N. Manolis and Y. Tzitzikas,
Interactive Exploration of Fuzzy RDF Knowledge Bases,
8th Extended Semantic Web Conference, ESWC 2011,
Heraklion, Greece, June 2011

Interactive Exploration of Fuzzy RDF Knowledge Bases

Nikos Manolis and Yannis Tzitzikas

Institute of Computer Science, FORTH-ICS, GREECE, and
Computer Science Department, University of Crete, GREECE
{manolisn|tzitzik}@ics.forth.gr

Abstract. In several domains we have objects whose descriptions are accompanied by a degree expressing their strength. Such degrees can have various application specific semantics, such as relevance, precision, certainty, trust, etc. In this paper we consider *Fuzzy RDF* as the representation framework for such "weighted" descriptions and associations, and we propose a novel model for *browsing and exploring* such sources, which allows formulating complex queries gradually and through plain clicks. Specifically, and in order to exploit the fuzzy degrees, the model proposes interval-based transition markers. The advantage of the model is that it significantly increases the discrimination power of the interaction, without making it complex for the end user.

1 Introduction

There are many practical situations where the descriptions of objects are accompanied by a degree. These degrees can be provided by humans or be the result of automated tasks. For instance, in [19] they express the degree of a feature that is extracted from data, in [7] they express the certainty of membership of a document to an aspect (automatically retrieved from the answer set of a keyword query), in [17] the membership of an object to a cluster, in [8] the evaluation scores of products under hierarchically organized criteria, in [13] they express the frequency of symptoms in a disease. Furthermore in an open environment like the Web, we may have data of various degrees of credibility, as well data which are copies or modifications of other data. As a result, data of the same entity can be erroneous, out-of-date, or inconsistent/conflicting in different data sources. Therefore, even if the primary data are not fuzzy, the integrated data as produced by an information integration system (that contains tasks like web extraction) can be fuzzy. Synopsizing, such degrees can capture various application specific semantics, such as *relevance*, *precision*, *certainty*, *trust*, etc.

Users would like to browse and explore such sources without having to be aware of the terminology, contents or query language of the source. Furthermore, the fuzzy degrees should be exploitable, allowing the users to reach states which are characterized by conditions that involve degrees. Finally, it should be possible to offer adequate support for recall-oriented information needs, and support a gradual focus restriction process (i.e. interactive search). However there is not any exploration/browsing approach for such sources.

The contribution of this paper lies in introducing a model for exploring such sources. The merit of the proposed model is that it defines formally and precisely the state space of an interaction that (a) allows users to locate the objects of interest, or to get overviews, without having to be aware of the terminology nor the query language of the underlying source, and without reaching states with empty results, (b) exploits fuzzy degrees for enhancing the discrimination power of the interaction, (c) generalizes the main exploration/browsing approaches for plain RDF/S sources (also clarifying issues regarding schema and instance cyclic property paths), (d) is query language independent, and (e) is visualization independent. Finally it discusses issues that concern the available query languages.

To grasp the idea, Fig. 1 shows an example a Fuzzy RDF KB where instance triples have fuzzy degrees. Properties are depicted by rectangles and the letters "d" and "r" denote the domain and the range of a property. Fat arrows denote subClassOf/subPropertyOf relationships, while dashed arrows denote instanceOf relationships. Note that various approaches have been proposed recently for annotating triples with a degree of truth and giving a semantics to such annotations, e.g. [16, 18], and can be used as the representation framework. Among the various possible approaches for exploiting fuzzy degrees at the interaction level, we propose one based on *intervals*, leading to a simple and intuitive dialog. The idea is to analyze the count information of each transition marker to more than one counts each corresponding to the objects whose membership degrees fall into a specified interval. An instance of the proposed interaction is sketched at Fig. 2. The figure depicts only the part of the UI (usually the left bar) that shows the transition markers (it does not show the object set). Fig. 2(a) ignores fuzzy degrees. Fig. 2(b) shows how fuzzy degrees are exploited. For example, consider a transition marker "z(20)". We can present it as "z [low(10), medium(4), high(6)]" where "low" may correspond to degrees (0,0.3], "medium" to (0.3,0.6] and "high" to (0.6,1]. So the user has now three clicks (i.e. three transitions) instead of one. Therefore the set of all possible foci is more, so the discrimination power of interaction increases. The increased discrimination power is useful especially in cases where the fuzzy degrees are derived by automatic methods. Since the results of such methods are vulnerable to errors and inaccuracies (i.e. the derived degrees may be lower or higher than those deserved), it would not be wise to exclude the descriptions having low degrees. Instead, it is better to make all of them available and let the user free to explore also the objects having low degrees if necessary. In brief the interaction leads to states whose extension corresponds to the answer of complex path expressions that involve interval-based conditions over the fuzzy degrees, e.g. queries of the form: *" Japanese cars for sale which are driven by persons who work at FORTH and know a person who knows Bob",* where each individual underlined part (value-based condition) is associated also with an interval-based condition over the fuzzy degrees.

The paper is organized as follows. Section 2 describes in brief related works. Section 3 introduces the least number of symbols and notations required for defining the interaction model. Section 4 introduces the model first for plain

