FSA-SPARQL: Fuzzy Queries in SPARQL

Jesús M. Almendros-Jiménez, Antonio Becerra-Terón
Dept. of Informatics
Universidad de Almería
04120 Almería (Spain)
jalmen@ual.es, abecerra@ual.es

Ginés Moreno
Dept. of Computing Systems
University of Castilla-La Mancha
02071 Albacete (Spain)
Gines.Moreno@uclm.es

SPARQL has been adopted as query language for the Semantic Web. RDF and OWL have also been established as vocabularies to describe ontologies in this setting. While RDF/OWL/SPARQL have been designed for querying crisp (i.e., non-fuzzy) information, some contexts require to manage uncertainty, vagueness and imprecise knowledge. In this paper we propose a SPARQL extension, called FSA-SPARQL (*Fuzzy Sets and Aggregators based SPARQL*) in which queries can involve different fuzzy connectives and (aggregation) operators. The language has been implemented as an extension of the ARQ Jena SPARQL engine and it is equipped with a Web tool from which queries can be executed on-line.

1 Introduction

The Semantic Web has adopted *SPARQL* [20, 22] as query language. *RDF* (and *OWL*) querying is a key element in the Semantic Web era, where many data resources are available from the Web (for instance, DBpedia[1], YAGO[2] and Open Street Map[3]). The *Linked Data* initiative[4] arises in this context to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF. A considerable number of SPARQL endpoints[5] are publicly reachable, and applications access Web data thanks to SPARQL mechanisms. Implementations of SPARQL[6] have been carried out by several institutions and vendors.

SPARQL has as main component *triple patterns subject-property-object* which makes possible to match RDF triples, and variables occurring in triple patterns are then filtered by Boolean conditions. The *FILTER* statement of SPARQL is used to express Boolean conditions on these variables. Ontologies (in RDF and OWL vocabularies) represent the knowledge about a certain topic of interest, and queries on ontologies aim to retrieve certain elements of the ontology. While RDF/OWL/SPARQL have been designed for querying crisp information, some contexts require to deal with uncertainty, vagueness and imprecise knowledge [25]. Fuzzy systems [1, 19] offer a richer mechanism for knowledge representation and many areas need to handle fuzzy components.

We can distinguish three lines of research in this context. Firstly, *extensions of RDF and OWL vocabularies* to cover with fuzziness (see [25, 17] for surveys). The most prominent contributions are [10, 24, 11]. OWL 2 [13] and Description Logic (DL) [6] have been extended to express fuzzy (*concept membership*, *role fulfilling*, *subconcept* and *subrole*, among others) relationships in [10] and [24]. In [11]

[1] http://wiki.dbpedia.org/
[2] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/
[3] http://owl.openstreetmap.org/
[4] http://linkeddata.org/
[5] https://www.w3.org/wiki/SparqlEndpoints
[6] https://www.w3.org/wiki/SparqlImplementations

Submitted to:
PROLE 2017

aggregation operators have also been integrated in OWL and DL. Secondly, fuzzy DL reasoners. The most relevant *fuzzy reasoners* are *fuzzyDL* [9], *DeLorean* [8] and *Fire* [23]. And finally, *querying of fuzzy ontologies* with SPARQL extensions [12, 26, 16, 23, 7].

Here, we focus on a SPARQL extension to cover with fuzziness. SPARQL fuzzy extensions (see [21] for a survey) are mainly represented by *f-SPARQL* [12, 26, 16]. In f-SPARQL fuzzy terms (for instance, *RECENT*, *HIGH* and *VERY*), and fuzzy operators (for instance, *CLOSE TO* and *AT LEAST*) might appear in FILTER conditions to relax SPARQL queries. Fuzzy predicates are based on trapezoidal membership relations. RDF data can contain in f-SPARQL fuzzy values in the form of real numbers of the *[0,1]* interval for some data property. Weighted mean is used to combine partial scores and the weights are specified by the *WITH* statement in FILTER conditions. For instance, the following query:

```
SELECT ?Movie ?Actor ?Date ?Rating
WHERE {
?Actor movie:actor-name "George Clooney".
?Movie movie:actor ? Actor .
?Movie dc:date ? Date .
?Movie dc:rate ? Rating .
FILTER (?Date = recent) WITH 0.8.
(?Rating = high) WITH 0.2.
}
```

retrieves the recent (importance 0.2) movies featuring George Clooney with a high rating (importance 0.8). *Top-k* solutions can be reported in f-SPARQL. f-SPARQL queries are translated into Boolean SPARQL queries, which can be executed by any SPARQL interpreter.

