Scalable Querying Services over Fuzzy Ontologies*

Jeff Z. Pan
Dept. of Computing Science
Univ. of Aberdeen
Aberdeen, UK

Giorgos Stamou,
Giorgo Stoilos
Dept. of Computer Science
NTUA
Athens, Greece

Stuart Taylor,
Edward Thomas
Dept. of Computing Science
Univ. of Aberdeen
Aberdeen, UK

ABSTRACT

Fuzzy ontologies are envisioned to be useful in the Semantic Web. Existing fuzzy ontology reasoners are not scalable enough to handle the scale of data that the Web provides. In this paper, we propose a framework of fuzzy query languages for fuzzy ontologies, and present query answering algorithms for these query languages over fuzzy DL-Lite ontologies. Moreover, this paper reports on implementation of our approach in the fuzzy DL-Lite query engine in the ONTOSEARCH2 system and preliminary, but encouraging, benchmarking results. To the best of our knowledge, this is the first ever scalable query engine for fuzzy ontologies.

Categories and Subject Descriptors

I.2.3 [**Deduction and Theorem Proving**]: Uncertainty, "fuzzy", and probabilistic reasoning; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation Languages

General Terms

Language, Algorithm, Experimentation

Keywords

Semantic Web, Lightweight Ontology Language, Fuzzy Ontology, Scalable Query Answering, Fuzzy SPARQL

1. INTRODUCTION

Fuzzy ontologies are envisioned to be useful in the Web. On the one hand, ontologies serve as basic semantic infrastructure, providing shared understanding of certain domain across different applications, so as to facilitate machine understanding of Web resources. On the other hand, being able to handle fuzzy and imprecise information is crucial to the Web. Web data are likely to be uncertain or conflicting and could raise trust issues. It has been argued that uncertainty representation and reasoning could help to harmonise and integrate Web data from different sources. To this end, W3C has recently set up an incubator group on Uncertainty Reasoning for the Web[1].

Although recently there have been quite a lot of work on Description Logics (DLs) based fuzzy ontology languages, e.g., [27, 26, 17, 12, 2, 25], there exist no fuzzy ontology reasoners that could be efficient and/or scalable enough to handle the scale of data that the Web provides. Interestingly, there currently exist two fuzzy ontology reasoners, namely the tableaux based fuzzy reasoner FiRE[2] [25, 24], which supports a nominal and datatype-free subset of fuzzy-OWL DL, i.e. fuzzy-$\mathcal{SHIN}$, and the mixed integer programming fuzzy reasoner *fuzzyDL*[3], which supports fuzzy-OWL Lite, namely fuzzy-$\mathcal{SHIF}(\mathcal{D})$ [28]. Like their crisp counterparts, fuzzy-$\mathcal{SHIN}$ and fuzzy-$\mathcal{SHIF}(\mathcal{D})$ come with (at least) EXPTIME computational complexity, thus the scalability of the above systems is doubtful. Following current research developments in crisp DLs, there is an effort on lightweight fuzzy ontology languages. In particular, Straccia [29] extended the DL-Lite ontology language [5] to fuzzy DL-Lite. DL-Lite, which is expressive enough to represent most features of UML class diagrams, enables highly efficient query answering procedures by making use of database technologies. There are major limitations on Straccia's query language for fuzzy DL-Lite: the proposed query language has the same syntax as the query language of the crisp DL-Lite, and thus does *not* allow one to specify either thresholds for query atoms (such as "tell me e-shops that are popular [with degrees at least 0.8] and sell good books [with degrees at least 0.9]"), or weights and preferences on query atoms (such as "get me all cars that are fast and fancy but consider speed more important [with weight 0.7] than design [with weight 0.3]"), so as to exploit fuzzy assertions in fuzzy ontologies. To the best of our knowledge, there exist no report on scalable query engines for fuzzy DL-Lite, let alone supporting more expressive fuzzy query languages.

This paper makes the following major contributions:

1. It proposes a general framework, which consists of threshold queries and general fuzzy queries (Section 3.1), for querying over fuzzy ontologies, covering all the existing query languages for fuzzy ontologies as well as some new ones that can be customised by users. Comparing with Straccia's query language, the threshold query language is flexible as it allows one to specify a threshold for each query atom (as shown in the above ex-

*This paper is based on a poster titled "Expressive Querying over Fuzzy DL-Lite Ontologies" presented in the 2007 International Workshop on Description Logics (DL2007).

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

[1] http://www.w3.org/2005/Incubator/urw3
[2] http://www.image.ece.ntua.gr/~nsimou
[3] http://gaia.isti.cnr.it/~straccia

ample). In fact, entailment of threshold queries generalises the entailment problem of fuzzy assertions. On the other hand, the general fuzzy query language is a general form of the fuzzy threshold query language, which in turn is a general form of Straccia's query language. General fuzzy queries are motivated by the field of fuzzy information retrieval [8] where *weighted Boolean queries* [33] have been proposed for retrieving fuzzy information from fuzzy relational databases. Our general fuzzy query language generalises most former approaches of weighted Boolean queries [20, 3, 34, 4] and several new approaches, like the p-norm approach [21], the geometric mean approach [6], the so called fuzzy weighted t-norm queries from Chortaras et. al. [7], which in turn generalise weighted min queries [23], and aggregation queries from Vojtas [32]. Thus, the main strength of the general fuzzy query language is the openness of the use of semantics of conjunction and that of the degree-associated atoms. Consequently, the framework can accommodate different intuitive meaning on the associated degrees, like preferences, degrees of importance, fuzzy thresholds and more.

2. It not only provides an abstract syntax (Section 3.1) for the proposed framework, but also shows how to extend the SPARQL (a well known Semantic Web query language) syntax for the proposed query languages in the framework (Section 3.2). Our extension uses specially formatted SPARQL comments, thus the fuzzy queries are still valid SPARQL queries, and it does not affect current SPARQL tools and implementations.

3. It not only proposes the syntax of query languages, but also provides semantics (Section 3.1) and algorithms for answering such queries over arbitrary fuzzy DL-Lite ontologies together with sound and complete proofs (w.r.t. the semantics); and the algorithms cover all the mentioned languages in the framework (Section 3.3).

4. To the best of our knowledge, it presents the *first ever* scalable query engine for fuzzy ontologies, based on the ONTOSEARCH2 system[4] [16], which consists of, among others, a query engine for DL-Lite and one for fuzzy DL-Lite. The performance of the fuzzy DL-Lite query engine is evaluated against a benchmark (a fuzzy variant of the Lehigh University Benchmark (LUBM) [10], called f-LUBM) that we propose, which is the first of its kind and against which future implementations can also be evaluated. The query engine is able to handle millions of individuals, according to the preliminary but encouraging evaluation (Section 4).

5. It presents a use cases about searching ontology based on keyword-plus-entailment searches, so as to show how to apply our efficient querying support for fuzzy ontologies. The use case application is available online (Section 5).

[4]http://www.ontosearch.org/

2. PRELIMINARIES

2.1 DL-based Ontologies and Query Answering

Due to the limitation of space, we do not provide a formal introduction of Description Logics (DLs), but rather point the reader to [1]. It should be noted that, even for the smallest propositionally closed DL $\mathcal{ALC}$ (which only provides the class constructors $\neg C, C \sqcap D, C \sqcup D, \exists R.C$ and $\forall R.C$), the complexity of logical entailment is EXPTIME. Recently, Calvanese et. al. proposed DL-Lite, which has a low reasoning overhead (worst case polynomial time) [5]. A DL-Lite ontology ($\mathcal{O}$) is a set of axioms of the following forms:

1. class inclusion axioms: $B \sqsubseteq C$ where B is a basic class $B := \mathsf{A} \mid \exists R \mid \exists R^-$ and C is a general class $C := B \mid \neg B \mid C1 \sqcap C2$ (where A denotes an named class and R denotes a named property);

2. functional property axioms: $\mathsf{Func}(R), \mathsf{Func}(R^-)$, where R is a named property;

3. individual axioms: $B(\mathtt{a}), R(\mathtt{a}, \mathtt{b})$ where $\mathtt{a}$ and $\mathtt{b}$ are named individuals.