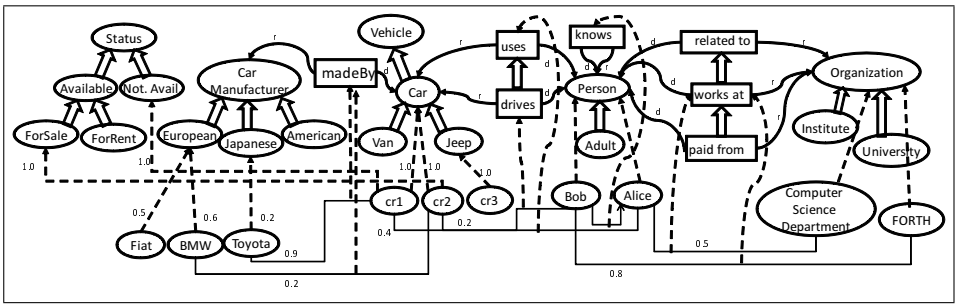


Fig. 1. A Fuzzy RDF KB

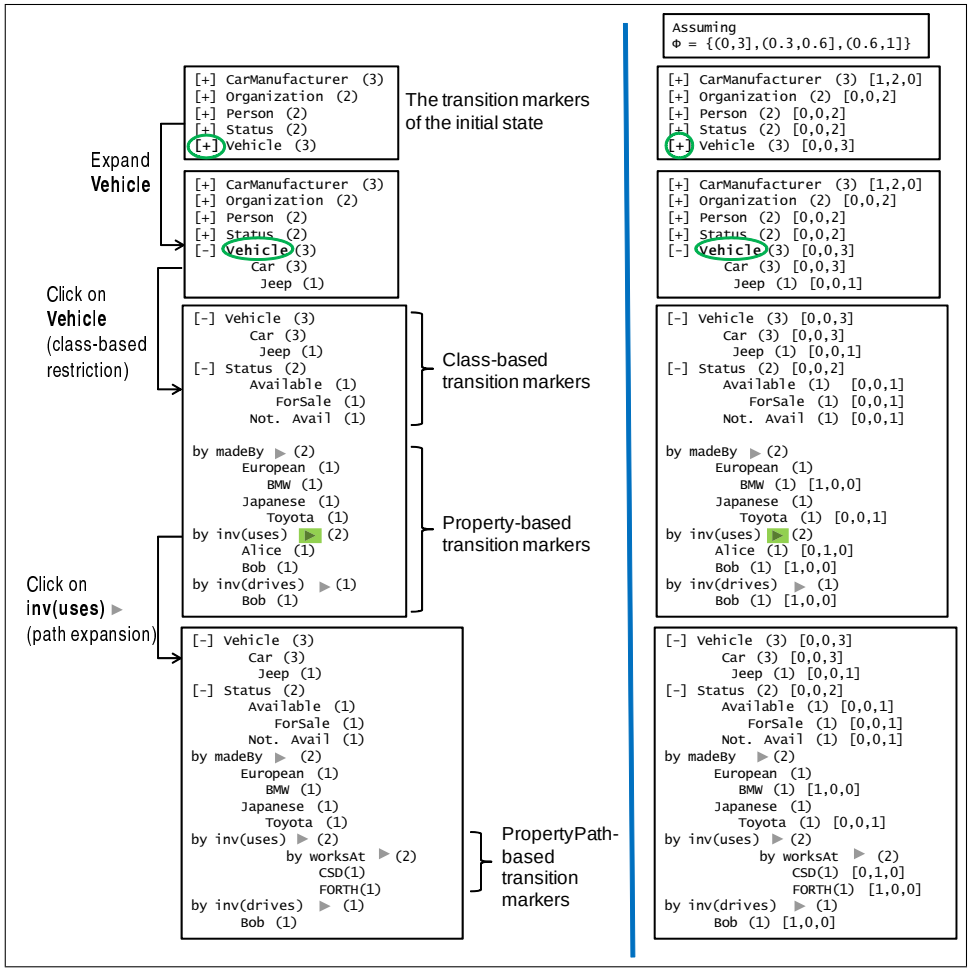


Fig. 2. Sketch of the GUI part for transition markers
(a): ignores fuzzy degrees. (b): count information based on intervals over the fuzzy degrees

RDF/S and then extends it for Fuzzy RDF. Section 5 discusses query language issues, and finally Section 6 concludes the paper.

2 Related Work

Some browsing approaches are applicable to simple structures (like attribute-value pairs), while others to complex information structures (e.g. OWL-based KBs). Therefore one important aspect is how the underlying information is structured. There are several options, some of them follow: attribute-value pairs with flat values (e.g. `name=Yannis`), attribute-value pairs with hierarchically organized values (e.g. `location=Crete`), set-valued attributes (either flat or hierarchical) (e.g. `accessories={ABS, ESP}`), multi-entity (or object-oriented) (e.g. RDF, linked open data), and relational databases. Furthermore, we could have fuzziness and we can consider this as an independent aspect (e.g. there are fuzzy extensions of the RDF model such as [10, 16]).

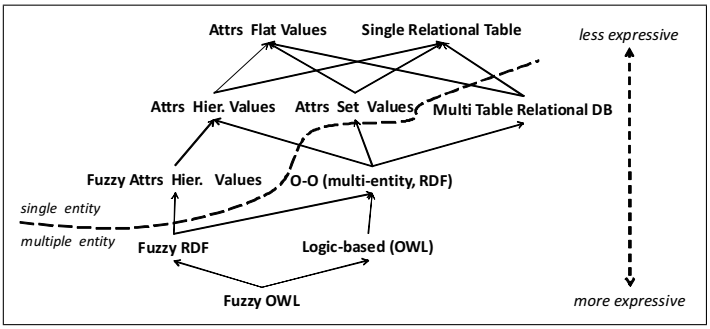


Fig. 3. Categories of Information Spaces

Fig. 3 shows the above categories organized hierarchically where an option X is a (direct or indirect) child of an option Y if whatever information can be expressed in Y can also be expressed in X. The value of this diagram is that it depicts the fact that if a browsing approach is appropriate for an option X, then certainly it is appropriate for all options which are parents of X. For instance, a browsing approach appropriate for Fuzzy RDF is also appropriate for plain RDF, as well as for sources formed by attributes with fuzzy and hierarchically organized values. Just indicatively, the following table lists some browsing approaches grouped according to their applicability. In this paper we focus on Fuzzy RDF and we are interested in generic approaches (not requiring special configuration), that exploit schema information (if available). In contrast, [6] deals with fuzzy annotations of documents w.r.t. a particular ontology using a view-based approach where the browsable elements (views) are predefined by specialists.

