

statement is entirely true or false due to the involvement of vague concepts, such as "cheap" (we cannot always say whether a hotel is cheap or not). Here, we will focus on fuzzy logic only.

In the SW, the standard Semantic Web Languages (SWLs) such as *triple languages* RDF & RDFS [9], *conceptual languages* or *frame-based languages* of the OWL 2 family [43] and *rule languages* such as RIF [50] are playing a dominant role. It also emerges that often in SW contexts, data are typically very large and dominate the intentional level of the ontologies. Hence, in that case one could still accept reasoning, specifically query answering, that is exponential on the intentional part, but it is mandatory that reasoning and query answering is polynomial in the data size, i.e. in *data complexity* [67].

In this chapter, we will briefly discuss a relatively novel issue for SWLs with a huge data repository, namely the problem of *evaluating ranked top-k queries*. Usually, an answer to a query is a set of tuples that satisfy a query. Each tuple does or does not satisfy the predicates in the query. However, very often the information need of a user involves so-called *fuzzy predicates* [32]. For instance, a user may need: "Find *cheap* hotels *near* to the conference location". Here, *cheap* and *near* are scoring predicates. Unlike the classical case, tuples satisfy now these predicates to a degree. In the former case the degree depends, e.g., on the price, while in the latter case it depends e.g. on the distance between the hotel location and the conference location. Therefore, a major problem we have to face with in such cases is that now an answer is a set of tuples *ranked* according to their *degree*. This poses a non-negligible challenge in case we have to deal with a huge amount of data records. Indeed, virtually every tuple may satisfy a query with a non-zero degree and, thus, has to be ranked. Computing all these degrees, ranking them and then selecting the top-k ones is not feasible in practice for large size databases [32].

While there are many works addressing the top-k problem for vague queries in databases (cf. [10, 14, 22, 23, 30, 31, 34, 35, 39]), little is known for the corresponding problem in knowledge representation and reasoning and specifically for SWLs. For instance, [68] considers non-recursive logic programs in which the score combination function is a function of the score of the atoms in the body. The work [55] considers non-recursive logic programs as well, though the score combination function is more expressive and may consider so-called expensive fuzzy relations (the score may depend on the value of an attribute, see [14]). However, a score combination function is allowed in the query rule only. We point out that in the case of non-recursive rules, we may rely on a query rewriting mechanism, which, given an initial query, rewrites it, using rules and/or axioms of the KB, into a set of new queries until no new query rule can be derived (this phase may require exponential time relative to the size of the KB, but is polynomial in the size of the facts). The obtained queries may then be submitted directly to a top-k retrieval database engine. The answers to each query are then merged using the disjunctive threshold algorithm (DTA) given in [55]. The works [54, 56, 58, 63] (see also [61]) address the top-k retrieval problem for the description logic *DL-Lite/DLR-Lite* [7, 12], though recursion is allowed among the axioms. Again, the score combination function may consider expensive fuzzy relations. However, a score combination function is allowed in the query only. The work [60] shows an application of top-k retrieval to the case of multimedia information retrieval by relying on a fuzzy variant of *DLR-Lite*. [57] addresses the top-k retrieval for general (recursive) LPs and is closest to this work. [37] slightly extends [57] as it allows also DLR-Lite axioms to occur and tries to rely as much as possible on current top-k database technology. However, these two works exhibits incorrect algorithms, which have been corrected in [64]. In this latter work, it is additionally shown that we can smoothly extend the top-k problem to the top-k-n problem. This latter problem has been shown to be fundamental in electronic Matchmaking [47, 48]. Moreno et al. [45] uses a threshold mechanism in the query for reducing the amount of computation needed to answer a propositional query in a tabulation-based procedure for propositional multi-adjoint logic programs and, thus, does not address the top-k retrieval problem. Chortaras et al. and Damasio et al. [15, 16, 18, 19] propose query answering procedures, which are based on unification to compute answers, but do not address the top-k retrieval problem. It is unclear yet whether unification-based query driven query answering procedures can be combined with a threshold mechanism in such a way to compute top-k answers.

In this chapter, we will provide the basic notions and salient references to be known concerning the top-k retrieval problem for Semantic Web languages.[1]

In the following, we will proceed as follows. We overview briefly SWLs and relate them to their logical counterpart. Then, we briefly sketch a general framework for ontology-based access to databases, the related top-k retrieval problem and algorithmic solutions to this problem.

2 Semantic Web Languages: Overview

Semantic Web Languages (SWL) are standard languages used to provide a formal description of concepts, terms, and relationships within a given knowledge domain. There are essentially three family of languages: namely, *triple languages* RDF & RDFS [9] (*Resource Description Framework*), *conceptual languages* of the OWL 2 family (*Ontology Web Language*) [43] and *rule languages* of the RIF family (*Rule Interchange Format*) [50]. While their syntactic specification is based on XML [69], their semantics is based on logical formalisms, which will be the focus here: briefly,

- RDFS is a logic having intensional semantics and the logical counterpart is ρdf [41].
- OWL 2 is a family of languages that relate to *Description Logics* (DLs) [7].
- RIF relates to the *Logic Programming* (LP) paradigm [36].
- Both OWL 2 and RIF have an extensional semantics.

[1] By purpose we will neglect the details of these works (including description of algorithms and implementations).

2.1 RDF and RDFS

The basic ingredients of *RDF* are *triples*, which are of the form

$$(s, p, o) ,$$

such as $(umberto, likes, tomato)$, stating that *subject s* has *property p* with *value o*. In *RDF Schema* (RDFS), which is an extension of RDF, additionally some special keywords (subclass, subproperty, property domain and range and instance of specifications) may be used as properties to further improve the expressivity of the language. For instance we may also express that the class of 'tomatoes is a subclass of the class of vegetables', $(tomato, \mathsf{sc}, vegetables)$, while Zurich is an instance of the class of cities, $(zurich, \mathsf{type}, city)$.