Description Logics have a well-defined model-theoretic semantics, which are provided in terms of interpretations. An *interpretation* $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set of objects and $\cdot^{\mathcal{I}}$ is an *interpretation function*, which maps each class C to a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each property R to a subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Typical reasoning ontology services include ontology consistency checking (i.e., whether there exists an interpretation of an ontology), subsumption checking (i.e., whether the interpretation of a class C_1 is a subset of the interpretation of a class C_2 in all interpretations of the ontology), instance checking (i.e. whether an assertion is logically implied by an ontology) and query answering. In this paper, we will focus on query answering. A conjunctive query (CQ) q is of the form

$$q(X) \leftarrow \exists Y.conj(X, Y) \tag{1}$$

or simply $q(X) \leftarrow conj(X, Y)$, where $q(X)$ is called the head, $conj(X, Y)$ is called the body, X are called the distinguished variables, Y are existentially quantified variables called the non-distinguished variables, and $conj(X, Y)$ is a conjunction of atoms of the form $\mathsf{A}(v)$, $R(v_1, v_2)$, where A, R are respectively *named* classes and *named* properties, v, v_1 and v_2 are *individual* variables in X and Y or individual names in $\mathcal{O}$. As usual, an interpretation $\mathcal{I}$ satisfies an ontology $\mathcal{O}$ if it satisfies all the axioms in $\mathcal{O}$; in this case, we say $\mathcal{I}$ is a model of $\mathcal{O}$. Given an evaluation $[X \mapsto S]$ (where S is a set of individuals), if every model $\mathcal{I}$ of $\mathcal{O}$ satisfies $q_{[X \mapsto S]}$, we say $\mathcal{O}$ entails $q_{[X \mapsto S]}$; in this case, S is called a *solution* of q. A disjunctive query (DQ) is a set of conjunctive queries sharing the same head. Theoretically, allowing only named classes and properties as atoms is not a restriction, as we can always define such named classes and properties in ontologies. Practically, this should not be an issue as querying against *named* relations is a usual practice when people query over relational databases.

After some careful query rewriting by DL-Lite reasoners [5], query answering over DL-Lite ontologies can be carried out by an SQL engine, so as to take advantage of exist-

ing query optimisation strategies and algorithms provided by modern database management systems.

2.2 f-DL-LITE

Straccia [29] proposed fuzzy DL-Lite (or *f-DL-Lite* for short), which extends DL-Lite core with fuzzy assertions of the forms $B(\mathtt{a}) \geq n, R(\mathtt{a},\mathtt{b}) \geq n$, where B is basic class, R is a property, $\mathtt{a}$ and $\mathtt{b}$ are individuals and n is a real number in the range $[0,1]$. The semantics of f-DL-Lite ontologies is defined in terms of *fuzzy interpretations* [27]. A fuzzy interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where the domain $\Delta^{\mathcal{I}}$ is a non-empty set of objects and $\cdot^{\mathcal{I}}$ is a fuzzy interpretation function, which maps:

- an individual $\mathtt{a}$ to an element of $\mathtt{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$,
- a named class A to a membership function $\mathsf{A}^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow [0,1]$, and
- a named property R to a membership function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0,1]$.

Using the fuzzy set theoretic operations [11], fuzzy interpretations can be extended to interpret f-DL-Lite class and property descriptions. Following Straccia [29], we use the Lukasiewicz negation, $c(a) = 1 - a$ and the Gödel t-norm for interpreting conjunctions, $t(a,b) = \min(a,b)$. The semantics of f-DL-Lite class and property descriptions, and f-DL-Lite axioms are depicted in Table 1. Given the above semantics, it is obvious that crisp assertions $B(a)$, $R(a,b)$ are special forms of fuzzy assertions where $n = 1$.

Syntax	Semantics
$\exists R$	$(\exists R)^{\mathcal{I}}(o_1) = \sup_{o_2 \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(o_1,o_2)\}$
$\neg B$	$(\neg B)^{\mathcal{I}}(o) = 1 - B^{\mathcal{I}}(o)$
$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}}(o) = t(C_1^{\mathcal{I}}(o), C_2^{\mathcal{I}}(o))$
R^-	$(R^-)^{\mathcal{I}}(o_2,o_1) = R^{\mathcal{I}}(o_1,o_2)$
$B \sqsubseteq C$	$\forall o \in \Delta^{\mathcal{I}}, B^{\mathcal{I}}(o) \leq C^{\mathcal{I}}(o)$
$\mathsf{Func}(R)$	$\forall o_1 \in \Delta^{\mathcal{I}}, \sharp\{o_2 \mid R^{\mathcal{I}}(o_1,o_2) > 0\} = 1$
$B(\mathtt{a}) \geq n$	$B^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}}) \geq n$
$R(\mathtt{a},\mathtt{b}) \geq n$	$R^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}},\mathtt{b}^{\mathcal{I}}) \geq n$

Table 1: Semantics of f-DL-Lite class and property descriptions, and f-DL-Lite axioms

Similarly to crisp DL lite, fuzzy-DL-Lite, provides means to specify *role-typing* and *participation constraints* but interestingly it assigns fuzzy meaning on them. More precisely, a role-typing assertion of the form $\exists R \sqsubseteq A_1$ (resp. $\exists R^- \sqsubseteq A_2$) states that the first (resp. second) component of R belongs to A_1 (resp. A_2) at-least to the membership degree that the relation R holds, i.e. $R^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}},\mathtt{b}^{\mathcal{I}}) \leq A_1^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}})$ (resp. $(R^-)^{\mathcal{I}}(\mathtt{b}^{\mathcal{I}},\mathtt{a}^{\mathcal{I}}) = R^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}},\mathtt{b}^{\mathcal{I}}) \leq A_2^{\mathcal{I}}(\mathtt{b}^{\mathcal{I}})$).

2.3 Abstract and RDF/XML Syntax

Since DL-Lite (resp. f-DL-Lite) is a sub-language of OWL (resp. f-OWL DL), we provide an abstract and RDF/XML syntax for DL-Lite and f-DL-Lite ontologies in this subsection, following the paradigm of OWL DL [18]. OWL DL ontologies in RDF/XML syntax can be generated from those written in the abstract syntax, using the official mapping between the two kind of syntax provided in [18]. Our proposed abstract syntax for DL-Lite core is based on that of the DL-Lite specified in the OWL1.1 member submission (an extension of OWL DL) in the tractable fragments document [9]. It is slightly different from that in the OWL 1.1 document, mainly due to the fact that it uses the OWL DL syntax (which is slightly different from that of OWL 1.1) and RDF/XML serialisation.