Information Space	System
Attrs with flat values	Elastic Lists [14]
Attrs with hierarchical values	Flamenco [20], Mitos WSE [12]
Object-Oriented (e.g. RDF)	GRQL [2], BrowseRdf[11], Ontogator [9], VisiNav [4],
FRDF (Fuzzy RDF)	Fuzzy view-based Semantic Search [6]
OWL	Odalisque [1]

3 Fuzzy RDF

Consider a set of RDF triples K and let $\mathcal{C}(K)$ be its closure. We shall denote with C the set of classes, with Pr the set of properties, with $\leq_{cl}$ the subclassOf relation between classes, and with $\leq_{pr}$ the subpropertyOf relation between properties. The instances of a class $c \in C$ are $inst(c) = \{o \mid (o, type, c) \in \mathcal{C}(K)\}$, while the instances of a property $p \in Pr$ are $inst(p) = \{\ (o, p, o') \mid (o, p, o') \in \mathcal{C}(K)\}$. Now we introduce some notations for Fuzzy RDF. Each instance triple t (either class or property instance triple) is accompanied by a fuzzy degree, which we shall denote by $directdegree(t)$. We can now define the degree of t, denoted by $degree(t)$, based on the semantics of RDF, and the axioms of Fuzzy Set Theory[1]. Specifically,

$degree(o, type, c) = \max\{\ directdegree(o, type, c') \mid c' \leq_{cl} c\}$

$degree(o, p, o') = \max\{\ directdegree(o, p', o') \mid p' \leq_{pr} p\}$.

Let $\Phi = \{\varphi_1, \ldots \varphi_m\}$ be a set of intervals in [0,1]. We define:

$inst(c, \varphi) = \{\ o \in inst(c) \mid degree(o, type, c) \in \varphi\}$

$inst(p, \varphi) = \{\ (o, p, o') \in inst(p) \mid degree(o, p, o') \in \varphi\}$.

4 The Interaction Model

The interaction is modeled by a *state space*. Each *state* has an *extension* and a number of *transitions* leading to other states. Each transition is signified by a *transition marker* accompanied by a number showing the size of the extension of the targeting state (we will refer to this with *count* information). Such view abstracts from the various visualization approaches; in general each state has one or more visualization modes for its extension as well as its transition markers.

4.1 The Interaction Model for Plain RDF/S

The objective is to define a precise and concise model capturing the essentials of RDF browsing approaches, which later will be extended to capture Fuzzy RDF.

Consider that we are in the context of one RDF/S KB with a single namespace with classes C and properties Pr. If s denotes a state we shall use $s.e$ to denote its extension. Let's start from the *initial state(s)*. Let s_0 denote an artificial initial state. We can assume that $s_0.e = URI \cup LIT$, i.e. its extension contains every URI and literal of the KB. Alternatively, the extension of the initial state can be the result of a keyword query, or a set of resources provided by an external access method. Given a state we shall show how to compute the transitions that are available to that state. From s_0 the user can move to states corresponding to the maximal classes and properties, i.e. to one state for each $maximal_{\leq_{cl}}(C)$ and each $maximal_{\leq_{pr}}(Pr)$. Specifically, each $c \in maximal_{\leq_{cl}}(C)$ (resp. $p \in maximal_{\leq_{pr}}(Pr)$) yields a state with extension $inst(c)$ (resp. $inst(p)$).

We will define formally the transitions based on the notion of *restriction* and *join*. To this end we introduce some auxiliary definitions. We shall use p^{-1} to denote the inverse direction of a property p, e.g. if $(d, p, r) \in Pr$ then

[1] We will adopt Zadeh's theory and consequently we shall use min/max, however one could also adopt alternative definitions for the operators $\bigotimes$, $\bigoplus$ (e.g. as done in [16]).

$p^{-1} = (r, inv(p), d)$, and let Pr^{-1} denote the inverse properties of all properties in Pr. If E is a set of resources, p is a property in Pr or Pr^{-1}, v is a resource or literal, $vset$ is a set of resources or literals, and c is a class, we define the following notations for *restricting* the set E:

$$Restrict(E, p : v) = \{\ e \in E \mid (e, p, v) \in inst(p)\}$$
$$Restrict(E, p : vset) = \{\ e \in E \mid \exists\ v' \in vset \text{ and } (e, p, v') \in inst(p)\}$$
$$Restrict(E, c) = \{\ e \in E \mid e \in inst(c)\}$$

Now we define a notation for *joining* values, i.e. for computing values which are linked with the elements of E:

$$Joins(E, p) = \{\ v \mid \exists\ e \in E \text{ and } (e, p, v) \in inst(p)\}$$

We can now define precisely transitions and transition markers. Suppose we are in a state s with extension $s.e$.

Class-based browsing. The classes that can be used as class-based transition markers, denoted by $TM_{cl}(E)$, are defined by:

$$TM_{cl}(E) = \{c \in C \mid Restrict(E, c) \neq \emptyset\} \quad (1)$$

If the user clicks on a $c \in TM_{cl}(s.e)$, then the extension of the targeting state s' is defined as $s'.e = Restrict(s.e, c)$, and its count information is $s'.count = |s'.e|$. For example, suppose the user selects the class `Vehicle`. The user can then view its instances and follow one of the following class-based transition markers: `Vehicle, Car, Jeep, Status, Available, ForSale, Not.Available`. Notice that `ForRent` and `Van` are not included because their extension (and thus their intersection with the current extension) is empty.

The elements of $TM_{cl}(s.e)$ can be hierarchically organized (based on the subclass relationships among them). Specifically the layout (e.g. the indentation in a text-based layout) of the transition markers can be based on the relationships of the reflexive and transitive reduction of the restriction of $\leq_{cl}$ on $TM_{cl}(s.e)$ (i.e. on $R^{refl,trans}(\leq_{cl\ |\ TM_{cl}(s.e)})$). In our case, we can get what is shown in Fig. 4(a). Furthermore based on the relationship between the extensions $s.e$ and $s'.e$ a transition (or transition marker) can be characterized as a zoom-in/out/side transition.

Let's now focus on ***property-based browsing***. Suppose the user has focused on `Car`, and the extension of this state is {`cr1, cr2, cr3`}. He can further restrict the extension also through the properties. Roughly each property whose domain or range is the class `Car`, or a superclass of `Car` (in general any property that used in the resources in $s.e$), can be considered as a facet of the instances of `Car`. For example, consider a property `madeBy` whose domain is the class `Car` and suppose its range was the `String Literal` class. In that case the firm names of the current extension can be used as transition markers. Now suppose that the range of the property `madeBy` is not literal, but the class `CarManufacturer` (as shown in the figure). In this case, the firms (URIs in this case) of the current extension can again be used as transition markers, as shown in Fig. 4(b). Notice that `Fiat` is not shown as it is not related to the current focus (i.e. to `cr1`, `cr2`

and `cr3`)[2]. Formally, the properties (in their defined or inverse direction) that can be used for deriving transition markers are defined by:

$$Props(s) = \{p \in Pr \cup Pr^{-1} \mid Joins(s.e, p) \neq \emptyset\} \quad (2)$$

For each $p \in Props(s)$, the corresponding transition markers are defined by $Joins(s.e, p)$, and if the user clicks on a value v in $Joins(s.e, p)$, then the extension of the new state is $s'.e = Restrict(s.e, p : v)$.