On the other hand, the *FSAQL* query language has been proposed in [7], which resembles SPARQL and it is able to query fuzzy values assigned to RDF statements. Additionally, the authors of [23] propose a SPARQL-based encoding of conjunctive queries expressed in fuzzy DL. Top-k [18] and *skyline* queries [14] have also been studied in extensions of SPARQL. Finally, the *PrefSPARQL* extension of SPARQL proposed in [14] introduces user preferences in the FILTER clause with the *PREFERRING* and *PRIOR TO* clauses as well as with the introduction of *HIGHEST*, *LOWEST*, *IF-THEN-ELSE*, *BETWEEN*, *AROUND*, *MORE THAN* and *LESS THAN* operators.

In this paper we propose a SPARQL extension, called *FSA-SPARQL* (*Fuzzy Sets and Aggregators based SPARQL*) in which queries can involve different fuzzy (*Product*, *Łukasiewicz* and *Gödel*) connectives, fuzzy operators *VERY*, *MORE_OR_LESS*, *CLOSE_TO*, *AT_MOST* and *AT_LEAST*, as well as fuzzy aggregation operators also called aggregators: *MEAN*, *WSUM*, *WMIN* and *WMAX*. FSA-SPARQL works on fuzzy RDF data, handling fuzzy concept membership, fuzzy role fulfilling, fuzzy subconcept and fuzzy subrole relationships. Thus, FSA-SPARQL is a fuzzy query language on fuzzy data. FSA-SPARQL queries are translated into crisp (that is, standard or non-fuzzy) SPARQL queries, and thus FSA-SPARQL queries can be executed by any SPARQL interpreter. Additionally, FSA-SPARQL enables also to specify a threshold on queries.

FSA-SPARQL amalgamates the proposed f-SPARQL [12] operators (i.e., *VERY*, *MORE_OR_LESS*, *CLOSE_TO*, *AT_MOST* and *AT_LEAST*) together with the fuzzy OWL 2 aggregation operators [11] (i.e., *MEAN*, *WSUM*, *WMIN* and *WMAX*). FSA-SPARQL is inspired by our previous works on XPath/XQuery and the fuzzy (multi-adjoint) logic programming language MALP [3, 2, 5, 4]. While in the case of XPath (and XQuery) the goal was to incorporate fuzzy logic connectives and aggregators into both languages in order to work with XML data, here our proposal aims to represent data and queries by RDF and SPARQL, respectively.

The differences with regard to f-SPARQL [12] is that here we deal with fuzzy concept membership,

fuzzy role fulfilling, fuzzy subconcept and fuzzy subrole relationships. For instance, we can now express that *George Clooney* is an *actor* with a degree of 1, while he is a *TV movie actor* with a degree of 0.4. We can also express that *George Clooney* is the *leading role* of *Ocean's Eleven* with a degree of 0.4. Also we can express that a *movie producer* is a subconcept of artist with a degree of 0.4, as well as a *cast partner* is a subrole of *friend* with a degree of 0.4. This kind of fuzzy specifications cannot be defined in f-SPARQL. Additionally, connectives coming from standard fuzzy logics (i.e., *Łukasiewicz*, *Product* and *Gödel* logics) and fuzzy aggregators (i.e., *MEAN*, *WSUM*, *WMIN* and *WMAX*) cannot be used in f-SPARQL. Our language has also greater expressivity power than FSAQL [7] enabling the use of fuzzy connectives and (aggregation) operators.

With regard to data, RDF data has been equipped with *truth values* associated to concept membership, role fulfilling, subconcept relationships and subrole relationships. Thus, our data model is similar to the FSAQL data model. We have adopted a simple data model, but it has several advantages. Firstly, the data model has a RDF/XML serialization which is compatible with *Protégé* [15] and other ontology tools, and also fuzzy data can be edited and partially debugged [7] (via ontology reasoners) with Protégé. The fuzzy OWL 2 proposal of [10] uses *OWL annotations* to specify fuzzy data, and the RDF/XML serialization is compatible with *Protégé* but, since editing is not supported by *Protégé*, a plugin must be used.[8] In the case of [24], the RDF/OWL serialization is not compatible with *Protégé*. Secondly, SPARQL queries (and FSA-SPARQL queries) can be easily expressed against our data model. The authors of [10] have not still studied how to use SPARQL for expressing queries against their fuzzy OWL 2 proposal. The authors are mainly concerned at using a fuzzy DL reasoner on fuzzy OWL 2 data. Fuzzy OWL annotations, which are essentially XML annotations, make harder the specification of queries with SPARQL. This is the main reason we have not adopted the fuzzy OWL 2 proposal. While fuzzy OWL 2 offers a wide range of mechanisms to describe fuzzy data, our approach is restricted to truth values (real numbers of the [0,1] interval) associated to *rdf:type*, *rdfs:subClassOf* and *rdfs:subPropertyOf* axioms, as well as to object/data property relationships. Our proposal is also inspired by [11], where authors introduce fuzzy sets and aggregators in DL. However, our work is focused on extending SPARQL with fuzzy aggregators, in such a way that the user can play with them to express preferences.