Besides disallowing several expressive OWL DL constructors, DL-Lite restricts the use of several of the allowed constructors and especially w.r.t. which side of class axioms they can appear. Thus, the definition of an abstract syntax is slightly trickier than that of OWL. In OWL DL, for example, disjointness, equivalence and subclass axioms are defined by the following abstract syntax:

```
axiom ::= 'DisjointClasses(' description description
                                        { description } ')' |
          'EquivalentClasses(' description { description } ')' |
          'SubClassOf(' description description ')'
```

where "description" can be any OWL DL class description. In DL-Lite, only basic classes are allowed on the left-hand side of axioms, while general classes are allowed only on the right-hand side. Thus, the above abstract syntax should be adjusted to:

```
axiom ::= 'DisjointClasses(' basicClass basicClass
                                        { basicClass } ')' |
          'EquivalentClasses(' basicClass { basicClass } ')' |
          'SubClassOf(' basicClass generalClass ')'
```

where "basicClass" and "generalClass" represent the basic and general classes of DL-Lite, respectively. The abstract syntax for these two elements should also follow the restrictions of DL-Lite; e.g., a "generalClass" can be an intersection of classes while a basic class is not. By using the RDF/XML serialisation mapping described for OWL DL, one is able to obtain DL-Lite and f-DL-Lite ontologies in RDF/XML syntax. For example, the following class axiom in RDF/XML syntax is a valid DL-Lite axiom,

```
<owl:Class rdf:ID="C">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:complementOf>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#R"/>
              <owl:someValuesFrom rdf:resource="&owl;Thing"/>
            </owl:Restriction>
          </owl:complementOf>
        </owl:Class>
        <owl:Class>
          <owl:complementOf rdf:resource="#A"/>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

which corresponds to the axiom $\mathsf{C} \sqsubseteq \neg\exists R \sqcap \neg\mathsf{A}$.

Finally, fuzziness in the individual axioms of f-DL-Lite is defined by a restriction of the abstract syntax of facts of f-OWL DL presented in [26], since we are only allowing the inequality "$\geq$". The abstract syntax is the following:

```
individual ::= 'Individual(' [individualID] {annotation}
               {'type'( type ')' [ineqType] [degree]}
               {value [ineqType] [degree] } ')'
ineqType   ::= '>='
degree     ::= real-number-between-0-and-1-inclusive
```

Similarly, we can follow the RDF/XML serialization proposed in [26] to have fuzzy individual axioms in a f-DL-Lite file. For example, stating that o is Heavy to degree at-least 0.7 could be specified by the following RDF/XML syntax.