(a)
```
Vehicle(3)
  Car(3)
    Jeep(1)
Status(2)
  Available(1)
    ForSale(1)
  Not. Avail(1)
```

(b)
```
by madeBy(2)
   BMW(1)
   Toyota(1)
```

(c)
```
by madeBy(2)
  European(1)
    BMW(1)
  Japanese(1)
    Toyota(1)
```

(d)
```
by inv(uses)(2)
   Alice(1)
   Bob(1)
by inv(drives)(1)
   Bob(1)
```

(e)
```
by inv(uses)(2)
  by worksAt(2)
     CSD(1)
     FORTH(1)
```

Fig. 4. Examples of transition markers

Furthermore, the transition markers of a property $p \in Props(s)$, i.e. the set $Joins(s.e, p)$, can be categorized based on their classes. In our example, the firms can be categorized through the subclasses of the class `CarManufacturer`. These classes can be shown as intermediate nodes of the hierarchy that lead to particular car firms, as shown in Fig. 4(c). These classes can be computed easily, they are actually given by $TM_{cl}(Joins(s.e, p))$. Furthermore, these values can be used as *complex transition markers*, i.e. as shortcuts allowing the user to select a set of values with disjunctive interpretation (e.g. he clicks on `Japanese` instead of clicking to every Japanese firm). Specifically, suppose the user clicks on such a value vc. The extension of the target state s' will be:

$$s'.e = Restrict(s.e, p : Restrict(Joins(s.e, p), vc)) \quad (3)$$

Returning to our example, and while the user has focused on cars, apart from `madeBy`, the user can follow transitions based on the properties `inv(drives)` and `inv(uses)`, as shown in Fig. 4(d). In addition, the elements of $Props(s)$ can be hierarchically organized based on the subProperty relationships among them.

We should be able to extend the above for ***property paths*** of length greater than one. This is needed for restricting the extension through the values of complex attributes (e.g. addresses that may be represented as blank nodes) or through the relationships (direct or indirect) with other resources. For example, one may want to restrict the set of cars so that only cars which are used by persons working for CSD (Computer Science Department) are shown. In that case we would like transition markers of the form shown in Fig. 4(e). It should also be possible the successive "application" of the same property. For example, the user may want to focus to all friends of the friends of `Bob`, or all friends of Bob at

[2] Since `cr3` does not participate to a `madeBy` property, an alternative approach is to add an artificial value, like `NonApplicable/Uknown`, whose count would be equal to 1, for informing the user that one element of his focus has not value wrt `madeBy`.

distance less than 5. Let's now define precisely, this *property path*-based browsing (expansion and cascading restriction). Let $p_1, \ldots, p_k$ be a sequence of properties. We call this sequence *successive in* s if $Joins(Joins(\ldots(Joins(s.e, p_1), p_2)\ldots p_k) \neq \emptyset$. Obviously such a sequence does not lead to empty results, and can be used to restrict the current focus. Let $M_1, \ldots M_k$ denote the corresponding set of transition markers at each point of the path. Assuming $M_0 = s.e$, the transition markers for all i such that $1 \leq i \leq k$, are defined as:

$$M_i = Joins(M_{i-1}, p_i) \tag{4}$$

What is left to show is how selections on such paths restrict the current focus. Suppose the user selects a value v_k from M_k. This will restrict the set of transitions markers in the following order $M_k, \ldots, M_1$ and finally it will restrict the extension of s. Let $M'_k, \ldots M'_1$ be the restricted set of transitions markers. They are defined as follows: $M'_k = \{v_k\}$, while for each $1 \leq i < k$ we have:

$$M'_i = Restrict(M_i, p_{i+1} : M'_{i+1}) \tag{5}$$

for instance, $M'_1 = Restrict(M_1, p_2 : M'_2)$. Finally, the extension of the new state s' is defined as $s'.e = Restrict(s.e, p_1 : M'_1)$. Equivalently, we can consider that M'_0 corresponds to $s'.e$ and in that case Eq. 5 holds also for $i = 0$.

For example, consider an ontology containing a path of the form: `Car--hasFirm-->Firm--ofCountry-->Country` and three cars `cr1, cr2, cr3`, the first being `BMW`, the second `VW`, the third `Renault`. The first two firms come from `Germany` the last from `France`. Suppose the user is on `Cars`, and expands the path `hasFirm.ofCountry`. If he selects `Germany`, then the previous list will become `BMW, VW` (so `Renault` will be excluded) and the original focus will be restricted to `cr1` and `cr2`. It follows that path clicks require disjunctive interpretation of the matched values in the intermediate steps.

The above can be applied also for successive applications of the same property, e.g. `inv(drives).knows`^{2}`.paidFrom` is a property path that can be used to restrict cars to those cars whose drivers know some persons who in turn know some persons who are paid from a particular organization.

Entity Type Switch. So far we have described methods to restrict the current extension. Apart from the current extension we can move to other objects. At the simplest case, from one specific resource we move to one resource which is directly or indirectly connected to that. Now suppose that the current focus is a *set* of resources (e.g. cars). Again we can move to one or more resources which are directly or indirectly connected (to all, or at least one) of the resources of the current focus. For example, while viewing a set of cars we can move to (and focus on) the list of their firms (in this way we interpret disjunctively the elements associated with every object of the focus). To capture this requirement it is enough to allow users to move to a state whose extension is the *current set of transition markers*. For example, consider a user who starts from the class `Persons`, and then restricts his focus to those persons who `workAt FORTH`. Subsequently he restricts his focus through the property `drives`, specifically he restricts his focus to `European`. At that point he asks to change the entity type

to `Cars`. This means that the entity type of the extension of the new state should be `Cars`, and the extension of the new state will contain *European cars which are driven by persons working at FORTH*. The property `drives` (actually its inverse direction), is now a possible facet of the current focus (and a condition is already active, based on the session of the user). Furthermore, the user can proceed and restrict his focus (*European cars which are driven by persons working at FORTH*) to those which are `ForSale`, and so on.

