Querying Fuzzy RDF Knowledge Graphs Data

Guanfeng Li
College of Information Engineering
Ningxia University
Yinchuan, China
wdxgb@nxu.edu.cn

Weijun Li
College of Computer Science & Engineering
North Minzu University
Yinchuan, China
liweijun@nun.edu.cn

Hairong Wang
College of Computer Science & Engineering
North Minzu University
Yinchuan, China
skylan1977@163.com

***Abstract*—Knowledge graphs have become increasingly popular over the past years. The most popular data model for representing knowledge graphs is the Resource Description Framework (RDF). In an open Web environment, it is common case that the RDF knowledge graphs data from different sources may often contain inconsistent or imprecise information. Efficient querying of fuzzy RDF knowledge graphs is therefore of increasing importance. In this paper, we firstly introduce practical extensions of the RDF model to represent vagueness based on fuzzy graph. Then we extend the concept of SPARQL graph pattern making it possible to query an element-level fuzzy RDF knowledge graphs with high degree of truth. Finally, we present a backtracking algorithm for calculating the set of homomorphisms from fuzzy SPARQL graph into a fuzzy RDF graph to demonstrate our proposed methods.**

Keywords—fuzzy RDF, knowledge graphs, SPARQL, fuzzy query

I. INTRODUCTION

Knowledge graphs, as the latest achievement of the development of symbolism, are the main representation form of intelligent data. With the development of knowledge graphs, more and more intelligent data have been released in the form of the Resource Description Framework (RDF) [1]. RDF expose data as triples consisting of subject, predicate and object. RDF datasets can be modeled as labeled directed graphs, where each triple defines an edge from the subject to the object vertex under label predicate. In the construction of the knowledge graph, it is common case that the knowledge graph data from different sources may often contain inconsistent or imprecise information. The study of combining knowledge graph data and imprecision /uncertainty has become an emerging topic for various RDF knowledge graph applications. Particularly the theory of fuzzy sets [2] have been employed to model and handle imprecise information in RDF datasets.

From the perspective of uncertainty level, the fuzzy RDF knowledge graph data model mainly includes two types: one is a triple-level fuzzy model [3, 12, 14, 26], and the other is an element-level fuzzy model [4, 5]. Formally, the triple-level fuzzy RDF model is a 4-tuple ($\langle s, p, o\rangle$ [n]), where s, p and o are subject, predicate and object respectively, and n is a numeric value between 0 and 1. Although this typed of fuzzy RDF model is simple, the added numeric value can be interpreted as various different meanings in the real world such as uncertainty, trust, provenance and so on. The element-level fuzzy RDF model formally is fuzzy triple $\langle \mu_s/s, \mu_p/p, \mu_o/o\rangle$, where s is fuzzy subject and $\mu_s \in [0, 1]$ denote the degree of subject to the universe of an RDF dataset, p is fuzzy predicate and $\mu_p \in [0, 1]$ express the fuzzy degree to the property or relationship being described, o is fuzzy object and $\mu_o \in [0, 1]$ represent the fuzzy degree of the property value. This is a general fuzzy RDF graph model that considers the element-level fuzziness based on fuzzy graph theory [27]. It is capable to describe semantic relationships between fuzzy concepts. Throughout this paper, we consider the data model based on fuzzy graph which synthesizes the fuzzy RDF model defined in [5].

With the increasing amount of fuzzy RDF data which is becoming available, the way we query fuzzy RDF data is a crucial subject for supporting knowledge graph applications in various domains [13]. SPARQL [10], the official W3C recommendation as an RDF query language, provides basic functionalities in order to query RDF data through graph patterns. But classical SPARQL lacks of some expressiveness and usability capabilities to deal with vagueness and imprecise aspects as it follows a Boolean querying of crisp RDF data. As a result, the need to query about the structure and vagueness information in fuzzy RDF knowledge graph applications, has motivated research into extending SPARQL languages to be more expressive than before.