```
<Heavy rdf:about="o" owlx:ineqType="≥" owlx:degree="0.7"/>
```

The full specification of the f-DL-Lite (and consequently DL-Lite) abstract syntax can be found in the Appendix. The ONTOSEARCH2 system allows users to submit their crisp and fuzzy ontologies to its repository. Furthermore, it provides an RDF/XML syntax checker for DL-Lite.

3. QUERYING F-DL-LITE ONTOLOGIES

In this section, we present a general framework for representing expressive fuzzy queries over f-DL-Lite ontologies. More precisely, we will introduce two query languages for f-DL-Lite ontologies. The first language extends conjunctive queries with thresholds for atoms in queries. Entailment of threshold queries generalises the entailment problem of fuzzy assertions. The second language is a general fuzzy query language, motivated by the field of fuzzy information retrieval [8], where weighted Boolean queries have been proposed [33, 20, 3, 34]. As it was showed in [4] most of these approached could be represented under a general framework using general fuzzy operators, like t-norms and fuzzy implications. Our general fuzzy query language extends these results by allowing more fuzzy operators and thus generalising many of the recent approaches like the query language proposed in [29] for fuzzy DL-Lite, weighted t-norm queries [7], which in turn generalise weighted min queries [23], p-norms [21], fuzzy aggregations [32] and the geometric mean [6]. In order to enable such types of queries in the Semantic Web, we also propose the extension of the SPARQL [19] query language, so as to represent the queries in our general framework. In what follows, we first introduce these new query languages, providing their syntax and semantics. We then present the extension of SPARQL, and finally we provide algorithms of query answering for queries in the proposed query languages.

3.1 Two New Query Languages

3.1.1 Threshold Queries

As noted in [5] in DL-Lite the instance checking problem is a special case of conjunctive queries. Since f-DL-Lite extends DL-Lite with fuzzy assertions, it would be natural to define a query language so that the entailment of such queries could generalise entailment checking of fuzzy assertions. Accordingly, we define *conjunctive threshold queries* (CTQ) which extend atoms $\mathsf{A}(v), R(v_1, v_2)$ in conjunctive queries of the form (1) into the following forms $\mathsf{A}(v) \geq t_1, R(v_1, v_2) \geq t_2$, where $t_1, t_2 \in (0, 1]$ are thresholds. It turns out that threshold queries are very important types of queries since in [13] the authors used them in order to devise a reasoning algorithm for the fuzzy language fuzzy-CARIN.

Example 1. *We can query models who are tall with a degree no less than 0.7 and light with a degree no less than 0.8 with the following conjunctive threshold query:*

$$q(v) \leftarrow \mathsf{Model}(v) \geq 1, \mathsf{Tall}(v) \geq 0.7, \mathsf{Light}(v) \geq 0.8.$$

It is obvious that threshold queries are more flexible than queries of the form (1) in that users can specify different thresholds for different atoms in their queries.

Formally, given an f-DL-Lite ontology $\mathcal{O}$, a conjunctive threshold query q_T and an evaluation $[X \mapsto S]$, we say $\mathcal{O}$ entails q_T (denoted as $\mathcal{O} \models q_T$) if every interpretation $\mathcal{I}$ of $\mathcal{O}$ satisfies the following condition: for each atom $\mathsf{A}(v) \geq t_1$ $(R(v_1, v_2) \geq t_2)$ of q_T, we have $\mathsf{A}^{\mathcal{I}}(v)_{X \mapsto S} \geq t_1$ (resp. $R^{\mathcal{I}}(v_1, v_2)_{X \mapsto S} \geq t_2$). In this case, S is called a *solution* of q_T. A disjunctive threshold query (DTQ) is a set of conjunctive threshold queries sharing the same head.

3.1.2 General Fuzzy Queries

Since f-DL-Lite associates assertions with degrees of truth, another useful feature for its query language is to associate degrees of truth with answers in answer sets of queries over f-DL-Lite ontologies. In threshold queries, an evaluation $[X \mapsto S]$ either satisfies the query entailment or not; hence, answers of such queries are crisp. In this subsection, we introduce general fuzzy queries which allow fuzzy answers. Syntactically, like the query language proposed in [33] and threshold queries, general fuzzy conjunctive queries (GFCQ) extend the atoms $\mathsf{A}(v), R(v_1, v_2)$ of conjunctive queries of the form (1) into ones with the following form $\mathsf{A}(v) : k_1, R(v_1, v_2) : k_2$, where $k_1, k_2 \in (0, 1]$ are degrees. .

The strength of the general fuzzy query language is the openness of the use of fuzzy operations. Indeed, as many theoretical and practical studies [32, 4] have pointed out, the choice of fuzzy operations is usually context dependent. Following the style of the semantics of fuzzy-SWRL [17] the existential quantifier is interpreted as sup, while we leave the semantics of the conjunction (G) and that of the degree-associated atoms (a) open. To simplify the presentation of the semantics, we use a unified representation $atom_i(\bar{v})$ for atoms in general fuzzy conjunctive queries. Given an f-DL-Lite ontology $\mathcal{O}$, an interpretation $\mathcal{I}$ of $\mathcal{O}$, a general fuzzy conjunctive query q_F and an evaluation $[X \mapsto S]$, the degree of truth of q_F under $\mathcal{I}$ is

$$d = \sup_{S' \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}} \{G_{i=1}^{n}\ a(k_i, atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']})\}$$

where $k_i \in (0, 1]$ are degrees ($1 \leq i \leq n$), $atom_i$ are atoms in the query, G is the semantic function for conjunctions and a is the semantic function for degree-associated atoms. $S : d$ is called a *candidate solution* of q_F. When $d > 0$, $S : d$ is called a *solution* of q_F. Furthermore, the semantic functions should satisfy the following condition:

$$\text{If } atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']}) = 0 \text{ for all possible } S', d = 0. \quad (2)$$

A general fuzzy disjunctive query (GFDQ) is a set of general fuzzy conjunctive queries sharing the same head. The disjunction is interpreted as the s-norm (u) of disjuncts.

In what follows, we give some examples of the semantic functions for conjunctions and degree-associated atoms.

1. *Fuzzy threshold queries*: If we use t-norms (t) as the semantic function for conjunctions and R-implications (ω_t) [11] as the semantic function for degree-associated

atoms, we get fuzzy threshold queries, in which the degree of truth of q_F under $\mathcal{I}$ is

$$d = \sup_{S' \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}} \{ t_{i=1}^{n} \ \omega_t(k_i, atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']}) \}.$$

Given some S', if for all atoms we have $atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']} \geq k_i$, since $\omega_t(x, y) = 1$ when $y \geq x$ [11], we have $d = 1$; this corresponds to threshold queries introduced earlier. On the other hand, different from threshold queries, if $0 < atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']} < k_i$, then $d \neq 0$ because of use of the R-implication which filters (penalises) the degree $atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']}$ against the fuzzy threshold k_i.

As it was shown in [4] many of the approaches for weighted Boolean queries that have been proposed [3, 20] are actually special cases of fuzzy threshold queries.

2. *Straccia's query language [29]*: It is also a sub-language of the fuzzy threshold query language, where all $k_i = 1$. Since $\omega_t(1, y) = y$ [11], the degree of truth of q_F under $\mathcal{I}$ is

$$d = \sup_{S' \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}} \{ t_{i=1}^{n} \ atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']} \}.$$

3. *Fuzzy aggregation queries*: If we use fuzzy aggregation functions [11], such as $G(x) = \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} k_i}$, for conjunctions and $a(k_i, y) = k_i * y$ as the semantic function for degree-associated atoms, we get fuzzy aggregation queries, in which the degree of truth of q_F under $\mathcal{I}$ is

$$d = \sup_{S' \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}} \frac{\sum_{i=1}^{n} \ k_i * atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']}}{\sum_{i=1}^{n} \ k_i}.$$

Moreover, we can show that many existing approaches of weighted Boolean queries could be represented under the framework of fuzzy aggregation queries. More precisely, Salton, Fox and Wu [21] proposed the model of p-norms where the semantic function is given by the following equation:

$$d = \sup_{S' \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}} \left(\frac{\sum_{i=1}^{n} \ (atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']})^w}{n} \right)^{1/w}$$

where $w_1 = w_2 = \ldots = w_n = w$ and $w \in (0, +\infty]$. On the other hand, S.J. Chen and S.M. Chen [6] propose the *geometric mean* [11] as a semantic function for weighted Boolean queries:

$$d = \sup_{S' \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}} \left(\prod_{i=1}^{n} \ atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']} \right)^{1/w}.$$

Under the framework of fuzzy aggregation functions both these equations are special cases of the *generalized mean* function which is given by the equation $d_w = \left(\frac{\sum_{i=1}^{n} a_i^w}{n} \right)^{1/w}$ where $w \in \mathbb{R}^*$. If $w \in (0, +\infty]$ then we have the approach of Salton et. al. [21], while if $w \to 0$, then function d converges to the geometric mean [11].

4. *Fuzzy weighted t-norm queries*: If we use generalised weighted t-norms [7] as the semantic function for conjunction, we get fuzzy weighted queries, in which the degree of truth of q_F under $\mathcal{I}$ is

$$d = \sup_{S' \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}} \{ \min_{i=1}^{n} \ u(\overline{k} - k_i, t(\overline{k}, atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']})) \},$$

where $\overline{k} = \max_{i=1}^{n} k_i$. The main idea of this type of queries is that they provide an aggregation type of operation, on the other hand an entry with a low value for a low-weighted criterion should not be critically penalized. Moreover, lowering the weight of a criterion in the query should not lead to a decrease of the relevance score, which should mainly be determined by the high-weighted criteria (see [7] for more details).

Yager [34], proposes the use of and S-implication [11] (in contrast to R-implications of fuzzy threshold queries), i.e. the function:

$$d = \sup_{S' \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}} \{ \min_{i=1}^{n} \ u(1 - k_i, atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']}) \},$$

This is a special case of fuzzy weighted t-norms, where $\overline{k} = 1$, since $t(1, a) = a$. A similar approach was proposed by Sanchez [23].

It is easy to show that all the above four specific fuzzy query languages satisfy the condition (2).

3.2 Supporting Querying with SPARQL

After presenting the abstract syntax and semantics of our proposed languages, in this section, we show how to extend the syntax of SPARQL [19], a well known Semantic Web query language, for the proposed languages. We call our extension f-SPARQL. SPARQL is a query language (candidate recommendation from the W3C Data Access Working Group) for getting information from RDF graphs. SPARQL allows for a query to constitute of triple patterns, conjunctions, disjunctions and optional patterns. A SPARQL query is a quadruple $Q = (V, P, DS, SM)$, where V is a *result form*, P is a *graph pattern*, DS a *data set* and SM a set of solution modifiers. Among others, SPARQL allows for select queries, formed in a SELECT-FROM-WHERE manner. The result form represents the set of variables appearing in the SELECT, the dataset forms the FROM part, constituted by a set of IRIs of RDF documents, while the graph pattern forms the WHERE part which is constituted by a set of RDF triples.

Query	::=	Prologue (SelectQuery | ConstructQuery | DescribeQuery | AskQuery)
SelectQuery	::=	'SELECT' ('DISTINCT' | 'REDUCED')? (Var+ | '*') DatasetClause* WhereClause SolutionModifier
WhereClause	::=	'WHERE'? GroupGraphPattern
GroupGraphPattern	::=	'{' TriplesBlock? ((GraphPatternNotTriples | Filter) '.'? TriplesBlock?)* '}'

In order to maintain backward compatibility with existing SPARQL tools, we propose to use specially formatted SPARQL comments to specify extra information needed in our proposed languages (see Table 2). Firstly, one should declare the query type before a select query. For example, #TQ# declares a threshold query, while #GFCQ:SEM=FUZZY THRESHOLD# declares a general fuzzy query, with the fuzzy threshold semantic functions. Secondly, following each triple in the WHERE clause, one can use #TH# (resp. #DG#) to specify a threshold in a threshold query (resp. a degree in

Query	::=	Prologue (*QueryType* SelectQuery | ConstructQuery | DescribeQuery | AskQuery)
TriplesBlock	::=	TriplesSameSubject ('.' *TripleWeight Degree* TriplesBlock?)?
QueryType	::=	'#TQ# \n' | '#GFCQ:SEM=' *FuzzySemantics* '# \n'
FuzzySemantics	::=	'AGGREGATION' | 'FUZZYTHRESHOLD' | 'FUZZYTHRESHOLD-1' | 'FUZZYWEIGHTEDNORMS'
TripleWeight	::=	'#TH#' | '#DG#'
degree	::=	real-number-between-0-and-1-upper-inclusive

Table 2: Syntax of Fuzzy SPARQL

a general fuzzy query). For instance, the threshold query presented in Example 1 (Section 3.1) can be represented by the following f-SPARQL query:

```
#TQ#
SELECT ?x WHERE {
  ?x rdf:type Model .   #TH# 1.0
  ?x rdf:type Tall .    #TH# 0.7
  ?x rdf:type Light .   #TH# 0.8
}
```

In the case of general fuzzy queries, one must specify the semantic functions (i.e. a and G). Below is an example fuzzy threshold query.