From a computational point of view, one computes the so-called *closure* (denoted $cl(\mathscr{K})$) of a set of triples $\mathscr{K}$. That is, one infers all possible triples using inference rules [40, 41, 49], such as

$$\frac{(A, \mathsf{sc}, B), (X, \mathsf{type}, A)}{(X, \mathsf{type}, B)}$$

if A subclass of B and X instance of A then infer that X is instance of B,

and then store all inferred triples into a relational database to be used then for querying. Note that making all implicit knowledge explicit is viable due to the low complexity of the closure computation, which is $\mathscr{O}(|\mathscr{K}|^2)$ in the worst case.

2.2 OWL Family

The Web Ontology Language *OWL* [42] and its successor *OWL 2* [17, 43] are "object oriented" languages for defining and instantiating ontologies. An OWL ontology may include descriptions of classes, properties and their instances, such as

```
class Person partial Human
      restriction (hasName someValuesFrom String)
      restriction (hasBirthPlace someValuesFrom Geoplace)
```

The class Person is a subclass of class Human and has two attributes: hasName having a string as value, and hasBirthPlace whose value is an instance of the class Geoplace.

Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences. For example, if an individual Peter is an instance of the class Student, and Student is a subclass of Person, then one can derive that Peter is also an instance of Person in a similar way as it happens for RDFS. However, let us note that OWL is more expressive than RDFS, as the decision problems for OWL are in higher complexity classes [46] than for RDFS.

OWL 2 [17, 43] is an update of OWL 1 adding several new features, including an increased expressive power. OWL 2 also defines several *OWL 2 profiles*, i.e. OWL 2 language subsets that may better meet certain computational complexity requirements or may be easier to implement. The choice of which profile to use in practice will depend on the structure of the ontologies and the reasoning tasks at hand. The OWL 2 profiles are:

OWL 2 EL. It is particularly useful in applications employing ontologies that contain very large numbers of properties and/or classes (basic reasoning problems can be performed in time that is polynomial with respect to the size of the ontology [4]). The EL acronym reflects the profile's basis in the $\mathscr{EL}$ family of description logics [4].
OWL 2 QL. It is aimed at applications that use very large volumes of instance data, and where query answering is the most important reasoning task. In OWL 2 QL, conjunctive query answering can be implemented using conventional relational database systems. Using a suitable reasoning technique, sound and complete conjunctive query answering can be performed in LOGSPACE with respect to the size of the data (assertions) [3, 13]. The QL acronym reflects the fact that query answering in this profile can be implemented by rewriting queries into a standard relational Query Language such as SQL [66].
OWL 2 RL. It is aimed at applications that require scalable reasoning without sacrificing too much expressive power. OWL 2 RL reasoning systems can be implemented using rule-based reasoning engines as a mapping to *Logic Programming* [36], specifically *Datalog* [66], exists. The RL acronym reflects the fact that reasoning in this profile can be implemented using a standard rule language [24]. The computational complexity is the same as for Datalog [20] (polynomial in the size of the data, EXPTIME w.r.t. the size of the knowledge base).

2.3 RIF Family

The *Rule Interchange Format* (RIF) aims at becoming a standard for exchanging rules, such as

```
Forall?Buyer ?Item ?Seller
  buy(?Buyer ?Item ?Seller): −sell(?Seller ?Item ?Buyer)
```

Someone buys an item from a seller if the seller sells that item to the buyer

among rule systems, in particular among Web rule engines. RIF is in fact a family of languages, called *dialects*, among which the most significant are:

RIF-BLD. The *Basic Logic Dialect* is the main logic-based dialect. Technically, this dialect corresponds to Horn logic with various syntactic and semantic extensions. The main syntactic extensions include the frame syntax and predicates with named arguments. The main semantic extensions include datatypes and externally defined predicates.

RIF-PRD. The *Production Rule Dialect* aims at capturing the main aspects of various production rule systems. Production rules, as they are currently practiced in mainstream systems like Jess[2] or JRules,[3] are defined using ad hoc computational mechanisms, which are not based on a logic. For this reason, RIF-PRD is not part of the suite of logical RIF dialects and stands apart from them. However, significant effort has been extended to ensure as much sharing with the other dialects as possible. This sharing was the main reason for the development of the RIF Core dialect;

RIF-Core. The *Core Dialect* is a subset of both RIF-BLD and RIF-PRD, thus enabling limited rule exchange between logic rule dialects and production rules. RIF-Core corresponds to Horn logic without function symbols (i.e., Datalog) with a number of extensions to support features such as objects and frames as in F-logic [33].

RIF-FLD. The *Framework for Logic Dialects* is not a dialect in its own right, but rather a general logical extensibility framework. It was introduced in order to drastically lower the amount of effort needed to define and verify new logic dialects that extend the capabilities of RIF-BLD.

3 Ontology-Based Databases

We illustrate here a simple, though general enough framework to present the top-k retrieval problem for the various SWLs sketched in the previous section.

To start with, the scoring space is the set (n positive integer)

$$L_n = \left\{0, \frac{1}{n}, \ldots, \frac{n-2}{n-1}, 1\right\}.$$

Then a *Knowledge Base* (KB) $\mathscr{K} = \langle \mathscr{F}, \mathscr{O}, \mathscr{M} \rangle$ consists of a *facts component* $\mathscr{F}$, an *Ontology component* $\mathscr{O}$ and an *mapping component* $\mathscr{M}$, which are defined below. Informally, the facts component is the relational database in which the extensional data is stored, the ontology component is the intentional level and describes the important relations about the world we are modelling and the mapping component defines the bridge between the low-level database schema vocabulary and the abstract ontology vocabulary. While the facts and mapping component is syntactically equal for all SWLs, the ontology component is language dependent. The same applies for

[2] http://www.jessrules.com/

[3] http://www.ilog.com/products/jrules/

the query language. It is also assumed that relations occurring in $\mathscr{F}$ do not occur in $\mathscr{O}$ (so, we do not allow that database relation names occur in $\mathscr{O}$).

3.1 The Facts Layer

$\mathscr{F}$ is a finite set of expressions of the form

$$R(c_1, \ldots, c_n) \colon s \ ,$$

where R is an n-ary relation, every c_i is a constant, and s is a degree in L_n.