We have developed a Web tool[9] from which FSA-SPARQL can be executed on-line. The implementation of FSA-SPARQL has been carried out on top of ARQ Apache Jena engine.[10] On the one hand, fuzzy connectives and operators have been incorporated to ARQ Jena using the *Expression Functions* mechanism. On the other hand, FSA-SPARQL queries are translated into crisp SPARQL enriched by fuzzy connectives and operators. In order to provide semantics to FSA-SPARQL we will define a mapping from FSA-SPARQL to (pure) crisp SPARQL.

The structure of the paper is as follows. Section 2 motivates the framework with some examples. In Section 3 we describe the extension of RDF to cover fuzziness. Section 4 presents FSA-SPARQL and its semantics based on a FSA-SPARQL to crisp SPARQL transformation as well as the developed Web tool to execute FSA-SPARQL queries on-line. Finally, in Section 5 we conclude and present future work.

[7] Current crisp (non-fuzzy) OWL reasoners can be used to partially debug our fuzzy RDF, while a more complete and sophisticated mechanism is out of the scope of the paper, and it is considered as future work.

[8] `http://www.umbertostraccia.it/cs/software/FuzzyOWL/`

[9] `http://minerva.ual.es:8080/FSASPARQL`

[10] `https://jena.apache.org/documentation/query/index.html`

2 Motivation

Fuzzy sets and aggregators can be used to express fuzzy SPARQL queries as follows. Firstly, Boolean connectives can be replaced by *Product*, *Łukasiewicz* and *Gödel* conjunctions and disjunctions. Secondly, fuzzy (binary) aggregators (i.e., *MEAN*, *WSUM* (Weighted sum), *WMIN* (Weighted minimum) and *WMAX* (Weighted maximum)) can be used to express user preferences. Finally, fuzzy operators (i.e., *VERY*, *MORE_OR_LESS*, *CLOSE_TO*, *AT_MOST* and *AT_LEAST*) can be used to play with fuzzy values. For instance, the following FSA-SPARQL query:

```
SELECT ?Name ?Rank
WHERE {
?Movie movie:name ?Name .
?Movie movie:leading_role (?Actor ?l) .
?Actor movie:name "George Clooney".
{?Movie f:type (movie:Thriller ?c)} UNION
{?Movie f:type (movie:Comedy ?c)} .
?Movie f:type (movie:Good ?r) .
BIND(f:OR_GOD(f:AND_LUK(?r,?l),?c) as ?Rank)
}
ORDERBY DESC(?Rank)
```

expresses: "Retrieve good thriller/comedy movies in which *George Clooney* is the leading role". Here, we can see the main differences between crisp SPARQL and FSA-SPARQL. Firstly, triple patterns can have an associated truth degree. For instance, *?Movie movie:leading_role (?Actor ?l)*, is a triple pattern in which *?l* represents the truth degree associated to the object property *leading_role*. It means that the degree in which *?Actor* is the *leading_role* of the movie *?Movie* is *?l*. The same thing happens with the triple patterns *?Movie f:type (movie:Thriller,?c)* and *?Movie f:type (movie:Good,?r)* but here, *?c* and *?r* represent the degree in which the *?Movie* is a *"Thriller"*, and *"Good"*, respectively. Now, fuzzy connectives are used to combine fuzzy values. The Łukasiewicz conjunction (*and*) is used to combine *?r* and *?l*, while the *Gödel* disjunction (*or*) is used to combine them with *?c*.

The following example shows how to give preference to certain answers using several fuzzy operators and aggregators.

```
SELECT ?Name ?p ?d
WHERE {
?Hotel hotel:name ?Name .
?Hotel rdf:type hotel:Hotel .
?Hotel hotel:price ?p .
?Hotel f:type (hotel:Good ?g) .
?Hotel f:type (hotel:Elegant ?e) .
BIND(f:WSUM(0.1,f:MEAN(f:MORE_OR_LESS(?e),
f:VERY(?g)),0.9,f:CLOSE_TO(?p,100,50)) as ?d)
}
```

Here, we plan to retrieve hotels being very good and more or less elegant, with a price around 100 dollars, but giving much more relevance to the price. To express preferences the fuzzy aggregator *WSUM* is used. In this case *0.1* to elegance and quality versus *0.9* to price. Additionally, *VERY* is used to qualify the degree associated to *"Good"*, while *MORE_OR_LESS* is used to qualify the degree associated to *"Elegant"*. The aggregator *MEAN* is used to combine both degrees. Finally, a fuzzy operator *CLOSE_TO* is used to approximate prices to 100 dollars. Let us remark that neither f-SPARQL nor FSAQL are able to express the previous two queries.