4.2 The Interaction Model for Fuzzy RDF

The general idea is that each transition of the model is now analyzed into $|\Phi|$ transitions, one for each $\varphi \in \Phi$, and each one is signified by its count (therefore now we will have $|\Phi|$ instead of 1 counts). To define these transitions we extend the previous definitions so that each of them takes an interval as additional parameter. Specifically, each *Restrict* takes as input an additional parameter φ, and the same for *Joins*, i.e.:

$$\begin{aligned}
Restrict(E, c, \varphi) &= \{\ e \in E \mid e \in inst(c, \varphi)\} \\
Restrict(E, p : v, \varphi) &= \{\ e \in E \mid (e, p, v) \in inst(p, \varphi)\} \\
Restrict(E, p : vset, \varphi) &= \{\ e \in E \mid \exists\ v' \in vset \text{ and } \ (e, p, v') \in inst(p, \varphi)\} \\
Joins(E, p, \varphi) &= \{\ v \mid \exists\ e \in E \text{ and } \ (e, p, v) \in inst(p, \varphi)\}
\end{aligned}$$

Regarding *class-based transitions*, it follows that for each tm in $TM_{cl}(s.e)$ we now have one $TM_{cl}(s.e, \varphi)$ for each $\varphi \in \Phi$, where:
$TM_{cl}(s.e, \varphi) = \{c \in C \mid Restrict(s.e, c, \varphi) \neq \emptyset\}$, and if the user clicks on a $c \in TM_{cl}(s.e, \varphi)$, then $s'.e = Restrict(s.e, c, \varphi)$.

Regarding *property-based transitions*, for each $p \in Props(s)$, the corresponding transition markers in plain RDF were defined by $Joins(s.e, p)$. Now, each element in $Joins(s.e, p)$ is analyzed to one $Joins(s.e, p, \varphi)$ for each $\varphi \in \Phi$. If the user clicks on a value v in $Joins(s.e, p, \varphi)$, then $s'.e = Restrict(s.e, p : v, \varphi)$. Regarding *presentation*, we do not show intervals, instead we show the corresponding count information. For example, for each $e \in E = \bigcup_{\varphi \in \Phi} TM_{cl}(s.e, \varphi)$ we show e once and its counts for each $\varphi \in \Phi$. Analogously for properties.

Let's now focus on *property paths*. For example consider two property instances pi_1 and pi_2 that form a path (e.g. `(cr2,inv(uses),Bob,0.2)` and `(Bob,worksAt,CSD,0.8)`), each associated with a fuzzy degree d_1 and d_2 respectively. The degree of path $pi_1 \cdot pi_2$ is $\min(d_1, d_2)$, in our case 0.2, since each path actually corresponds to a conjunction. This means that if the user's focus is cars and he wants to restrict it through the organization of the users of the cars, then the path $pi_1 \cdot pi_2$ will be taken into account for computing the count of the transition marker `CSD` whose interval encloses the degree $\min(d_1, d_2)$. To define this precisely, we first introduce some notations. Let $pp = p_1, \ldots, p_k$ be a property path. An *instance path of pp* is a sequence of the form $ip = (v_0, p_1, v_1) \cdot \ldots \cdot (v_{k-1}, p_k, v_k)$ where for all $1 \leq i \leq k$: $(v_{i-1}, p_i, v_i) \in \mathcal{C}(K)$. The *degree* of an instance path ip is defined as the *minimum* degree of its edges (property instance triples). The *degree of a path from o to o'* over pp, denoted as $degree(o, pp, o')$, is the *maximum*

degree of all instance paths of pp between these two objects. We can now define *joins* and *restrictions* based on *fuzzy paths*:

$$Joins(E, pp, \varphi) = \{\ v_k \mid \exists\ e \in E \text{ such that } \ degree(e, pp, v_k) \in \varphi\} \quad (6)$$

$$Restrict(E, pp : v_k, \varphi) = \{\ e \in E \mid degree(e, pp, v_k) \in \varphi\} \quad (7)$$

Now we will analyze the algorithmic aspect of the above (since the previous two definitions were declarative). Consider a property path $pp = p_1 \cdot ... \cdot p_k$. The transition markers at each stage are defined as before, i.e. $M_i = Joins(M_{i-1}, p_i)$. For each individual element $e \in s.e$ we define the set of transition markers of level i (where $1 \leq i \leq k$) which are associated with it, as:

$$ETM_i(e) = \{m_i \in M_i \mid \exists\ s \in ETM_{i-1}(e) \ \text{ and } \ (s, p_i, m_i) \in inst(p_i)\} \quad (8)$$

assuming that $ETM_0(e) = \{e\}$.

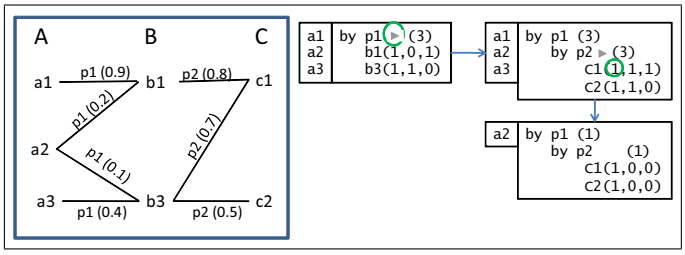


Fig. 5. Interaction over fuzzy paths

For example consider the case shown at Fig. 5. Let A be the extension of the current state ($M_0 = A = s.e$). For a path consisting only of one property p_1 we have that $M_1 = Joins(M_0, p_1) = \{b1, b3\}$, while for $p_1 \cdot p_2$ we have $M_2 = Joins(M_1, p_2) = \{c1, c2\}$. Now, for each element $e \in s.e$ we have the following sets of transition markers of level i:

level 0: $ETM_0(a1) = \{a1\}$ $ETM_0(a2) = \{a2\}$ $ETM_0(a3) = \{a3\}$
level 1: $ETM_1(a1) = \{b1\}$ $ETM_1(a2) = \{b1, b3\}$ $ETM_1(a3) = \{b3\}$
level 2: $ETM_2(a1) = \{c1\}$ $ETM_2(a2) = \{c1, c2\}$ $ETM_2(a3) = \{c1, c2\}$