For each relation R, we represent the facts $R(c_1, \ldots, c_n) \colon s$ in $\mathscr{F}$ by means of a relational $n+1$-ary table T_R, containing the records $\langle c_1, \ldots, c_n, s \rangle$. We assume that there cannot be two records $\langle c_1, \ldots, c_n, s_1 \rangle$ and $\langle c_1, \ldots, c_n, s_2 \rangle$ in T_R with $s_1 \neq s_2$ (if there are, then we remove the one with the lower degree). Each table is sorted in descending order with respect to the degrees. For ease, we may omit the degree component and in such case the value 1 is assumed.

3.2 The Mapping Layer

The mapping component is a set of "mapping statements" that allow to connect classes and properties to physical relational tables. Essentially, this component is used as a wrapper to the underlying database and, thus, prevents that relational table names occur in the ontology. Formally, a *mapping statement* (see [63]) is of the form

$$R \mapsto (c_1, \ldots, c_n) \colon c_{score}.sql \ ,$$

where sql is a SQL statement returning n-ary tuples $\langle c_1, \ldots, c_n \rangle$ ($n = 1, 2$) with score determined by the c_{score} column. The tuples have to be ranked in decreasing order of score and, as for the fact component, we assume that there cannot be two records $\langle \mathbf{c}, s_1 \rangle$ and $\langle \mathbf{c}, s_2 \rangle$ in the result set of sql with $s_1 \neq s_2$ (if there are, then we remove the one with the lower score). The score c_{score} may be omitted and in that case the score 1 is assumed for the tuples. We assume that R occurs in $\mathscr{O}$, while all of the relational tables occurring in the SQL statement occur in $\mathscr{F}$.

3.3 An RDFS Ontology Layer

The ontology component is used to define the relevant abstract concepts and relations of the application domain by means of axioms and defines the so-called *intentional level*. To what concerns us here, we will consider the essential features of RDFS only and follow [25, 40, 41] by considering the "core" part of RDFS, called ρdf [41] (read rho-df, the ρ from restricted RDF). Specifically, $\mathcal{O}$ is a finite set of *axioms* having the form

$$(s, p, o) \; ,$$

where p is a property that may belong to ρdf^-:

$$\rho\text{df}^- = \rho\text{df} \setminus \{\mathsf{type}\} = \{\mathsf{sp}, \mathsf{sc}, \mathsf{dom}, \mathsf{range}\} \; ,$$

The keywords in ρdf^- may be used in triples as properties. Informally,

- (p, sp, q) means that property p is a *sub-property* of property q;
- (c, sc, d) means that class c is a *subclass* of class d;
- (p, dom, c) means that the *domain* of property p is c; and
- (p, range, c) means that the *range* of property p is c.

Note that we don't allow the use of ρdf triples of the form (a, type, b) with meaning "*a* is of *type b*" in the intentional level.

3.4 An OWL 2 Ontology Layer

The OWL 2 ontology layer $\mathcal{O}$ consists of a finite set of OWL 2 class and property axioms (see, [43]) of the form

$$C \sqsubseteq D$$
$$R \sqsubseteq P$$

where C, D are OWL 2 classes (*concepts*) and R and P are OWL 2 properties . The intuition of an axiom of the form $C \sqsubseteq D$ (resp. $R \sqsubseteq P$) is that any instance of class C (resp. role R) is an instance of class D (resp. P) as well. For computational reasons (to have a query answering procedure that is worst case polynomial), we will consider here specifically the case of the logical counterpart of the three main OWL 2 profiles, namely OWL 2 EL, OWL 2 QL and OWL 2 RL only.

3.4.1 The Case of OWL 2 EL

The importance of the $\mathcal{EL}$ DL family [4, 6, 26] is due to the fact that it is the logical counterpart of the OWL 2 EL profile [44], i.e. OWL 2 EL constructs can be mapped into the DL $\mathcal{EL}^{++}$(d). We recall that it enables polynomial time algorithms for all the standard reasoning tasks[4] and, thus, it is particularly suitable for applications where very large ontologies are needed.

For illustrative purposes, we consider here the following sublanguage of $\mathscr{E}\mathscr{L}^{++}$(d) together with its FOL reading [4–6]:

Concept expressions	**FOL-reading**
A	$A(x)$
$C \sqcap D$	$C(x) \wedge D(x)$
$\exists R.C$	$\exists y.R(x,y) \wedge C(x)$
Axioms	**FOL-reading**
$C \sqsubseteq D$	$\forall x.C(x) \Rightarrow D(x)$
$R_1 \circ R_2 \sqsubseteq R$	$\forall x \forall y \forall z.R_1(x,z) \wedge R_2(z,y) \Rightarrow R(x,y)$
$\mathsf{dom}(R) \sqsubseteq C$	$\forall x.R(x,y) \Rightarrow C(x)$
$\mathsf{ran}(R) \sqsubseteq C$	$\forall y.R(x,y) \Rightarrow C(y)$
$\mathsf{ref}(R)$	$\forall x.R(x,x)$.

3.4.2 The Case of OWL 2 QL

The importance of the DL-Lite DL family [3, 11–13] is due to the fact that it is the logical counterpart of the OWL 2 QL profile [44], i.e.OWL 2 QL constructs can be mapped into the DL DL-Lite$_R$(d), which, we recall, was designed so that sound and complete query answering is in LOGSPACE (more precisely, in AC^0) with respect to the size of the data, while providing many of the main features necessary to express conceptual models such as UML class diagrams and ER diagrams.

We next recap succinctly the DL-Lite DL family [13]. We start with the language DL-Lite$_{core}$ that is the core language for the whole family. Concepts and roles are formed according to the following syntax (A is an atomic concept, P is an atomic role and P^- is its inverse):[5]

$$\begin{aligned} B &\longrightarrow A \mid \exists R \\ C &\longrightarrow B \mid \neg B \\ R &\longrightarrow P \mid P^- \\ E &\longrightarrow R \mid \neg R\ . \end{aligned}$$

B denotes a *basic concept*, that is, a concept that can be either an atomic concept or a concept of the form $\exists R$, where R denotes a *basic role*, that is, a role that is either an atomic role or the inverse of an atomic role. C denotes a concept, which can be a basic concept or its negation, whereas E denotes a role, which can be a basic role or its negation. Sometimes we write $\neg C$ (resp., $\neg E$) with the intended meaning that

[4] That is, the ontology satisfiability problem, the subsumption problem and the instance checking problem.