3 Fuzzy RDF for FSA-SPARQL

As commented in the introduction section, our fuzzy version of RDF is able to assign truth degrees to concept membership, role fulfilling, subconcept and subrole relationships. This is the same case as fuzzy OWL 2 [10], but in our case the truth degree is a value of the [0,1] interval, while in fuzzy OWL 2 the truth degrees can be assigned by a function.

With this small RDF extension we are able to express vague and imprecise knowledge about membership relationships. We can define a set of fuzzy concepts, for instance, *Excellent*, *Good*, *Average* and *Low*, and we can state the membership of a certain individual to this set of concepts. Each membership relationship has an associated truth degree. For instance, *Ocean's Eleven* movie can be stated as member of *Excellent* with a degree *0.4*, and the same for *Good*, *Average* and *Low*. The set of fuzzy concepts are RDF concepts (i.e., classes) and they can be defined using *Protégé* tool. The question now is how to assign *Ocean's Eleven* to each concept and how to state truth degrees. With this aim, a fuzzy vocabulary (whose namespace is **f**) has been defined including an object property **f:type** and a data property called **f:truth**. Additionally, an (artificial) individual is created for each pair (concept, truth degree). In the example, an (artificial) individual *i* is created for (*Excellent*, *0.4*). Now, the following triples are declared in order to assign *Ocean's Eleven* to (*Excellent*, *0.4*): (*Ocean's Eleven f:type i*), (*i rdf:type Excellent*) and (*i f:truth 0.4*). In order to assign another concept to the same movie, a new individual *i'* has to be considered and the same procedure must be followed: (*Ocean's Eleven f:type i'*), (*i' rdf:type Good*) and (*i' f:truth 0.5*).

In the case of role fulfilling, let us suppose that *leading_role* is a fuzzy role and we are interested in stating different truth degrees to each actor of a movie. In this case, a new element of the fuzzy vocabulary is introduced called **f:item**. Let us suppose the movie *Ocean's Eleven* where *Brad Pitt* and *George Clooney* can be considered leading roles in a degree of 0.5. In this case, two (artificial) individuals *b* and *g* are used in the following triples: (*Ocean's Eleven leading_role g*), (*Ocean's Eleven leading_role b*), (*g f:item George Clooney*), (*g f:truth 0.5*), (*b f:item Brad Pitt*) and (*b f:truth 0.5*).

In the case of *rdfs:subClassOf* and *rdfs:subPropertyOf* instead of artificial individuals, artificial concepts/individuals and roles/individuals are created. For instance, if *cast member* is a subrole of *friend* with a degree of 0.4, then (*cast member f:subPropertyOf p*), (*p rdfs:subPropertyOf friend*) and (*p f:truth* 0.4) are created, where *p* is the artificial role/individual.[11] Thus the fuzzy vocabulary also includes **f:subClassOf** and **f:subPropertyOf**.

This RDF based representation is fully compatible with *Protégé*, and it can be edited. Additionally, FSA-SPARQL queries can be easily translated into crisp SPARQL assuming this representation of fuzzy data. The translation will be defined in Section 4.

4 FSA-SPARQL Query Language

FSA-SPARQL language is an extension of crisp SPARQL in which triple patterns *subject-property-object*: *s p (o t)* are allowed. Thus, objects in triple patterns can be a pair[12] of elements. *o* represents the original object and *t* the truth degree. Crisp SPARQL is a sublanguage of FSA-SPARQL and thus triple patterns can still be *s p o*. Not all the predicates have to be fuzzy. For instance, in the previous examples *name* is not a fuzzy predicate. Additionally, the set of predicates is extended with fuzzy

[11] Let us remark that *p* here plays the role of both property and individual.

[12] A pair can be handled in standard SPARQL by using RDF lists.

$\text{AND\_PROD}(x,y) = x*y$	$\text{OR\_PROD}(x,y) = x+y-x*y$
$\text{AND\_GOD}(x,y) = \min(x,y)$	$\text{OR\_GOD}(x,y) = \max(x,y)$
$\text{AND\_LUK}(x,y) = \max(x+y-1,0)$	$\text{OR\_LUK}(x,y) = \min(x+y,1)$
$\text{MEAN}(x,y) = \frac{x+y}{2}$	$\text{WSUM}(w,x,u,y) = w*x+u*y$
$\text{WMAX}(w,x,u,y) = \max(\min(w,x),\min(u,y))$	$\text{WMIN}(w,x,u,y) = \min(\max(1-w,x),\max(1-u,y))$
$\text{VERY}(x) = x^2$	$\text{MORE\_OR\_LESS}(x) = \sqrt{x}$
$\text{CLOSE\_TO}(x,l,\alpha) = \frac{1}{1+(\frac{x-l}{\alpha})^2}$ $\text{AT\_LEAST}(x,l,\alpha) = \begin{cases} 0 & \text{if } x \le \alpha \\ \frac{x-\alpha}{l-\alpha} & \text{if } \alpha < x < l \\ 1 & \text{if } x \ge l \end{cases}$	$\text{AT\_MOST}(x,l,\alpha) = \begin{cases} 1 & \text{if } x \le l \\ \frac{\alpha-x}{\alpha-l} & \text{if } l < x < \alpha \\ 0 & \text{if } x \ge \alpha \end{cases}$

Figure 1: FSA-SPARQL connectives and operators

versions of RDF/RDFS elements *f:type*, *f:subClassOf* and *f:subPropertyOf*. The set of fuzzy connectives and operators in FSA-SPARQL is defined in Figure 1.