In addition, for each element $e \in s.e$ and transition marker $m_i \in ETM_i(e)$, we introduce a value denoted by $Deg(e, m_i)$, which is actually the degree of a path from e to m_i over pp (note that if pp is empty then we assume $Deg(e, e) = 1$). This value can be computed gradually (i.e. as the path gets expanded) as follows:

$$Deg(e, m_i) = \max_{m_{i-1} \in ETM_{i-1}(e)} \{\min(degree(m_{i-1}, p_i, m_i), Deg(e, m_{i-1}))\} \quad (9)$$

In our example we have: $Deg(a2, b1) = \max_{a \in ETM_0(a2)}\{\min(degree(a, p_1, b1), Deg(a2, a))\} = \min(degree(a2, p_1, b1), Deg(a2, a2)) = 0.2$. Analogously, $Deg(a2, b3) = 0.1$. Now, the degree of $a2$ to the transition marker $c1$ is computed as: $Deg(a2, c1) = \max_{b \in ETM_1(a2)}\{\min(degree(b, p_2, c1), Deg(a2, b))\} =$ $\max\{\min(degree(b1, p_2, c1), Deg(a2, b1)), \min(degree(b3, p_2, c1), Deg(a2, b3))\}$ $= \max\{\min(0.8, 0.2), \min(0.7, 0.1)\} = \max\{0.2, 0.1\} = 0.2$. Analogously, $Deg(a1, c1) = 0.8$ and $Deg(a3, c1) = 0.4$.

Finally, the *count* for each m_i of M_i that corresponds to φ is given by:

$$count(m_i, \varphi) = |\{e \in s.e \mid Deg(e, m_i) \in \varphi\}| \tag{10}$$

e.g. at Fig. 5 and for $\varphi = (0, 0.3]$, we have $count(c1, \varphi) = 1$. By clicking on the count $count(m_i, \varphi)$ the extension of the current state is restricted as follows $s'.e = \{e \in s.e \mid Deg(e, m_i) \in \varphi\}$.

4.3 Path Expansion and Cycles

Here, we examine how the interaction model for Fuzzy RDF behaves in case we have cycles at schema and instance level. At schema level, a property sequence may have the same starting and ending class forming a cycle (cyclic properties can be considered as a special case where the length of the sequence is 1). In the context of the proposed interaction model, when we have a cyclic schema path (e.g. $pp =$`inv(uses).worksAt.owns` as it is shown in Fig. 6) we may reach transitions markers which are also elements of the initial focus $s.e$, e.g. a transition marker m_i such that $m_i \in s.e$, and we may have to compute its $Deg(m_i, m_i)$. Consider the property path $p_1.p_2.p_3.p_4$ where $p_1 = inv(uses)$, $p_2 = knows$, $p_3 = worksAt$, $p_4 = owns$. If $cr1$ belongs to $s.e$, then $ETM_0(cr1) = \{cr1\}$, $ETM_1(cr1) = \{Alice\}$, $ETM_2(cr1) = \{Bob\}$, $ETM_3(cr1) = \{CSD\}$ and $ETM_4(cr1) = \{cr1\}$. To compute the appropriate count information of the transition marker *cr1*, it is enough to compute $Deg(cr1, cr1)$ according to Eq. 9, as the path $pp = p_1.p_2.p_3.p_4$ is not empty. The point is that the proposed model can handle cycles at schema level without requiring any further configuration.

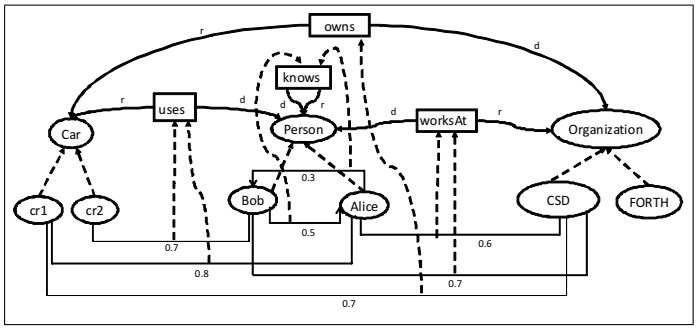
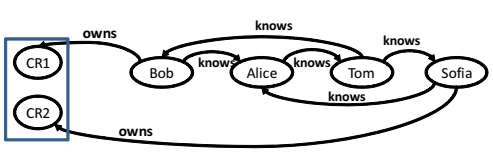


Fig. 6. A Fuzzy RDF graph with cycles at schema and instance level.



TMs for `inv(owns).knows`^{2}		ipath from cr1	ipath from cr2
i=1	A(2)	B.**A**	S.**A**
i=2	T(2)	B.A.**T**	S.A.**T**
i=3	S(2), B(2)	B.A.T.**S**, B.A.T.**B**	S.A.T.**S**, S.A.T.**B**
i=4	A(2)	B.A.T.S.**A**, B.A.T.B.**A**	S.A.T.S.**A**, S.A.T.B.**A**
i=5	T(2)	B.A.T.S.A.**T**, B.A.T.B.A.**T**	S.A.T.S.A.**T**, S.A.T.B.A.**T**

Fig. 7. Instance cycles example

Now consider a user who wants to restrict the initial focus set, being a set of persons, through other persons connected with them through the property *knows* at depth *m*. For this reason the user expands the property *knows m* times and then selects a person. However, at some point we may want to stop suggesting path expansions, in order to avoid prompting to the user the same set of transition markers (which may periodically restrict the initial focus in the same way). For example, in Fig. 7 we can see that when the property `knows` is expanded over 3 times, the transition markers and their count info is periodically being repeated[3]. We propose adopting the following policy: *stop path expansion when each object of the s.e has been made accessible (i.e. restrictable) through all transition markers that are possible.* Below we explain how we can compute the max number of expansion steps. Let $\Gamma = (N, R)$ be a directed graph. A *path* from n_1 to n_{k+1}, is a sequence of edges of the form $(n_1, n_2) \dots (n_k, n_{k+1})$ where $(n_i, n_{i+1}) \in R$, and $i \neq j$ implies that $n_i \neq n_j$. The length of such a path is k, and we shall write $n_1 \overset{k}{\rightsquigarrow} n_{k+1}$ to denote that there exists a path of length k from n_1 to n_{k+1}. We shall also write $n \overset{*}{\rightsquigarrow} n'$ to denote that exists one or more paths from n to n'. Now we define the distance from n to n' as the length of the *shortest* path from n to n', i.e. $Dist(n \rightarrow n') = \min\{\ k \mid n \overset{k}{\rightsquigarrow} n'\}$. Given two subsets A and B of N (i.e. $A, B \subseteq N$), we define the distance from A to B as the *maximum* distance between any pair of nodes form these sets, i.e. $Dist(A, B) = \max\{\ Dist(a \rightarrow b) \mid a \in A, b \in B\}$.