[5] The FOL-reading of concept $\exists R$ is: set of x such that $\exists y.R(x,y)$.

$\neg C = \neg A$ if $C = A$ (resp., $\neg E = \neg R$ if $E = R$), and $\neg C = A$, if $C = \neg A$ (resp., $\neg E = R$, if $E = \neg R$).[6]

Inclusion axioms are of the form

$$B \sqsubseteq C$$

We might include $B_1 \sqcup B_2$ (FOL-reading: set of x such that $B_1(x) \vee B_2(x)$) in the constructs for the left-hand side of inclusion axioms and $C_1 \sqcap C_2$ in the constructs for the right-hand side. In this way, however, we would not extend the expressive capabilities of the language, since these constructs can be simulated by considering that $B_1 \sqcup B_2 \sqsubseteq C$ is equivalent to the pair of assertions $B_1 \sqsubseteq C$ and $B_2 \sqsubseteq C$, and that $B \sqsubseteq C_1 \sqcap C_2$ is equivalent to $B_1 \sqsubseteq C_1$ and $B \sqsubseteq C_2$. Similarly, we might add $\bot$ to the constructs for the left-hand side and $\top$ to those for the right-hand side.

Eventually, DL-Lite$_R$ is now obtained by extending DL-Lite$_{core}$ with the ability of specifying inclusion axioms between roles of the form

$$R \sqsubseteq E.$$

where R and E are defined as above.

3.4.3 The Case of OWL 2 RL

The importance of the Horn-DL family [24, 65] is due to the fact that it is the logical counterpart of the OWL 2 RL profile [44], i.e.OWL 2 RL constructs can be mapped into Horn-DL. This is achieved by defining a syntactic subset of OWL 2, which is amenable to implementation using rule-based technologies. Essentially, the restrictions are designed so as to avoid the need to infer the existence of individuals not explicitly present in the knowledge base, and to avoid the need for nondeterministic reasoning. Here we report a subset of the DL specification of OWL 2 RL. Specifically, concepts are formed according to the following syntax (the FOL-reading of concept $\forall R.C$ is: set of x such that $\forall y.R(x, y) \Rightarrow C(y)$):

$$\begin{array}{lcl} B & \longrightarrow & A \mid B_1 \sqcap B_2 \mid B_1 \sqcup B_2 \mid \exists R.B \\ C & \longrightarrow & A \mid C_1 \sqcap C_2 \mid \neg B \mid \forall R.C \mid \\ R & \longrightarrow & P \mid P^- \end{array}$$

Inclusion axioms have the form

[6] Of course, for any interpretation $\mathcal{I}$, $(\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$.

$$\begin{array}{ll} B & \sqsubseteq C \\ R_1 & \sqsubseteq R_2 \\ R_1 & = R_2 \\ \mathsf{dom}(R) & \sqsubseteq C \\ \mathsf{ran}(R) & \sqsubseteq C \\ R \circ R \sqsubseteq R. & \end{array}$$

3.5 A RIF Ontology Layer

A RIF ontology layer $\mathscr{O}$ consists of a finite set of RIF rules. For illustrative purposes, we consider here RIF-Core rules only and recall that RIF-Core corresponds to Horn logic without function symbols (i.e., Datalog [66]) with a number of extensions to support features such as objects and frames as in F-logic [33]. To what concerns us, a rule is of the form

$$p(\mathbf{x}) \leftarrow \exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y}),$$

where $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction[7] of n-ary predicates $p_i(\mathbf{z}_i)$ and $\mathbf{z}_i$ is a vector of distinguished or non-distinguished variables. Specifically, we say that $p(\mathbf{x})$ is the *head* and $\exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y})$ is the *body* of the rule, $\mathbf{x}$ is a vector of variables occurring in the body, called the *distinguished variables*, $\mathbf{y}$ are so-called *non-distinguished variables* and are distinct from the variables in $\mathbf{x}$, each variable occurring in p_i is either a distinguished or a non-distinguished variable. If clear from the context, we may omit the existential quantification $\exists \mathbf{y}$. The intended meaning of a rule such as (3.5) is that the head $p(\mathbf{x})$ is true whenever the body $\exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y})$ is true.

4 Top-k Queries

Having defined how extensional data (a database) and intentional data (an ontology) may be represented, it remains to define how we may query the data. To this end, we define the notion of conjunctive query, which is at the heart of the standard Semantic Web query language SPARQL [52, 53]. Strictly speaking, SPARQL is a query language for data that is stored natively as RDFS or viewed as RDF via middleware. From a logical point of view, its logical counterpart are the well-known notions of *conjunctive/disjunctive* queries. As such, we may see SPARQL essentially as a query language for databases and, indeed, has much in common with SQL.

While SPARQL has originally been proposed to query RDFS graphs only, in the meanwhile, by relying on the representation of OWL and RIF in RDFS, SPARQL is being used to query OWL 2 and RIF ontologies as well, via the definition of the so-called *entailment regimes*. In fact, what correct answers to a SPARQL query

[7] We use the symbol ‘,’ to denote conjunction in the rule body.

are depends on the used entailment regime [51] and the vocabulary from which the resulting answers can be taken.

To what concerns our presentation here, we will consider the essential logical counterpart of SPARQL: namely, conjunctive queries. Specifically, a *simple query* is of the rule-like form

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y})$$

where $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms whose notion is SWL dependent. Specifically, For RDFS, an atom is a triple in which variables and constants may occur; for OWL 2, an atom is n-ary FOL atom ($n = 1, 2$) in which variables and constants may occur; while for RIF, an atom is an n-ary FOL atom in which variables and constants may occur.

We additionally allow built-in atoms involving build-in predicates (properties) having a *fixed interpretation*. For instance, an RDFS query is

$$q(x_1, x_2) \leftarrow (x, worksFor, google), (x, hasSalary, s), (s, <, 23000)$$

and is asking for Google employees earning less than 23000. Here $<$ is a build-in predicate.

For convenience, we write "functional built-in predicates"[8] as *assignments* of the form $x := f(\mathbf{z})$.