4.1 Examples

Now, we would like to show some examples of FSA-SPARQL in order to explain how the user can express preferences by using different fuzzy connectives and operators.

Example 1 In this example, we will be mainly concerned with the combination of different conjunctive and disjunctive connectives coming from quite standard fuzzy logics. Adjectives like *pessimistic*, *realistic* and *optimist* are sometimes applied to the *Łukasiewicz*, *Product* and *Gödel* fuzzy logics, respectively, since conjunctions satisfy that, for any pair of real numbers x and y in $[0,1]$, we have:

$$0 \le \text{AND\_LUK}(x,y) \le \text{AND\_PROD}(x,y) \le \text{AND\_GOD}(x,y) \le 1$$

In contrast, the contrary holds for the disjunction operations:

$$0 \le \text{OR\_GOD}(x,y) \le \text{OR\_PROD}(x,y) \le \text{OR\_LUK}(x,y) \le 1$$

Note that it is more difficult to satisfy a condition based on a pessimistic conjunction/disjunction (i.e, inspired by the *Łukasiewicz* and *Gödel* fuzzy logics, respectively) than with *Product* logic based operators, while the optimistic versions of such connectives are less restrictive, obtaining greater truth degrees on answers. This is a consequence of the following chain of inequalities:

$$\begin{aligned} 0 \le \text{AND\_LUK}(x,y) \le \text{AND\_PROD}(x,y) \le \text{AND\_GOD}(x,y) \le \\ \le \text{OR\_GOD}(x,y) \le \text{OR\_PROD}(x,y) \le \text{OR\_LUK}(x,y) \le 1 \end{aligned}$$

The following query is quite similar to the first one seen in Section 2, since we are looking again for good thriller/comedy movies in which *George Clooney* be the leading role. Here, *?l* represents the degree in which *?Actor* is the *leading_role* of the movie *?Movie*, while *?c* and *?r* represent the degree in which the *?Movie* is a *"Thriller"*, and *"Good"*, respectively. Now, we can opt among several options if we plan to use a conjunction connective to combine *?r* and *?l*, following with a disjunctive operator to combine them with *?c*, and finally assigning the returned truth degree to variable *?Rank*. Initially, let us collect both connectives from the Product logic, which represent the intermediate case:

```
SELECT ?Name ?Rank
WHERE {
?Movie movie:name ?Name .
?Movie movie:leading_role (?Actor ?l) .
?Actor movie:name "George Clooney".
?Movie f:type (movie:Thriller ?c) .
?Movie f:type (movie:Good ?r) .
BIND(f:OR_PROD(f:AND_PROD(?r,?l),?c)
    as ?Rank) .
FILTER (?Rank > 0.7)
}
```

The condition "f:OR_PROD(f:AND_PROD(?r,?l),?c)" is satisfied by the following three films with their associated degrees:

The Descendants $\Rightarrow$ 0.70
Ocean's Eleven $\Rightarrow$ 0.760
The American $\Rightarrow$ 0.96

where George Clooney is the leading role with a degree of 1 in The Descendants and The American, and with 0.5 in Ocean's Eleven, The Descendants is a thriller in a degree of 0.4, and is a comedy in a degree of 0.4, Ocean's Eleven is a thriller in a degree of 0.7, and is a comedy in a degree of 0.9, and finally, The American is a thriller in a degree of 0.9, and a comedy in a degree of 0.1.

The same sequence of movies is reported by the pessimistic version of the condition, that is "f:OR-_GOD (f:AND_LUK (?r,?l),?c)", but reducing the truth degrees to 0.5, 0.7 and 0.9, respectively, while the optimistic version "f:OR_LUK(f:AND_GOD(?r,?l),?c)" would assign higher degrees to the films, i.e., 0.9, 1 and 1, respectively.

Finally, by simply adding the command "FILTER (?Rank > 0.7)" at the end of such queries, the system would simply generate one answer in the pessimistic case (The American $\Rightarrow$ 0.9), two results in the intermediate case (Ocean's Eleven $\Rightarrow$ 0.760 and The American $\Rightarrow$ 0.96) and the three films in the optimistic case (The Descendants $\Rightarrow$ 0.9, Ocean's Eleven $\Rightarrow$ 1, and The American $\Rightarrow$ 1).