Returning to our problem N is the set of all nodes of the RDF graph. For a property $p \in Pr$ we can define the edges $R_p = \{(a, b) \mid (a, p, b) \in \mathcal{C}(K)\}$. We can now define the reachable nodes from a node n (through p property instances) as $Reachable_p(n) = \{\ n' \mid n \overset{*}{\rightsquigarrow}_p n'\}$, where the meaning of the subscript p is that paths are formed from elements of R_p. Being at a state s, the maximum number of path expansion steps (for property p) that is required are:

$$MaxExpansionSteps(s, p) = Dist(s.e, \bigcup_{n \in s.e} Reachable_p(n))$$

With this number of steps it is guaranteed that each object of $s.e$ has been made accessible (restrictable) through all tms (transition markers) which are possible. The proof is trivial: the path starting from an object o with length bigger than $MaxExpansionSteps(s, p)$ will not encounter a tm that has not already been reached.

Now consider path expansions over different properties (i.e. `inv(owns).knows.knows`). In such cases we would like to to identify the maximum expansion steps for each p that is used in the expansion, or the maximum expansion steps in general. Let *pset* be a set of properties (i.e. $pset \subseteq Pr$). We can define the edges by considering all properties in *pset*, i.e. $R_{pset} = \{(a, b) \mid p \in pset,\ (a, p, b) \in \mathcal{C}(K)\}$. Now we can define $Reachable_{pset}(n) = \{\ n' \mid n \overset{*}{\rightsquigarrow}_{pset} n'\}$, where the subscript *pset* means that paths consist of edges in R_{pset}. The set $Reachable_{pset}(n)$ is the

[3] Only the first letter of a name is shown and paths over `knows` are depicted as sequences of such letters.

set of all tms through which n is accessible. Therefore the tms of all objects in $s.e$, which are accessible through paths using property instances in $pset$, are given by $\bigcup_{n \in s.e} Reachable_{pset}(n)$. Being at a state s, the maximum number of path expansion steps (using properties from $pset$) that is required is:

$$MaxExpansionSteps(s, pset) = Dist(s.e, \bigcup_{n \in s.e} Reachable_{pset}(n))$$

5 Query Language Issues

We have defined the interaction model using only extensions (not intentions), since the expression of the intention depends on the Query Language (QL), or the abstraction of the QL that one adopts. However the underlying information source may be accessible through a QL. Table 1 shows the notation we have used for defining RDF browsing, and their expression in SPARQL. In this description we consider that the extension of the current state is stored in a class with name `ns:temp`[4]. Furthermore we assume that the closure of the KB is stored. However, we should note that *Virtuoso* [3], supports an extended SPARQL version with *subclassOf* and *subproperty* inference at query level. This means that triples entailed by subclass or subproperty statements are not physically stored, but they are added to the result set during query answering [5]. This means that the SPARQL expressions of Table 1 would not require another change.

Notation	Expression in SPARQL
$Restrict(E, p : vset)$, $vset = \{v_1, ..., v_k\}$	`select ?x where { ?x rdf:type ns:temp; ns:p ?V.` `Filter (?V = ns:v_1 ||...|| ?V = ns:v_k)}`
$Restrict(E, c)$	`select ?x where { ?x rdf:type ns:temp; rdf:type ns:c.}`
$Joins(E, p)$, where $E = \{e_1, ..., e_k\}$	`select Distinct ?v where { ?x ns:p ?v.` `Filter (?x = ns:e_1 ||...|| ?x = ns:e_k)}`
$TM_{cl}(s.e)$ and counts	`select Distinct ?c count(*) where{?x rdf:type ?c; rdf:type ns:temp.}` `group by ?c`
$Props(s)$	`select Distinct ?p where{ {?x rdf:type ns:temp; ?p ?v.}` `UNION {?m rdf:type ns:temp. ?n ?p ?m. }}`
$Joins(s.e, p)$ and counts	`select Distinct ?v count(*) where{ ?x rdf:type ns:temp; ns:p ?v.}` `groupby ?v`

Table 1. SPARQL-expression of Notations for RDF Browsing

Regarding QLs for Fuzzy RDF, there is not any standardized (or widely adopted) extension of SPARQL. Some recently proposed deductive systems, e.g. [16, 15], support fuzzy answering over unions of conjunctive queries, by computing the closure of a Fuzzy RDF graph (i.e. $degree(o, type, c)$ is computed as we have defined it, however $degree(o, p, o')$ is not directly supported), storing it into a relational DB, and then using internally SQL queries. For instance, [16] uses MonetDB with the following schema: `type(subject, object,degree)`, `subclassOf(subject, object, degree)`, `subpropertyOf(subject, object, degree)`, and a table $prop_i$`(subject, object, degree)` for every distinct property p_i. Table 2 shows directly the SQL queries that are needed by our interaction model for Fuzzy RDF (again E could also be defined through another query).

[4] Instead of `ns:temp` we could have a set of SPARQL graph patterns. For reasons of space, we do not describe the query construction method.

[5] Similar in spirit with the online approach [5] to query the Web of Linked Data by traversing RDF links during run-time.