As next, we extend the query language by allowing so-called aggregates to occur in a query. Essentially, aggregates may be like the usual SQL aggregate functions such as SUM, AVG, MAX, MIN.

For instance, suppose we are looking for employees that work for some company. We would like to know the average salary of their employment. Such a query may be expressed as

$$\begin{aligned} q(x, avgS) \leftarrow\ & (x, worksFor, y), (x, hasSalary, s), \\ & \mathsf{GroupedBy}(x), \\ & avgS := \mathsf{AVG}[s]\ . \end{aligned}$$

Essentially, we group by the employee, consider for each employee the salaries, and compute the average salary value for each group. That is, if $g = \{\langle t, t_1\rangle, \ldots, \langle t, t_n\rangle\}$ is a group of tuples with the same value t for employee x, and value t_i for s, then the value of $avgL$ for the group g is $(\sum_i t_i)/n$.

Formally, let @ be an aggregate function with

$$@ \in \{\mathsf{SUM}, \mathsf{AVG}, \mathsf{MAX}, \mathsf{MIN}, \mathsf{COUNT}\}$$

then a query with aggregates is of the form

[8] A predicate $p(\mathbf{x}, y)$ is functional if for any $\mathbf{t}$ there is *unique* t' for which $p(\mathbf{t}, t')$ is true.

$$\begin{aligned} q(\mathbf{x}, \alpha) \leftarrow\ & \exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y}), \\ & \mathsf{GroupedBy}(\mathbf{w}), \\ & \alpha\colon = @[f(\mathbf{z})] \end{aligned} \tag{1}$$

where **w** are variables in **x** or **y**, each variable in **x** occurs in **w** and any variable in **z** occurs in **y**.

Eventually, we further allow to order answers according to some ordering functions. For instance, assume that additionally we would like to order the employee according to the average salary of employment. Then such a query will be expressed as

$$\begin{aligned} q(x, avgS) \leftarrow\ & (x, worksFor, y), (x, hasSalary, s), \\ & \mathsf{GroupedBy}(x), \\ & avgS\colon = \mathsf{AVG}[s], \\ & \mathsf{OrderBy}(avgS)\ . \end{aligned}$$

Formally, a query with ordering is of the form

$$q(\mathbf{x}, z) \leftarrow \exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y}), \mathsf{OrderBy}(z)$$

or, in case grouping is allowed as well, it is of the form

$$\begin{aligned} q(\mathbf{x}, z, \alpha) \leftarrow\ & \exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y}), \\ & \mathsf{GroupedBy}(\mathbf{w}), \\ & \alpha\colon = @[f(\mathbf{z})], \\ & \mathsf{OrderBy}(z)\ . \end{aligned} \tag{2}$$

Eventually, we define a *top-k query* as a query limiting the result to the top-k scoring answers, i.e.

$$\begin{aligned} q(\mathbf{x}, z, \alpha) \leftarrow\ & \exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y}), \\ & \mathsf{GroupedBy}(\mathbf{w}), \\ & \alpha\colon = @[f(\mathbf{z})], \\ & \mathsf{OrderBy}(z), \\ & \mathsf{Limit}(k) \end{aligned} \tag{3}$$

We refer the reader to e.g. [2] for an exact formal definition in case of RDFS, to [63] for the case of OWL 2 and to [64] for the case of RIF.

5 Top-k Query Answering Methods

Having now illustrated what a top-k query is, it remains to illustrate the basic methods in computing the top-k answers efficiently. The methods depend on the chosen SWL. In the following, let $\mathscr{K} = \langle \mathscr{F}, \mathscr{O}, \mathscr{M} \rangle$ be a KB.

5.1 The RDFS Case

So far, we have here essentially two options. The first option is as follows.

1. Convert all facts in $\mathscr{F}$ into a set $T_{\mathscr{F}}$ of triples, by using the mapping layer $\mathscr{M}$. Consequently, $\mathscr{K} = \langle \mathscr{F}, \mathscr{O}, \mathscr{M} \rangle$ can be mapped into a pure RDFS triple set
$$T_{\mathscr{K}} = \mathscr{O} \cup T_{\mathscr{F}},$$
called *RDFS graph*, which has the same order of size.
2. Given $T_{\mathscr{K}}$, we then compute its closure $cl(T_{\mathscr{K}})$, whose size is bounded by $\mathscr{O}(|T_{\mathscr{K}}|^2)$ and store it into a relational database. Note that there are several ways to store the closure in a database (see [1, 28]). Essentially, either we may store all the triples in table with four columns *subject, predicate, object, score*, or we use a table for each predicate, where each table has columns *subject, object, score*. The latter approach seems to be better for query answering purposes. Top-k query answering for RDFS reduces then to top-k query answering over relational databases for which efficient solutions exists already.

Another option consists in using top-k retrieval technologies for rule-based KBs and is defined as follows.

1. Compute the closure $cl(\mathscr{O})$ of $\mathscr{O}$.
2. Map all the triples in $cl(\mathscr{O})$ into Datalog rules using e.g. the mapping rules below (see also e.g. [27–29])
$$\begin{aligned}(p, \mathsf{sp}, q) &\mapsto q(x) \leftarrow p(x)\\ (c, \mathsf{sc}, d) &\mapsto d(x) \leftarrow c(x)\\ (p, \mathsf{dom}, c) &\mapsto c(x) \leftarrow p(x, y)\\ (p, \mathsf{range}, c) &\mapsto c(y) \leftarrow p(x, y).\end{aligned}$$
obtaining the rule layer $\mathscr{O}_{\mathscr{K}}$. Let $\mathscr{K}' = \langle \mathscr{F}, \mathscr{O}_{\mathscr{K}}, \mathscr{M} \rangle$ be the resulting rule-based KB.
3. Apply a top-k procedure for rule-based KBs to $\mathscr{K}'$ (see, e.g. [64] and Sect. 5.3).

5.2 The OWL 2 Profile Cases

5.2.1 OWL EL

An option consists in relying on the query reformulation method proposed in [38] to answer conjunctive queries by making use of standard relational database management systems (RDBMSs), and has some commonalities to the OWL QL case (next section). The central idea is to incorporate the consequences of $\mathscr{O}$ into the facts component $\mathscr{F}$.