Example 2. Consider now the following query also illustrated in Section 2:

```
SELECT ?Name ?p ?d
WHERE {
?Hotel hotel:name ?Name .
?Hotel rdf:type hotel:Hotel .
?Hotel hotel:price ?p .
?Hotel f:type (hotel:Good ?g) .
?Hotel f:type (hotel:Elegant ?e) .
BIND(f:WSUM(0.1,f:MEAN(f:MORE_OR_LESS(?e),
f:VERY(?g)),0.9,f:CLOSE_TO(?p,100,50)) as ?d)
```

Here, we work with variables containing crisp values like *?p*, which refers to the exact price of a given hotel, and other ones with a fuzzy taste (i.e., their values are real numbers in $[0,1]$) like *?g* and *?e* informing about the degree in which each hotel is good and elegant, respectively. Moreover, the following combination of such values is assigned to variable *?d* (which is reported preceded by the name and the price of the corresponding hotel on each answer):

• Firstly, we apply the also called *linguistic modifiers* "MORE_OR_LESS" and "VERY" to *?e* and *?g*, in order to respectively increase and decrease these values.

• The fuzzy operator "CLOSE_TO" is useful for modulating the degree in which a given price *?p* belongs to a fuzzy set whose membership function assigns the maximum value (1) to prices non greater than 100 dollars, and next, such degree linearly decreases until 0 for prices between 100 and 150 dollars.

• Finally, we use a couple of fuzzy aggregators for expressing two versions of the notion of *average*: the classical one (denoted by "MEAN") and a weighted variant (represented by "WSUM"). The high flexibility of this last version is very useful for expressing preferences since, in our particular example, we establish that the price of the hotel is 9 times more important than the concrete combination of *?e* and *?g*.

Let us remark once again that neither f-SPARQL nor FSAQL have enough expressive power to qualify ("VERY","MORE_OR_LESS"), modulate ("CLOSE_TO") and establish preferences ("WSUM") as done in this query.

4.2 FSA-SPARQL to Crisp SPARQL

Now, we will present the transformation of FSA-SPARQL to crisp SPARQL. This transformation makes possible to run FSA-SPARQL queries in any crisp SPARQL implementation. Also, it provides a semantics to the language. The transformation consists of two parts. Firstly, "fuzzy" triple patterns have to be transformed into a set of "crisp" triple patterns. The transformation is as follows.

(1) (s p (o t))	$\rightarrow$ *(s p ?x) (?x f:item o)*
	(?x f:truth t)
(2) (s f:type (o t))	$\rightarrow$ *(s f:type ?x)*
	(?x rdf:type o) (?x f:truth t)
(3) (s f:subClassOf (o t))	$\rightarrow$ *(s f:subClassOf ?x)*
	(?x rdfs:subClassOf o)
	(?x f:truth t)
(4) (s f:subPropertyOf (o t))	$\rightarrow$ *(s f:subPropertyOf ?x)*
	(?x rdfs:subPropertyOf o)
	(?x f:truth t)

Secondly, each fuzzy operator and connective has to be translated into the corresponding SPARQL formula using SPARQL numeric operators (i.e., *,+,-,/, *sqrt*, *pow*) as well as the *if* operator. Unfortunately, SPARQL is not equipped with max and min operators, and thus $\max(x,y)$ must be replaced by $if(x >= y,x,y)$ and $\min(x,y)$ by $if(x > y,y,x)$.
For instance, the query:

```
SELECT ?Name ?Rank
WHERE {
?Movie movie:name ?Name .
?Movie movie:leading_role (?Actor ?l) .
?Actor movie:name "George Clooney".
?Movie f:type (movie:Thriller ?c) .
?Movie f:type (movie:Good ?r) .
BIND(f:OR_GOD(f:AND_LUK(?r,?l),?c) as ?Rank)
}
ORDERBY DESC(?Rank)
```

is translated into crisp SPARQL as follows:

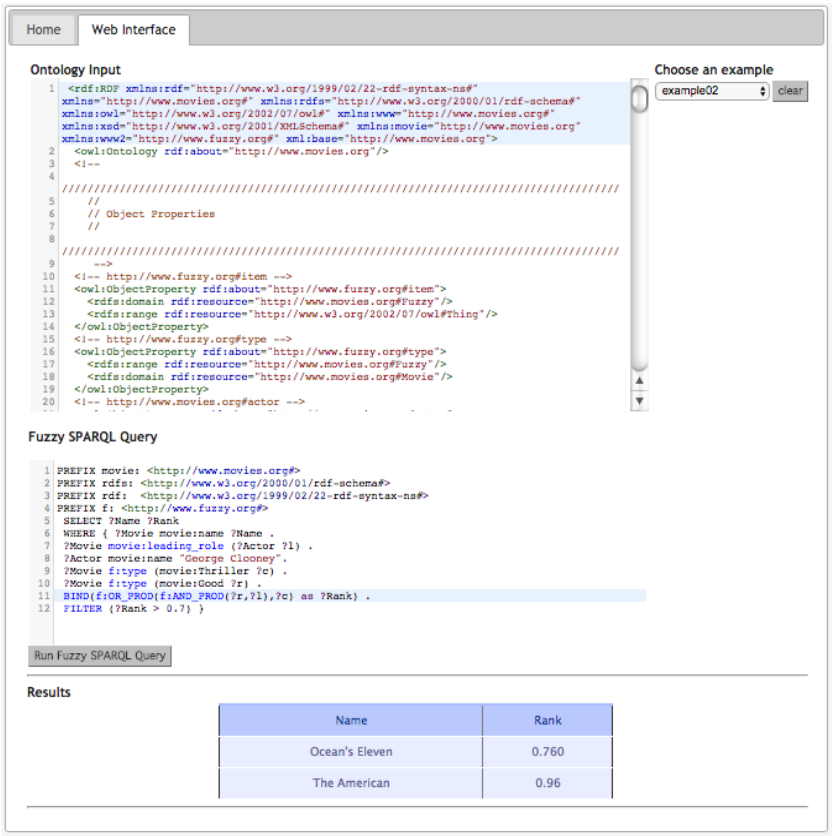

Figure 2: Web Tool for FSA-SPARQL

```
SELECT ?Name ?Rank
WHERE {
?Movie movie:name ?Name .
?Movie movie:leading_role ?x .
?x f:item ?Actor . ?x f:truth ?l .
?Actor movie:name "George Clooney".
?Movie f:type ? y . ?y rdf:type movie:Thriller .
?y f:truth ?c .?Movie f:type ?z .
?z rdf:type movie:Good . ?z f:truth ?r .
BIND(if(if(?r>=?l,?r,?l)>=?c,
  if(?r>=?l,?r,?l),?c) as ?Rank)
}
ORDERBY DESC(?Rank)
```

We have developed a Web tool: `http://minerva.ual.es:8080/FSASPARQL` from which FSA-SPARQL queries can be executed on-line. The tool includes examples to be run (including the ones shown in the paper). An ontology is loaded for each example and users can manually modify the ontology as well as the query. The ontology can be also edited with the Protégé tool and copied/pasted into the ontology input text area. A capture of the Web tool is shown in Figure 2.

5 Conclusions and Future Work

In this paper we have presented an extension of SPARQL to query fuzzy data. The extension is able to express fuzzy queries making use of fuzzy connectives and (aggregation) operators. Fuzzy data are modeled by fuzzy RDF axioms. We have provided semantics to the extension by translating the fuzzy version into the crisp one. A Web tool has been developed for enabling the on-line execution of queries. As future work we plan the following research lines. Firstly, we would like to work on mechanisms for debugging fuzzy data. While Protégé can be used for editing fuzzy data, we have that checking completeness (for instance, truth values should be assigned to each fuzzy axiom) and consistency (for instance, the sum of truth values should be equal to one) of data is not still possible. Secondly, we also plan to automatically generate an ontology with truth degrees from Linked Open Data resources. Unfortunately, fuzzy RDF data are not easy to find on the Web, and thus some mechanism should be established to fuzzify RDF data. Finally, the type of axioms handled by FSA-SPARQL can be extended allowing OWL axioms (equivalence, disjointness, same as, different from, etc.,).

Acknowledgements

This work has been partially supported by FEDER and the State Research Agency (AEI) of the Spanish Ministry of Economy and Competition under grants TIN2013-44742-C4-4-R and TIN2016-76843-C4-2-R (AEI/FEDER, UE).

References

[1] Jesús Alcalá-Fdez & Jose M Alonso (2016): *A Survey of Fuzzy Systems Software: Taxonomy, Current Research Trends, and Prospects*. IEEE Transactions on Fuzzy Systems 24(1), pp. 40–56.

[2] Jesús M Almendros-Jiménez, Alejandro Luna-Tedesqui & Ginés Moreno (2014): *Fuzzy xpath queries in XQuery*. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, pp. 457–472.

[3] Jesús M Almendros-Jiménez, Alejandro Luna-Tedesqui & Ginés Moreno (2015): *Fuzzy XPath through fuzzy logic programming*. New Generation Computing 33(2), pp. 173–209.

[4] Jesús M Almendros-Jiménez, Alejandro Luna-Tedesqui & Ginés Moreno (2015): *Thresholded debugging of XPath queries*. In: Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on, IEEE, pp. 1–9.

[5] Jesús M Almendros-Jiménez, Alejandro Luna-Tedesqui & Ginés Moreno (2016): *Debugging while interpreting fuzzy XPath queries*. In: Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on, IEEE, pp. 233–240.