Notation	Expression in SQL

$Restrict(E, c, \varphi)$	`select subject from type`
	`where object='c' and subject in E`
	`and degree>phi.low and degree<=phi.up`

$Restrict(E, p : vset, \varphi)$	`select subject from p`
	`where object in VSET and subject in E`
	`and degree>phi.low and degree<=phi.up`

$Joins(E, p_i)$	`select object from p_i where subject in E`

$TM_{cl}(s, \varphi)$ and counts	`select object, sum(case when degree>phi.lower`
	`and degree <= phi.upper then 1 else 0 end),`
	`from type`
	`where subject in s.e group by object`

$ETM(e, M_i, Prev)$	`select object from p_i` `where subject in Prev and object in M_i`

$Deg_{MIN}(e, subj, m_i, d)$,
where $subj \in ETM_{i-1}(e)$
and $d = Deg(e, subj)$
	`select`
	`case when degree > d then d else degree end as DEG_MIN`
	`from p where subject='subj' and object='m_i'`

Table 2. SQL-expression of Notations for Fuzzy RDF Browsing

Regarding property paths, at each step we can compute $M_i = Joins(M_{i-1}, p_i)$ with a single SQL query (as shown in Table 2). The difference of fuzzy paths vs non fuzzy paths, is that for moving from a stage i of the path to a stage $i+1$, we have for each $e \in s.e$ to keep (a) $ETM_i(e)$, and (b) $Deg(e, m_i)$ for each $m_i \in ETM_i(e) \subseteq M_i$. To compute $ETM_i(e)$ we can use a query of the form $ETM(e, M_i, Prev)$ (shown in Table 2) where $Prev = ETM_{i-1}(e)$. To compute $Deg(e, m_i)$ we can use a query of the form $Deg_{MIN}(e, subj, m_i, d)$ for every $subj \in ETM_{i-1}(e)$ (and accordingly $d = Deg(e, subj)$) and then get the max.

6 Conclusion

We proposed a session-based interaction model for exploring Fuzzy RDF KBs in a simple and intuitive manner. To exploit fuzzy degrees the model supports interval-based transition markers and this (exponentially) increases the discrimination power of the interaction. Roughly, for each condition of a (formulated on the fly) query, we have $|\Phi|$ refinements of that condition. This means that for states corresponding to queries with k conditions we now have $|\Phi|^k$ more states. This increase does not affect the friendliness of the interaction since only states leading to non-empty results are given. We also analyzed the query language requirements for realizing this model on top of the query layer. We do not include experimental measurements, since this is not the focus of this paper (and for reasons of space), however we should mention that to compute the class-based transition markers, and their fuzzy counts, for a dataset with 10^7 instances can take up to 3 secs for the case of $|\Phi| = 5$ in MonetDB[6]. Directions for further research regard ranking methods for the transition markers. For instance, fuzzy degrees can be exploited for clustering transition markers, or for ranking them through more refined methods than those proposed for plain RDF (e.g. [11]).

References

1. P. Allard and S. Ferré. Dynamic taxonomies for the semantic web. In *Procs of the 19th Intern. Conf. on Database and Expert Systems Application (DEXA)*, 2008.

[6] More in the extended version of this paper (on preparation).

2. N. Athanasis, V. Christophides, and D. Kotzinos. Generating On the Fly Queries for the Semantic Web: The ICS-FORTH Graphical RQL Interface (GRQL). In *Intern. Semantic Web Conf. (ISWC)*, 2004.
3. O. Erling and I. Mikhailov. RDF Support in the Virtuoso DBMS. In *Procs of 1st Conf. on Social Semantic Web*, 2007.
4. A. Harth. Visinav: Visual web data search and navigation. In *Procs of the 20th Intern. Conf. on Database and Expert Systems Applications (DEXA '09)*, 2009.
5. O. Hartig, C. Bizer, and J.-C. Freytag. Executing SPARQL Queries over the Web of Linked Data. In *Procs of ISWC '09*. Springer, 2009.
6. M. Holi and E. Hyvönen. Fuzzy view-based semantic search. In *Procs of the Semantic Web ASWC '06*, 2006.
7. C. Kohlschütter. Using link analysis to identify aspects in faceted web search. *SIGIR2006 Workshop on Faceted Search*, pages 55–59, 2006.
8. J. Lu, Y. Zhu, X. Zeng, L. Koehl, J. Ma, and G. Zhang. A fuzzy decision support system for garment new product development. In *Australasian Conf. on Artificial Intelligence*, pages 532–543, 2008.
9. E. Mäkelä, E. Hyvönen, and S. Saarela. Ontogator - A Semantic View-Based Search Engine Service for Web Applications. In *Procs of ISWC '06*, pages 847–860, 2006.
10. M. Mazzieri. A Fuzzy RDF Semantics to Represent Trust Metadata. In *1st Workshop on Semantic Web Applications and Perspectives (SWAP2004)*, 2004.
11. E. Oren, R. Delbru, and S. Decker. Extending Faceted Navigation for RDF Data. In *Procs of ISWC '06*, 2006.
12. P. Papadakos, S. Kopidaki, N. Armenatzoglou, and Y. Tzitzikas. On Exploiting Static and Dynamically-mined Metadata for Exploratory Web Searching. *Knowledge and Information Systems*. (accepted for publication in 2011).
13. G. M. Sacco. e-RARE: Interactive Diagnostic Assistance for Rare Diseases through Dynamic Taxonomies. In *DEXA Workshops*, 2008.
14. M. Stefaner, T. Urban, and M. Seefelder. Elastic lists for facet browsing and resource analysis in the enterprise. In *Procs of DEXA'08*, pages 397–401, 2008.
15. U. Straccia, N. Lopes, G. Lukacsy, and A. Polleres. A general framework for representing and reasoning with annotated semantic web data. In *Twenty-Fourth AAAI Conf. on Artificial Intelligence (AAAI-2010)*, 2010.
16. Umberto Straccia. A minimal deductive system for general fuzzy rdf. In *Procs of the 3rd Intern. Conf. on Web Reasoning and Rule Systems (RR '09)*, 2009.
17. M. Tilsner, O. Hoeber, and A. Fiech. Cubansea: Cluster-based visualization of search results. In *Procs of the Inter. Joint Conf. on Web Intelligence and Intelligent Agent Technology (WI-IAT'09)*, 2009.
18. Octavian Udrea, Diego Reforgiato Recupero, and V. S. Subrahmanian. Annotated RDF. *ACM Trans. Comput. Logic*, 11(2), 2010.
19. Jian-Kang Wu, A. Desai Narasimhalu, Babu M. Mehtre, Chian-Prong Lam, and Yong Jian Gao. Core: A content-based retrieval engine for multimedia information systems. *Multimedia Syst.*, 3(1):25–41, 1995.
20. K.-P Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. *In Procs of the SIGCHI Conf. on Human factors in Computing Systems*, 2003.