To capture this formally, the notion of combined first-order (FO) rewritability has been introduced. A DL enjoys combined FO rewritability if it is possible to effectively rewrite

1. $\mathscr{F}$ and $\mathscr{O}$ into an FO structure (independently of the query q); and
2. q and (possibly) $\mathscr{O}$ into a FO query q' (independently of $\mathscr{F}$) such that query answers are preserved, i.e., the answers to q' over the FO structure is the same as the answers to q over $\mathscr{F}$ and $\mathscr{O}$.

The connection to RDBMSs then relies on the well-known equivalence between FO structures and relational databases, and FO queries and SQL queries. The notion of combined FO rewritability generalises the notion of FO reducibility, where $\mathscr{O}$ is incorporated into the query q rather than into $\mathscr{F}$, as it happens for the OWL QL case, while the facts component $\mathscr{F}$ itself is used as a relational instance without any modification [13].

Hence, a top-k query answering procedure for OWL EL may consists of

1. rewriting, once for all, $\mathscr{F}$ and $\mathscr{O}$, using $\mathscr{M}$, into an relational database $DB_{\mathscr{K}}$;
2. rewriting a top-k query q using (possibly) $\mathscr{O}$ and $\mathscr{M}$ into a top-k SQL query to be submitted to the RDBMS containing $DB_{\mathscr{K}}$.

5.2.2 OWL QL

The OWL QL case proceeds similarly as for the OWL EL case, as DL-Lite [3, 13] enjoys the FO reducibility property. Specifically, a top-k query answering procedure for OWL QL may consists of (see [63]) rewriting a top-k query q using (possibly) $\mathscr{O}$ and $\mathscr{M}$ into a top-k SQL query to be submitted to the RDBMS containing $DB_{\mathscr{K}}$.

5.2.3 OWL RL

Concerning OWL RL, we employ the close connection between Horn-DL and Datalog. Specifically, an option to compute the top-k answers consist in using top-k retrieval technologies for rule-based KBs. To this end, we now define a recursive mapping function σ which takes a set of inclusion axioms and maps them into the following expressions:[9]

[9] For the sake of ease of presentation, we are not going to present the whole mapping for Horn-DL, but for a significant subset only that is sufficient to illustrate the main idea behind this translation.

$$\begin{aligned}
\sigma(R_1 \sqsubseteq R_2) &\mapsto \sigma_{role}(R_2, x, y) \leftarrow \sigma_{role}(R_1, x, y)\\
\sigma_{role}(R, x, y) &\mapsto R(x, y)\\
\sigma_r(R^-, x, y) &\mapsto R(y, x)
\end{aligned}$$

$$\begin{aligned}
\sigma(B \sqsubseteq C) &\mapsto \sigma_h(C, x) \leftarrow \sigma_b(B)\\
\sigma_h(A, x) &\mapsto A(x)\\
\sigma_h(C_1 \sqcap C_2, x) &\mapsto \sigma_h(C_1, x) \wedge \sigma_h(C_2, x)\\
\sigma_h(\forall R.C, x) &\mapsto \sigma_h(C, x) \leftarrow \sigma_{role}(R, x, y)\\
\sigma_b(A, x) &\mapsto A(x)\\
\sigma_b(C_1 \sqcap C_2, x) &\mapsto \sigma_b(C_1, x) \wedge \sigma_b(C_2, x)\\
\sigma_b(C_1 \sqcup C_2, x) &\mapsto \sigma_b(C_1, x) \vee \sigma_b(C_2, x)\\
\sigma_b(\exists R.C, x) &\mapsto \sigma_{role}(R, x, y) \wedge \sigma_b(C, y)
\end{aligned}$$

where y is a new variable.

We then transform the above generated expressions into rules by applying recursively the following mapping:

$$\begin{aligned}
\sigma_r((H \wedge H') \leftarrow B) &\mapsto \sigma_r(H \leftarrow B), \sigma_r(H' \leftarrow B)\\
\sigma_r((H \leftarrow H') \leftarrow B) &\mapsto \sigma_r(H \leftarrow (B \wedge H'))\\
\sigma_r(H \leftarrow (B_1 \vee B_2)) &\mapsto \sigma_r(H \leftarrow B_1), \sigma_r(H \leftarrow B_2)
\end{aligned}$$

Eventually, if none of the above three rules can be applied then

$$\sigma_r(H \leftarrow B) \mapsto H \leftarrow B\ .$$

Therefore, a top-k retrieval method of OWL RL is as follows:

1. Map $\mathscr{O}$ into Datalog rules using e.g. the mapping rules above obtaining the rule layer $\mathscr{O}_{\mathscr{K}}$. Let $\mathscr{K}' = \langle \mathscr{F}, \mathscr{O}_{\mathscr{K}}, \mathscr{M} \rangle$ be the resulting rule-based KB.
2. Apply a top-k procedure for rule-based KBs to $\mathscr{K}'$ (see, e.g. [64] and Sect. 5.3).

5.3 The RIF case

Concerning the RIF case, by exploiting the relationship to Datalog (specifically of RIF-Core), we may opt for a solution inherited from the logic programming context. We have already reported in Sect. 1 about various proposal developed so far. We recap here the main principle behind the so far most general approach [64].

The basic reasoning idea stems from the database literature (see, e.g. [35]) and consists in retrieving iteratively query answers and simultaneously computing a threshold

δ. The threshold δ has the fundamental property that any newly retrieved tuple will have a score less or equal than δ. As a consequence, as soon as we have retrieved k tuples greater or equal to δ, we may stop. Note that a distinguishing feature of this query answering procedure is that it does not determine all answers, but collects, during the computation, answers incrementally together and stops as soon as it has gathered k answers greater or equal than a computed threshold δ. The finiteness of the truth space guarantees the termination of this process, which otherwise may not terminate.

6 Conclusions

In this work, we briefly discussed about a relatively novel issue for SWLs with a huge data repository, namely the problem of evaluating ranked top-k queries. We have illustrated how this problem may be currently approached within the context of RDFS (the triple language), OWL 2 Profiles (frame-based languages) and RIF (rule language).