[6] Franz Baader (2003): *The description logic handbook: Theory, implementation and applications*. Cambridge university press.

[7] Afef Bahri, Rafik Bouaziz & Fäiez Gargouri (2010): *Querying fuzzy RDFS semantic annotations*. In: Fuzzy Systems (FUZZ-IEEE), 2010 IEEE International Conference on, IEEE, pp. 1–8.

[8] Fernando Bobillo, Miguel Delgado & Juan Gómez-Romero (2013): *Reasoning in fuzzy OWL 2 with DeLorean*. In: Uncertainty Reasoning for the Semantic Web II, Springer, pp. 119–138.

[9] Fernando Bobillo & Umberto Straccia (2008): *fuzzyDL: An expressive fuzzy description logic reasoner*. In: Fuzzy Systems (FUZZ-IEEE), 2008 IEEE International Conference on, pp. 923–930.

[10] Fernando Bobillo & Umberto Straccia (2011): *Fuzzy ontology representation using OWL 2*. International Journal of Approximate Reasoning 52(7), pp. 1073–1094.

[11] Fernando Bobillo & Umberto Straccia (2013): *Aggregation operators for fuzzy ontologies*. Applied Soft Computing 13(9), pp. 3816–3830.

[12] Jingwei Cheng, ZM Ma & Li Yan (2010): *f-SPARQL: a flexible extension of SPARQL*. In: International Conference on Database and Expert Systems Applications, Springer, pp. 487–494.

[13] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider & Ulrike Sattler (2008): *OWL 2: The next step for OWL*. Web Semantics: Science, Services and Agents on the World Wide Web 6(4), pp. 309–322.

[14] Marina Gueroussova, Axel Polleres & Sheila McIlraith (2013): *SPARQL with qualitative and quantitative preferences*. In: Proceedings of the 2nd International Conference on Ordering and Reasoning-Volume 1059, CEUR-WS. org, pp. 2–8.

[15] Holger Knublauch, Ray W Fergerson, Natalya F Noy & Mark A Musen (2004): *The Protégé OWL plugin: An open development environment for semantic web applications*. In: International Semantic Web Conference, Springer, pp. 229–243.

[16] Ruizhe Ma, Xiangyue Jia, Jingwei Cheng & Rafal A Angryk (2015): *SPARQL queries on RDF with fuzzy constraints and preferences*. Journal of Intelligent & Fuzzy Systems 30(1), pp. 183–195.

[17] ZM Ma, Fu Zhang, Hailong Wang & Li Yan (2013): *An overview of fuzzy description logics for the semantic web*. The Knowledge Engineering Review 28(01), pp. 1–34.

[18] Sara Magliacane, Alessandro Bozzon & Emanuele Della Valle (2012): *Efficient execution of top-k SPARQL queries*. In: International Semantic Web Conference, Springer, pp. 344–360.

[19] Víctor Pablos-Ceruelo & Susana Munoz-Hernandez (2014): *FleSe: A Tool for Posing Flexible and Expressive (Fuzzy) Queries to a Regular Database*. In Sigeru Omatu, Hugues Bersini, Juan M. Corchado, Sara Rodríguez, Paweł Pawlewski & Edgardo Bucciarelli, editors: Distributed Computing and Artificial Intelligence, 11th International Conference, Springer International Publishing, pp. 157–164. Available at `http://dx.doi.org/10.1007/978-3-319-07593-8_20`.

[20] Jorge Pérez, Marcelo Arenas & Claudio Gutierrez (2009): *Semantics and complexity of SPARQL*. ACM Transactions on Database Systems (TODS) 34(3), p. 16.

[21] Olivier Pivert, Olfa Slama & Virginie Thion (2016): *SPARQL extensions with preferences: a survey*. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, pp. 1015–1020.

[22] Bastian Quilitz & Ulf Leser (2008): *Querying distributed RDF data sources with SPARQL*. In: European Semantic Web Conference, Springer, pp. 524–538.

[23] Nikolaos Simou, Giorgos Stoilos & Giorgos Stamou (2013): *Storing and querying fuzzy knowledge in the semantic web using fire*. In: Uncertainty Reasoning for the Semantic Web II, Springer, pp. 158–176.

[24] Giorgos Stoilos, Giorgos Stamou & Jeff Z Pan (2010): *Fuzzy extensions of OWL: Logical properties and reduction to fuzzy description logics*. International Journal of Approximate Reasoning 51(6), pp. 656–679.

[25] Umberto Straccia (2013): *Foundations of Fuzzy Logic and Semantic Web Languages*. CRC Press.

[26] Hairong Wang, ZM Ma & Jingwei Cheng (2012): *fp-SPARQL: an RDF fuzzy retrieval mechanism supporting user preference*. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on, IEEE, pp. 443–447.