```
#GFCQ:SEM=FUZZYTHRESHOLD#
SELECT ?x WHERE {
  ?x rdf:type Model .    #DG# 1.0
  ?x rdf:type Tall .    #DG# 0.7
  ?x rdf:type Light .    #DG# 0.8
}
```

Table 2 presents the f-SPARQL syntax. f-SPARQL extends two of SPARQL's elements, namely the "Query" and the "TriplesBlock" element. As illustrated above, each select query is extended with the element *QueryType*. In particular, for general fuzzy queries, the declaration '#GFCQ:SEM=' is followed by the element *FuzzySemantics*, which is used to specify the semantic functions, such as the ones we presented in the previous section. More precisely, we use the keywords 'FUZZYTHRESHOLD', 'FUZZYTHRESHOLD-1', 'AGGREGATION' and 'FUZZYWEIGHTEDNORMS' to indicate the four fuzzy general queries we introduced in Section 3.1.2. When one uses 'FUZZYTHRESHOLD-1', the fuzzy threshold is set as 1, and the values specified by the #TH# comments are ignored. Finally, the "TriplesBlock" element is extended with the elements *TripleWeight* and *Degree*, which are used to associated a threshold or weight with each triple of the SPARQL query.

3.3 Query Answering

It should be noted that the query languages in the previous sections can be used with any fuzzy ontology languages. In order to provide efficient query answering services using our proposed query languages, we choose f-DL-Lite as our fuzzy ontology language. This sub-section provides algorithm to answer threshold queries and general fuzzy queries over f-DL-Lite ontologies.

Algorithms for answering queries in f-DL-Lite mainly consist of four steps (like the algorithm for crisp DL-Lite [5]): (i) normalisation of the set $\mathcal{T}$ of the class axioms of $\mathcal{O}$ by the procedure $\mathsf{Normalise}(\mathcal{T})$, which returns the normalised set $\mathcal{T}'$ of class axioms; (ii) normalisation and storage of the set $\mathcal{A}$ of individual axioms in $\mathcal{O}$ by the procedure $\mathsf{Store}(\mathcal{A})$ that normalise $\mathcal{A}$ and returns the relational database $\mathrm{DB}(\mathcal{A})$ of $\mathcal{A}$, as well as checking the consistency of $\mathcal{O}$ by the procedure $\mathsf{Consistency}(\mathcal{O}, \mathcal{T}')$; (iii) reformulation of the input query q against the normalised set $\mathcal{T}$ of the class axioms by the procedure $\mathsf{PerfectRef}(q, \mathcal{T}')$, which returns a set Q of (conjunctive) queries; (iv) transformation of the set Q of (conjunctive) queries into SQL queries by the procedure $\mathsf{SQL}(Q)$, as well as the evaluation of $\mathsf{SQL}(Q)$ by the procedure $\mathsf{Eval}(\mathsf{SQL}(Q), \mathrm{DB}(\mathcal{A}))$.

As steps (i) and (ii) are very similar to those for the crisp case, here we focus on steps (iii) and (iv) on answering conjunctive threshold queries and general fuzzy queries.

3.3.1 Answering Threshold Queries

Given an f-DL-Lite ontology $\mathcal{O}$, a conjunctive threshold query q_T, the procedure $\mathsf{Answer}_\mathsf{T}(\mathcal{O}, q_T)$ computes the solutions of q_T w.r.t. $\mathcal{O}$, following the above steps (i) - (iv).

Algorithm A-1: $\mathsf{Answer}_\mathsf{T}(\mathcal{O}, q_T)$
1: $\mathcal{T} = \mathsf{Class\text{-}Axioms}(\mathcal{O})$
2: $\mathcal{T}' = \mathsf{Normalise}(\mathcal{T})$ //normalisation of class axioms
3: $\mathcal{A} = \mathsf{Individual\text{-}Axioms}(\mathcal{O})$
4: $\mathrm{DB}(\mathcal{A}) = \mathsf{Store}(\mathcal{A})$ //normalisation and storage of individual axioms
5: **if** $\mathsf{Consistency}(\mathcal{O}, \mathcal{T}') =$ **false then**
6: **return** *inconsistent* //$\mathcal{O}$ is inconsistent
7: **end if**
8: **return** $\mathsf{Eval}(\mathsf{SQL}_\mathsf{T}(\mathsf{PerfectRef}_\mathsf{T}(q_T, \mathcal{T}')), \mathrm{DB}(\mathcal{A}))$

Algorithm A-2: $\mathsf{SQL}_\mathsf{T}(\mathrm{Q})$
1: $QS := \emptyset$
2: **for** every query q in Q **do**
3: sc:=$\mathsf{Select\text{-}Clause}(q)$ //construct the select-clause of q
4: fc:=$\mathsf{From\text{-}Clause}(q)$ //construct the from-clause of q
5: wc1:=$\mathsf{WC\text{-}Binding}(q)$ //construct the part of the where-clause about binding
6: wc2:=$\mathsf{WC\text{-}Threshold}(q)$ //construct the part of the where-clause that relates to thresholds
7: $QS := QS \cup \mathsf{Construct\text{-}SQL}$(sc,fc,wc1,wc2)
8: **end for**
9: **return** QS

The algorithms need some explanations. Firstly, if $\mathcal{O}$ is inconsistent, query answering is meaningless, since every tuple is a solution of every query w.r.t. $\mathcal{O}$.

Secondly, the procedure $\mathsf{PerfectRef}_\mathsf{T}(q_T, \mathcal{T}')$ of reformulating an input conjunctive threshold query q_T (into a set of conjunctive queries) is essentially the same as the $\mathsf{PerfectRef}(q, \mathcal{T}')$ for DL-Lite [5]. Here we do not repeat $\mathsf{PerfectRef}(q, \mathcal{T}')$ but explain its main ideas instead. $\mathsf{PerfectRef}_\mathsf{T}(q_T, \mathcal{T}')$ rewrites atoms in q_T based on positive inclusions (PIs) in $\mathcal{T}'$. For example, given a PI $B \sqsubseteq \mathsf{B}_1$, if $\mathsf{B}_1(v) \geq k$ is an atom of

q_T and B is a named class A (resp. of the form $\exists R$, or of the form $\exists R^-$), $\mathsf{B}_1(v) \geq k$ can be rewritten as $\mathsf{A}(v) \geq k$ (resp. $R(v,\_) \geq k$, or $R(\_,v) \geq k$, where $\_$ represent non-distinguished non-shared variables[5]). The generated query can be further written, based on the PIs in $\mathcal{T}'$. It can be shown that such rewriting process always terminates [5], and it produces a set of generated conjunctive queries from the input query q_{TQ}.

Thirdly, the procedure $\mathsf{SQL}_\mathsf{T}(Q)$ transforms a set of conjunctive threshold queries into SQL queries in an obvious way, taking into accounts the thresholds. Namely, it use Select-Clause, From-Clause, WC-Binding and WC-Threshold to construct the select-clauses, from-clauses and where-clauses of SQL queries, respectively. Due to limitation of space, we do not provide full details of these procedures but illustrate them with the following example. Given the query $q(v) \leftarrow \mathsf{Model}(v) \geq 1, \mathsf{Tall}(v) \geq 0.7, \mathsf{Light}(v) \geq 0.8$, Select-Clause($q$) returns the select-clause 'SELECT $tab_{\mathsf{Model}}[0]$', From-Clause(q) returns the from-clause 'FROM tab_{Model}, tab_{Tall}, tab_{Light}', WC-Binding(q) returns the binding part of the where-clause 'WHERE $tab_{\mathsf{Model}}[0] = tab_{\mathsf{Tall}}[0]$, $tab_{\mathsf{Model}}[0] = tab_{\mathsf{Light}}[0]$' and WC-Threshold($q$) returns the threshold-related part of the where-clause '$tab_{\mathsf{Model}}[1] \geq 1$, $tab_{\mathsf{Tall}}[1] \geq 0.7$, $tab_{\mathsf{Light}}[1] \geq 0.8$'. Construct-SQL puts all these together and returns the corresponding SQL query 'SELECT $tab_{\mathsf{Model}}[0]$ FROM tab_{Model}, tab_{Tall}, tab_{Light} WHERE $tab_{\mathsf{Model}}[0] = tab_{\mathsf{Tall}}[0]$, $tab_{\mathsf{Model}}[0] = tab_{\mathsf{Light}}[0]$, $tab_{\mathsf{Model}}[1] \geq 1$, $tab_{\mathsf{Tall}}[1] \geq 0.7$, $tab_{\mathsf{Light}}[1] \geq 0.8$'.

THEOREM 1. *Let $\mathcal{O}$ be an f-DL-Lite ontology, q_T a conjunctive threshold query and S a tuple of constants. S is a solution of q_T w.r.t. $\mathcal{O}$ iff $S \in \mathsf{Answer}_\mathsf{T}(\mathcal{O}, q_T)$.*

Proof: (Sketch) The proof of correctness is straight forward. The procedure $\mathsf{Answer}_\mathsf{T}$ differs from that of DL-Lite mainly in that it needs to take care of the thresholds (line 6 of Algorithm A-2) when constructing SQL queries. Given an evaluation $[X \mapsto S]$ an atom $\mathsf{A}(v) \geq t_1$ ($R(v_1, v_2) \geq t_2$), if $tab_{\mathsf{A}_{[X \mapsto S]}}[1] \geq t_1$ (resp. $tab_{R_{[X \mapsto S]}}[2] \geq t_2$), then we have $\mathsf{A}^{\mathcal{I}}(v)_{[X \mapsto S]} \geq t_1$ (resp. $R^{\mathcal{I}}(v_1, v_2)_{[X \mapsto S]} \geq t_2$). The proof for the other direction is similar.

3.3.2 Answering General Fuzzy Queries

Similarly, given an f-DL-Lite ontology $\mathcal{O}$, a general fuzzy conjunctive query q_F, the procedure $\mathsf{Answer}_\mathsf{F}(\mathcal{O}, q_F)$ computes the solutions of q_F w.r.t. $\mathcal{O}$.

Algorithm A-3: $\mathsf{Answer}_\mathsf{F}(\mathcal{O}, q_F, a, G)$
1: $\mathcal{T}$ = Class-Axioms($\mathcal{O}$)
2: $\mathcal{T}'$ = Normalise($\mathcal{T}$) //normalisation of class axioms
3: $\mathcal{A}$ = Individual-Axioms($\mathcal{O}$)
4: DB($\mathcal{A}$) = Store($\mathcal{A}$) //normalisation and storage of individual axioms
5: **if** Consistency($\mathcal{O}, \mathcal{T}'$) = **false then**
6: **return** *inconsistent* //$\mathcal{O}$ is inconsistent
7: **end if**
8: q = Remove-Degrees(q_F) //q is transformed from q_F by removing the degrees from q_F