While for relational databases a non negligible amount of solutions have been proposed so far, in the context of knowledge representation and reasoning, the development is still in its infancy, both from an algorithmic and implementation point of view. So far, for the languages RDFS, OWL QL, OWL EL and RIF ad hoc solution have been worked out, while for OWL RL a reduction to RIF is required. Note that for RDFS and OWL QL, a reduction to RIF exists as well and, thus, top-k techniques for this latter can be applied.

We believe that with the growth in size of data repositories accessible via SWLs, the top-k retrieval problem will emerge as a significant problem, as much as the top-k retrieval problem is for current Information Retrieval Systems [8].

References

1. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.: Sw-store: a vertically partitioned dbms for semantic web data management. VLDB J. **18**(2), 385–406 (2009)
2. Zimmermann, A.P.A., Lopes, N., Straccia, U.: A general framework for representing, reasoning and querying with annotated semantic web data. J. Web Semant. **11**, 72–95 (March 2012)
3. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. **36**, 1–69 (2009)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathscr{E}\mathscr{L}$ envelope. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence IJCAI-05, pp. 364–369, Edinburgh, UK. Morgan-Kaufmann Publishers, San Francisco (2005)
5. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proceedings of the 18th International Joint Conference on Artificial intelligence, pp. 325–330. Morgan Kaufmann, San Francisco (2003)

6. Baader, F., Brandt S., Lutz, C.: Pushing the $\mathscr{EL}$ envelope further. In: Clark, K., Patel-Schneider, P.F. (eds.) Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions (2008)
7. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.). The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
8. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman, Boston (1999)
9. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF schema. In: W3C Recommendation, W3C (2004). http://www.w3.org/TR/rdf-schema/
10. Bruno, N., Chaudhuri, S., Gravano, L.: Top-k selection queries over relational databases: mapping strategies and performance evaluation. ACM Trans. Database Syst. **27**(2), 153–187 (2002)
11. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: tractable description logics for ontologies. In: Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005) (2005)
12. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06), pp. 260–270 (2006)
13. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the dl-lite family. J. Autom. Reasoning **39**(3), 385–429 (2007)
14. Chang, K.C.-C., won Hwang, S.: Minimal probing: supporting expensive predicates for top-k queries. In: Proceedings of the SIGMOD Conference, pp. 346–357 (2002)
15. Chortaras, A., Stamou, G.B., Stafylopatis, A.: Integrated query answering with weighted fuzzy rules. In: Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-07), vol. 4724 in Lecture Notes in Computer Science, pp. 767–778. Springer (2007)
16. Chortaras, A., Stamou, G.B., Stafylopatis, A.: Top-down computation of the semantics of weighted fuzzy logic programs. In: Proceedings of the 1st International Conference on Web Reasoning and Rule Systems (RR-07), pp. 364–366 (2007)
17. Cuenca-Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: the next step for OWL. J. Web Semant. **6**(4), 309–322 (2008)
18. Damásio, C.V., Medina, M., Ojeda-Aciego, J.: A tabulation procedure for first-order residuated logic programs. In: Proceedings of the IEEE World Congress on Computational Intelligence (Sect. Fuzzy Systems) (WCCI-06), pp. 9576–9583 (2006)
19. Damásio, C.V., Medina, M., Ojeda-Aciego, J.: A tabulation procedure for first-order residuated logic programs. In: Proceedings of the 11th International Conference on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-06) (2006)
20. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Comput. Surv. **33**(3), 374–425 (2001)
21. Dubois, D., Prade, H.: Possibility theory, probability theory and multiple-valued logics: a clarification. Ann. Math. Artif. Intell. **32**(1–4), 35–66 (2001)
22. Fagin, R.: Combining fuzzy information: an overview. SIGMOD Rec. **31**(2), 109–118 (2002)
23. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: Symposium on Principles of Database Systems (2001)
24. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proceedings of the 12th International Conference on World Wide Web, pp. 48–57. ACM Press (2003)
25. Gutierrez, C., Hurtado, C., Mendelzon, A.O.: Foundations of semantic web databases. In: Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS-04). ACM Press (2004)
26. Haase, C., Lutz, C.: Complexity of subsumption in the $\mathscr{EL}$ family of description logics: acyclic and cyclic tboxes. In: Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N. (eds.) Proceedings of the 18th European Conference on Artificial Intelligence (ECAI08), vol. 178 of Frontiers in Artificial Intelligence and Applications, pp. 25–29. IOS Press (2008)

27. Ianni, G., Krennwallner, T., Martello, A., Polleres, A.: Dynamic querying of mass-storage RDF data with rule-based entailment regimes. In: Proceedings of the 8th International Semantic Web Conference (ISWC-09), vol. 5823 in Lecture Notes in Computer Science, pp. 310–327. Springer (2009)
28. Ianni, G., Krennwallner, T., Martello, A., Polleres, A.: A rule system for querying persistent rdfs data. In: Proceedings of the 6th European Semantic Web Conference on Semantic Web: Research and Applications (ESWC-2009), pp. 857–862 (2009)
29. Ianni, G., Krennwallner, T., Martello, A., Polleres, A.: A rule system for querying persistent rdfs data. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E.P.B. (eds.) ESWC, vol. 5554 of Lecture Notes in Computer Science, pp. 857–862. Springer (2009)
30. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K.: Supporting top-k join queries in relational databases. In: Proceedings of the 29th International Conference on Very Large Data, Bases (VLDB-03), pp. 754–765 (2003)
31. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K., Elmongui, H.G., Shah, R., Vitter, J.S.: Adaptive rank-aware query optimization in relational databases. ACM Trans. Database Syst. **31**(4), 1257–1304 (2006)
32. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. **40**(4), 1–58 (2008)
33. Kifer, M., Lausen, G.: Logical foundations of object-oriented and frame-based languages. J. ACM **42**(4), 741–843 (1995)
34. Li, C., Chang, K.C.-C., Ilyas, I.F.: Supporting ad-hoc ranking aggregates. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD-06), pp. 61–72. ACM Press, New York (2006)
35. Li, C., Chang, K.C.-C., Ilyas, I.F., Song, S.: RankSQL: query algebra and optimization for relational top-k queries. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD-05), pp. 131–142. ACM Press, New York (2005)
36. Lloyd, J.W.: Foundations of Logic Programming. Springer, Heidelberg (1987)
37. Lukasiewicz, T., Straccia, U.: Top-k retrieval in description logic programs under vagueness for the semantic web. In: Proceedings of the 1st International Conference on Scalable Uncertainty Management (SUM-07), vol. 4772 in Lecture Notes in Computer Science, pp. 16–30. Springer (2007)
38. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathscr{EL}$ using a relational database system. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI09). AAAI Press (2009)
39. Marian, A., Bruno, N., Gravano, L.: Evaluating top-k queries over web-accessible databases. ACM Trans. Database Syst. **29**(2), 319–362 (2004)
40. Marin, D.: A formalization of rdf. Technical report TR/DCC-2006-8, Deptartment of Computer Science, Universidad de Chile (2004). http://www.dcc.uchile.cl/cgutierr/ftp/draltan.pdf
41. Muñoz, S., Pérez, J., Gutiérrez, C.: Minimal deductive systems for rdf. In: Proceedings of the 4th European Semantic Web Conference (ESWC-07), vol. 4519 in Lecture Notes in Computer Science, pp. 53–67. Springer (2007)
42. OWL web ontology language overview. In: W3C (2004). http://www.w3.org/TR/owl-features/
43. OWL 2 web ontology language document overview. In: W3C (2009). http://www.w3.org/TR/2009/REC-owl2-overview-20091027/
44. OWL 2 web ontology language profiles. In: W3C (2009). http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/
45. Moreno, G., Julian, P., Medina, J., Ojeda, M.: Efficient thresholded tabulation for fuzzy query answering. In: Foundations of Reasoning Under Uncertainty, vol. 249 in Studies in Fuzziness and Soft Computing, pp. 125–141. Springer (2010)
46. Papadimitriou, C.H.: Computational Complexity. Addison Wesley, Reading (1994)
47. Ragone, A., Straccia, U., Di Noia, T., Di Sciascio, E., Donini, F.M.: Vague knowledge bases for matchmaking in p2p e-marketplaces. In: Proceedings of the 4th European Semantic Web Conference (ESWC-07), vol. 4519 in Lecture Notes in Computer Science, pp. 414–428. Springer (2007)

48. Ragone, A., Straccia, U., Di Noia, T., Di Sciascio, E., Donini, F.M.: Fuzzy matchmaking in e-marketplaces of peer entities using Datalog. Fuzzy Sets Syst. **160**(2), 251–268 (2009)
49. RDF semantics. In: W3C (2004). http://www.w3.org/TR/rdf-mt/
50. Rule interchange format (RIF). In: W3C (2011). http://www.w3.org/2001/sw/wiki/RIF
51. SPARQL 1.1 entailment regimes. In: W3C (2011). http://www.w3.org/TR/2011/WD-sparql11-entailment-20110512/
52. SPARQL 1.1 query language. In: W3C (2012). http://www.w3.org/TR/sparql11-query/
53. SPARQL query language for RDF. In: W3C (2008). http://www.w3.org/TR/rdf-sparql-query/
54. Straccia, U.: Answering vague queries in fuzzy DL-Lite. In: Proceedings of the 11th International Conference on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-06), pp. 2238–2245. E.D.K., Paris (2006)
55. Straccia, U.:. Towards top-k query answering in deductive databases. In: Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics (SMC-06), pp. 4873–4879. IEEE (2006)
56. Straccia, U.: Towards top-k query answering in description logics: the case of DL-Lite. In: Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA-06), vol. 4160 in Lecture Notes in Computer Science, pp. 439–451, Liverpool, UK. Springer (2006)
57. Straccia, U.: Towards vague query answering in logic programming for logic-based information retrieval. In: Proceedings of World Congress of the International Fuzzy Systems Association (IFSA-07), vol. 4529 in Lecture Notes in Computer Science, pp. 125–134, Cancun, Mexico. Springer (2007)
58. Straccia, U.: Fuzzy description logic programs. In: Marsala, C., Bouchon-Meunier, B., Yager, R.R., Rifqi, M. (eds.) Uncertainty and Intelligent Information Systems, Chap. 29, pp. 405–418. World Scientific, Singapore (2008)
59. Straccia, U.: Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In: Proceedings of 4th International Summer School, Tutorial Lectures on Reasoning Web, vol. 5224 in Lecture Notes in Computer Science, pp. 54–103. Springer (2008)
60. Straccia, U.: An ontology mediated multimedia information retrieval system. In: Proceedings of the 40th International Symposium on Multiple-Valued Logic (ISMVL-10), pp. 319–324. IEEE Computer Society (2010)
61. Straccia, U.: Softfacts: a top-k retrieval engine for ontology mediated access to relational databases. In: Proceedings of the 2010 IEEE International Conference on Systems, Man and Cybernetics (SMC-10), pp. 4115–4122. IEEE Press (2010)
62. Straccia, U.: Fuzzy logic, annotation domains and semantic web languages. In: Proceedings of the 5th International Conference on Scalable Uncertainty Management (SUM-11), vol. 6929 in Lecture Notes in Computer Science, pp. 2–21. Springer (2011)
63. Straccia, U.: Top-k retrieval for ontology mediated access to relational databases. Inf. Sci. **198**, 1–23 (2012)
64. Straccia, U., Madrid, N.: A top-k query answering procedure for fuzzy logic programming. Fuzzy Sets Syst. **205**, 1–29 (2012)
65. ter Horst, H.J.: Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. J. Web Semant. **3**(2–3), 79–115 (2005)
66. Ullman, J.D.: Principles of Database and Knowledge Base Systems, vols. 1, 2. Computer Science Press, Potomac (1989)
67. Vardi, M.: The complexity of relational query languages. In: Proceedings of the 14th ACM SIGACT Symposium on Theory of Computing (STOC-82), pp. 137–146 (1982)
68. Vojtás, P.: Fuzzy logic aggregation for semantic web search for the best (top-*k*) answer. In: Sanchez, E. (ed.) Fuzzy Logic and the Semantic Web, Capturing Intelligence, Chap. 17, pp. 341–359. Elsevier, Amsterdam (2006)
69. XML. In: W3C. http://www.w3.org/XML/