9: **return** Cal(q_F, EvalSQL(PerfectRef(q,$\mathcal{T}'$)),DB($\mathcal{A}$)), a, G)

Algorithm A-4: $\mathsf{Cal}(q_F, SS, a, G)$
1: $ANS := \emptyset$
2: **for** every tuple $S \in SS$ **do**
3: $ANS := ANS \cup$ Cal-Soln(q_F, S, a, G) //Calculate the solution $S : d$ based on the semantic functions a and G
4: **end for**
5: **return** ANS

The main tasks of $\mathsf{Answer}_\mathsf{F}$ are (i) to look for candidate solutions in which the degree is larger than 0, and then (ii) to calculate the precise degree. The task (i) can be performed by the DL-Lite query engine, since in the crisp case, the degree is either 0 or 1 (larger than 0). In other words, the DL-Lite procedures Eval, SQL, PerfectRef can be reused here (line 9 of Algorithm A-3). The task (ii) is done by the procedure Cal, which is a general algorithm to calculate the degree of each tuple S in the answer set SS returned by the DL-Lite engine, based on the chosen semantic functions a and G. Accordingly, we have the following theorem.

THEOREM 2. *Let $\mathcal{O}$ be an f-DL-Lite ontology, q_F a general fuzzy conjunctive query and $S : d$ a pair of a tuple of constants together with a truth degree, a a semantic function for conjunctions and G a semantic function for degree-associated atoms. $S : d$ is a solution of q_F w.r.t. $\mathcal{O}$ iff $(S : d) \in \mathsf{Answer}_\mathsf{F}(\mathcal{O}, q_F, a, G)$.*

4. IMPLEMENTATION AND EVALUATION

4.1 Implementation

Our implementation is based on the ONTOSEARCH2 system[6] [16, 31], which is an infrastructure for supporting ontology searching and query answering. The f-DL-Lite query engine is implemented as an extension of the crisp DL-Lite query engine in ONTOSEARCH2 [15], so as to support threshold queries and general fuzzy queries. The core part of the f-DL-Lite query engine includes implementations of Algorithms A-1 to A-4, which are presented in Section 3.3. The system was written in Java 5 and uses PostgreSQL 8.1 RDBMS for the repository storage. PostgreSQL was setup with default installation, no additional configuration was performed.

Users of the f-DL-Lite query engine can submit f-DL-Lite ontologies via the Web interface of ONTOSEARCH2[6], and then submit f-SPARQL queries against their target ontologies. The fuzzy query engine operates in two modes: TQ mode (for threshold queries) and GFCQ mode (for general fuzzy queries). When users submit an f-SPARQL query, the fuzzy query engine parses it, so as to determine the query type (whether the query is a threshold query or a general fuzzy query), as well as the thresholds (for threshold queries) or degrees (for general fuzzy queries), depending on the query types. Accordingly, the fuzzy query engine operates in either TQ mode (Algorithms A-1 and A2) or GFCQ mode (Algorithms A-3 and A-4).

Besides the DL-Lite query engine and the f-DL-Lite query engine, the ONTOSEARCH2 system consists of other components, such as the ontology search engine. In Section 5, we will show how the ontology search engine uses the f-DL-Lite query engine to perform keyword-plus-entailment searches.

4.2 Benchmark and Evaluation

In this section, we present some preliminary evaluation of the f-DL-Lite query engine presented in Section 4.1. We will first discuss the benchmark that we used in the evaluation, and then present the the evaluation results.

[5]A variable is *non-shared* if it does not appear in the body of the query for more than once.

[6]http://www.ontosearch.org/

To the best of our knowledge, there exists no benchmark for query answering over fuzzy ontologies that we could use to evaluate our f-DL-Lite query engine. Accordingly, we propose a fuzzy variant of the well known Lehigh University Benchmark (LUBM), called f-LUBM, which is the first of its kind and against which future implementation of f-DL-Lite query engines can also be evaluated. f-LUBM allows the use of fuzzy classes and restricts the expressive power of the underlying ontology to that of f-DL-Lite. More precisely, we added two fuzzy classes to the LUBM University ontology, namely "Busy" and "Famous". The former one is determined by the number of courses taught or taken by a member of staff or student, while the latter one is determined by the number of papers published. The values are calculated using the s-shaped curve functions $k_f(n)$ to calculate the fuzzy value for fame given n papers published, and $k_b(n)$ to calculate the fuzzy value for busyness given n courses taken:

$$k_f(n) = \frac{2}{1+exp(-0.1n)} - 1, \quad k_b(n) = \frac{2}{1+exp(-0.4n)} - 1$$

Based on the above two fuzzy classes, 4 extra queries are introduced to f-LUBM (there are 14 queries in LUBM). The first two are simple queries that ask for famous ones.

$$\begin{aligned} \text{f-LUBM-Q15}(v) \leftarrow \quad & \mathsf{Famous}(v) \geq 0.5 \\ \text{f-LUBM-Q16}(v) \leftarrow \quad & \mathsf{Famous}(v) : 0.5 \end{aligned}$$

f-LUBM-15 is a threshold query, while f-LUBM-16 is a general fuzzy query. The other two queries ask for all busy students which were taught by famous members of staff.

$$\begin{aligned} \text{f-LUBM-Q17}\ (v1) \leftarrow \quad & \mathsf{Student}(v1), \mathsf{Buzy}(v1) \geq 0.5, \mathsf{Faculty}(v2), \\ & \mathsf{Famous}(v2) \geq 0.5, teacherOf(v2, v3), \\ & takesCourse(v1, v3) \\ \text{f-LUBM-Q18}\ (v1) \leftarrow \quad & \mathsf{Student}(v1), \mathsf{Buzy}(v1) : 0.5, \mathsf{Faculty}(v2), \\ & \mathsf{Famous}(v2) : 0.5, teacherOf(v2, v3), \\ & takesCourse(v1, v3) \end{aligned}$$

Like in LUBM, it is possible to create arbitrarily large datasets for individual axioms, generated by a Java program in f-LUBM (http://www.csd.abdn.ac.uk/∼sttaylor/f-LUBM.zip). To test the f-DL-Lite query engine, we created datasets containing 1, 10 and 50 universities, with the largest data set (for 50 universities) containing 6,888,642 individuals. We used fuzzy aggregation queries as representatives for GFCQs in our test. In order to investigate the overhead of fuzzy queries, we compare the performance in the f-DL-Lite query engine with the DL-Lite query engine, which is used to answer the following two crisp queries.

$$\begin{aligned} \text{crisp-1}(v) \leftarrow \quad & \mathsf{Famous}(v) \\ \text{crisp-2}(v1) \leftarrow \quad & \mathsf{Student}(v1), \mathsf{Buzy}(v1), \mathsf{Faculty}(v2), \\ & \mathsf{Famous}(v2), teacherOf(v2, v3), \\ & takesCourse(v1, v3) \end{aligned}$$

The results are shown in Table 3. The first column lists the queries used in the test. The second (resp. third and fourth) column show the time (in millisecond) needed to answer the queries for 1 university (resp. 10 and 50 universities). In general, the evaluations match nicely with our expectation: crisp queries are faster than CTQs as the system needs to take care of more joins due to the thresholds, while CTQs are faster than GFCQs as the former ones do not require post-processing to calculate the degrees. Furthermore, it is encouraging to see that the performance of the fuzzy query engine is in most cases close to the performance of the crisp query engine. With the smallest data set, it is has almost identical performance, particularly on the more complex queries. As more data must be evaluated, the performance drops slightly. For the largest data set (containing 6,888,642 individuals), it only took the f-DL-Lite query engine (up to) a few seconds to answer each of the tested queries.

Table 3: Results of some f-LUBM queries

Query	T[1] (ms)	T[10] (ms)	T[50] (ms)
f-LUBM-Q15 (TQ)	179	536	1061
f-LUBM-Q16 (GFCQ)	220	683	1887
crisp-1	152	422	891
f-LUBM-Q17 (TQ)	532	845	2922
f-LUBM-Q18 (GFCQ)	520	973	3654
crisp-2	494	892	2523

5. USE CASE: ONTOLOGY SEARCHING

This section presents an online application[6], the ontology search engine in the ONTOSEARCH2 system, of the f-DL-Lite query engine presented in Section 4. One of the major limitations of existing ontology search engines is that searching is *only* based on keywords and metadata information of ontologies, rather than semantic entailments of ontologies (e.g., one wants to search for ontologies in which BassClarinet is a sub-class of Woodwind). On the other hand, searching only based on semantic entailments might not be ideal either, as synonyms appearing in the metadata could not be exploited.

By making use of the f-DL-Lite query engine, our ontology search engine supports keyword-plus-entailment searches, such as *searching for ontologies in which class X is a sub-class of class Y, and class X is associated with the keywords "Bass" and "Clarinet", while class Y is associated with the keyword "Woodwind"*. The search could be represented as the following threshold query:

```
1: #TQ#
2: SELECT ?x WHERE {
3:  ?x hasKeyword i-bass . #TH# 0.6
4:  ?x hasKeyword i-clarinet . #TH# 0.6
5:  ?x rdfs:subClassOf ?y .
6:  ?y hasKeyword i-woodwind . #TH# 0.7}
```

where i-bass, i-clarinet and i-woodwind are representative individuals for keywords "Bass", "Clarinet" and "Woodwind", resp. The thresholds 0.6 and 0.7 can be specified by users.

In order to support keyword-plus-entailment searches, our ontology search engine, for each indexed ontology, stores its semantic approximation (in DL-Lite) [15] and accompanies each ontology in its repository with an f-DL-Lite *meta-ontology*, which (i) materialises all TBox reasoning based on the semantic approximation and, most importantly, (ii) uses fuzzy assertions to represent associations of each class (property) and keywords[7] appearing in the metadata of the ontology, with some degrees. Keywords appearing in the ontology metadata are associated with scores based on ranking factors[8]. We use these scores to calculate the $tf \cdot idf$ [22] for

[7]As mentioned above, keywords are represented by representative individuals.

[8]http://www.seomoz.org/article/search-ranking-factors

each keyword, and normalise them using a sigmoid function such as the one shown in (3) to a degree between 0 and 1.

$$w(n) = \frac{2}{1.2^{-n} + 1} - 1 \quad (3)$$

Hence, the ontology search engine can use the f-DL-Lite query engine to query across all the meta-ontologies in its repository, so as to support keyword-plus-entailment searches. Further discussions of this use case go beyond the scope of this paper.

6. DISCUSSIONS

How to apply Description Logic based ontologies in the Web has been a pressing issue for the Semantic Web community [14]. Web applications require ontology engines to be able to handle fuzzy and imprecise data in an efficient and scalable manner, which has been a big challenge for existing fuzzy ontology engines. In this paper, we tackle this issue by providing efficient and scalable query services for lightweight fuzzy ontologies (in f-DL-Lite), based on the the two novel query languages we proposed for fuzzy ontologies. To the best of our knowledge, this paper is the first to report on implementations and evaluations of scalable and expressive conjunctive query answering over fuzzy ontologies. Our online use case application shows that the f-DL-Lite query engine can be used to enable keyword-plus-entailment searches.

Our proposed query languages are related with weighted Boolean queries [33, 4, 21] proposed in the field fuzzy information retrieval [8]. On the one hand, motivated by these extensions we propose a completely novel query language, that of threshold queries, which is very important since it generalises the entailment of fuzzy assertions. On the other hand, the main strength of the proposed general fuzzy query language is its openness of the use of semantics generalising most previous approaches, like those based on fuzzy implications [4], aggregation functions [21, 6, 32], Straccia's query language [29] for fuzzy DL-Lite and weighted t-norm queries [7]. With the general fuzzy query language, we could provide *top-k* query answering service over f-DL-Lite ontologies, similar to that proposed by Straccia [30] for a variant of DL-Lite ontologies. In short, the openness of the use of fuzzy semantics and the generality of the presented algorithms make the fuzzy querying service more adaptable for different application needs.

One aspect of our future work is to investigate scalable querying services for more expressive fuzzy ontology languages, such as Fuzzy OWL [26]. Another part of our future work should also consist of a survey on which query language/semantics could/should be used in what scenarios in Web applications.

Acknowledgements

This research was partially supported by the Nuffield FONTO-Rule project (NAL/32794), the FP6 Network of Excellence EU project Knowledge Web (IST-2004-507842), Integrated EU project X-Media (IST-2004-026978) and the Advanced Knowledge Technologies (GR/N15764/01). Giorgos Stoilos was also partially supported by the Greek Secretariat of Research and Technology (PENED Ontomedia 03 ED 475).

7. REFERENCES

[1] F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider, editors. *Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2002.

[2] F. Bobillo, M. Delgado, and J. Gómez-Romero. A crisp representation for fuzzy $\mathcal{SHOIN}$ with fuzzy nominals and general concept inclusions. In *Proc. of the 2nd International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 06), Athens, Georgia*, 2006.

[3] A. Bookstein. Fuzzy requests: An approach to weighted boolean searches. *Journal of the Americal Society for Information Science*, 31:240–247, 1980.

[4] G. Bordogna, P. Bosc, and G Pasi. Fuzzy inclusion in database and information retrieval query interpretation. In *Proceedings of the 1996 ACM symposium on Applied Computing*, pages 547–551, 1996.

[5] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, 2005.

[6] S.J. Chen and S.M. Chen. A new method for fuzzy information retrieval based on geometric-mean averaging operators. In *Workshop on Artificial Intelligence*.

[7] A. Chortaras, G. Stamou, and A. Stafylopatis. Adaptation of weighted fuzzy programs. In *Proc. of the International Conference on Artificial Neural Networks (ICANN 2006)*, pages 45–54. Springer, 2006.

[8] V. Cross. Fuzzy information retrieval. *Journal of Intelligent Information Systems*, 3:29–56, 1994.

[9] B. C. Grau. Tractable fragments of the OWL 1.1 web ontology language. `http://www.w3.org/Submission/owl11-tractable/`, 2006.

[10] Y. Guo, Z. Pan, and J. Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. *Journal of Web Semantics*, 3(2):158–182, 2005.

[11] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, 1995.

[12] Y. Li, B. Xu, J. Lu, and D. Kang. Discrete tableau algorithms for $\mathcal{FSHI}$. In *Proceedings of the International Workshop on Description Logics (DL 2006), Lake District, UK*, 2006.

[13] T. Mailis, G. Stoilos, and G. Stamou. Proceedings of the first international conference on web reasoning and rule systems (RR-07), 2007. In *Expressive Reasoning with Horn Rules and Fuzzy Description Logics*, 2007.

[14] P. Mika. Ontologies are us: A unified model of social networks and semantics. In *4th International Semantic Web Conference (ISWC 2005)*, 2005.

[15] J. Z. Pan and E. Thomas. Approximating OWL-DL Ontologies. In *Proc. of the 22nd National Conference on Artificial Intelligence (AAAI-07)*, 2007. To appear.

[16] J. Z. Pan, E. Thomas, and D. Sleeman. ONTOSEARCH2: Searching and Querying Web Ontologies. In *Proc. of WWW/Internet 2006*, pages 211–218, 2006.

[17] J.Z. Pan, G. Stoilos, G. Stamou, V. Tzouvaras, and I. Horrocks. f-SWRL: A fuzzy extension of SWRL.

Journal on Data Semantics, Special Issue on Emergent Semantics, 4090:28–46, 2006.

[18] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C, Feb. 2004. W3C Recommendation.

[19] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF, 2006. W3C Working Draft, http://www.w3.org/TR/rdf-sparql-query/.

[20] T. Radecki. Fuzzy set theoretical approach to document retrieval. *Journal of Information Processing & Management*, 15:235–245, 1979.

[21] G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *Journal of Communications of ACM*, 26:1022–1036, 1983.

[22] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.

[23] E. Sanchez. Importance in knowledge systems. *Information Systems*, 14(6):455–464, 1989.

[24] G. Stoilos, N. Simou, G. Stamou, and S. Kollias. Uncertainty and the semantic web. *IEEE Intelligent Systems*, 21(5):84–87, 2006.

[25] G. Stoilos, G. Stamou, J. Z. Pan, V. Tzouvaras, and I. Horrocks. Reasoning with very expressive fuzzy description logics. *Journal of Artificial Intelligence Research*, 30(5):273–320, 2007.

[26] G. Stoilos, G. Stamou, V. Tzouvaras, J.Z. Pan, and I. Horrocks. Fuzzy OWL: Uncertainty and the semantic web. In *Proc. of the International Workshop on OWL: Experiences and Directions*, 2005.

[27] U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

[28] U. Straccia. Description logics with fuzzy concrete domains. In *21st Conf. on Uncertainty in Artificial Intelligence (UAI-05)*, Edinburgh, 2005.

[29] U. Straccia. Answering vague queries in fuzzy DL-Lite. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, (IPMU-06)*, pages 2238–2245, 2006.

[30] U. Straccia. 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)*, 2006.

[31] E. Thomas, J. Z. Pan, and D. Sleeman. ONTOSEARCH2: Searching Ontologies Semantically. In *Proc. of OWL Experience Workshop*, 2007.

[32] P. Vojtás. Fuzzy logic programming. *Fuzzy Sets and Systems*, 124:361–370, 2001.

[33] W.G. Waller and D.H. Kraft. A mathematical model of a weighted boolean retrieval system. *Journal of Information Processing & Management*, 15:247–260, 1979.

[34] R.R Yager. A note on weighted queries in information retrieval systems. *Journal of the Americal Society for Information Science*, 38:23–24, 1987.

APPENDIX

A. ABSTRACT SYNTAX OF F-DL-LITE

This appendix contains a detailed presentation of the abstract syntax of fuzzy-DL-Lite. Firstly, we present the abstract syntax of crisp DL-Lite by restricting the elements of the OWL DL syntax. Then, we provide the definition of individual axioms in fuzzy-DL-Lite which additionally contain a membership degree and an inequality type.

Class Axioms

```
axiom        ::= 'Class(' classID ['Deprecated'] 'complete'
                 {annotation} { basicClass } ')'
axiom        ::= 'Class(' classID ['Deprecated'] 'partial'
                 {annotation} { generalClass } ')'
axiom        ::= 'DisjointClasses(' basicClass basicClass
                                              { basicClass } ')' |
                 'EquivalentClasses(' basicClass { basicClass } ')' |
                 'SubClassOf(' basicClass generalClass ')'
basicClass   ::= classID | restriction
generalClass ::= basicClass | 'complementOf(' { basicClass } ')' |
                 'intersectionOf(' { generalClass } ')'
restriction  ::= 'restriction(' individualvaluedPropertyID
                                'someValuesFrom( owl:Thing )' ')'
```

Property Axioms

```
axiom ::= 'ObjectProperty(' individualvaluedPropertyID
          ['Deprecated'] { annotation }
          [ 'inverseOf(' individualvaluedPropertyID ')' ]
          ['Functional' | 'InverseFunctional' ]
          { 'domain(' generalClass ')' } { 'range(' generalClass ')' } ')'
```

Fuzzy Assertions

```
individual ::= 'Individual(' [individualID] {annotation}
               {'type'( type ')' [ineqType] [degree]}
               {value [ineqType] [degree] } ')'
ineqType   ::= '>='
degree     ::= real-number-between-0-and-1-inclusive
value      ::= 'value(' individualvaluedPropertyID individualID ')' |
               'value(' individualvaluedPropertyID individual ')'
type       ::= description
```