Some works [6, 7, 8, 9] extend SPARQL by allowing to query crisp RDF through graph patterns using regular expressions but they do not address the fuzziness in their approaches. In order to make the expression of flexible queries, a variety of proposals, such as f-SPARQL [15] and SPARQL*f* [16], introduce fuzzy terms and fuzzy operators into FILTER expression of SPARQL queries. However, these works only considers crisp RDF graph. As far as fuzzy RDF graphs are concerned, some extended SPARQL query languages already exist. In [23], the authors propose FURQL (Fuzzy RDF Query Language), a SPARQL extension with navigational capabilities for querying fuzzy RDF data through fuzzy graph path patterns by using regular expressions. Fuzzy conditions can be also used to express fuzzy preferences on data. In [14], the authors propose FSA-SPARQL (Fuzzy Sets and Aggregators based SPARQL), which is focused on extending SPARQL with fuzzy aggregators, in such a way that the user can play with them to express preferences. However, the proposed fuzzy version of RDF in these works is able to express vague and imprecise knowledge about membership relationships. In other words, these works can only address the query problems of triple-level fuzzy RDF graph. In the element-level fuzzy RDF graph, there is a need to get the degree of a vertex and use it in a query. This motivates

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

us to investigate fuzzy query techniques suitable for element-level fuzzy RDF knowledge graph data.

In this paper, we present an extended querying language (called *ef-SPARQL: element-level fuzzy SPARQL*) for element-level fuzzy RDF graph with support of the full SPARQL fragment. It possible to express regular expression patterns instead of predicates in the RDF triple, which allows to express preferences on data through fuzzy conditions and on the structure of the data graph through fuzzy regular expression. The input pattern follows the form of our proposed SPARQL extension is different with the previous work [24]. The main differences are as follows: Firstly, a truth degree can be associated to triple patterns. Secondly, a truth degree can be associated to a given variable in the pattern of my query. In other word, we can get the degree of a vertex and use it in a query. We carefully defined the syntax and semantic of an extension of the query pattern graph that makes it possible to express and interpret such queries. Finally, we present a case study to explain our proposed methods.

The rest of the paper is organized as follows. Firstly, a discussion of some related works is given in Section 2. The preliminary knowledge of the fuzzy RDF data model is introduced in Section 3. Section 4 defines the adopted syntax and semantics of SPARQL and discusses its use to query fuzzy RDF knowledge graph data. We then discuss implementation issues in Section 5. Finally, conclusion and future works are exposed in section 6.

II. RELATED WORK

In this section, we will present a brief literature review of the related work of SPARQL extensions with flexible querying functionalities. Extensions of SPARQL have been studied with the aim to cover fuzziness and user preferences (see [13] for a survey).

Different researchers have extended SPARQL so as to make it more flexible. Kochut et al. [8] and Anyanwu et al. [7] introduced respectively SPARQLeR and SPARQ2L, two extensions of the classical query language SPARQL with path patterns that represent paths between vertices in the RDF graph. They both extend the SPARQL graph patterns with path triple patterns in the predicate position. This makes it possible to express more complex queries such as subgraph extraction queries, reachability queries, etc. Alkhateeb et al. [6] developed the query language PSPARQL(Path SPARQL) that includes regular path queries (RPQs), but the authors focused on the syntactic and semantic redefinition of the RDF model making it possible to express regular expression patterns and to only describe a proof-of-concept implementation. In [28], the same authors extend (P)SPARQL by adding the ability to express complex constraints over regular expressions during path search. Pérez et al. [9] introduced a new query language named nSPARQL, which makes it possible to express complex navigation statements in an RDF graph using nested regular expressions in the predicate position. These languages are more expressive and extend SPARQL by allowing querying RDF data through graph path patterns. However, these query languages do not address the fuzzy querying issue in their approaches.

From another perspective, some approaches [15, 16] extend SPARQL with fuzzy querying functionalities to support user preferences. Cheng et al. [15] proposed the f-SPARQL query language that supports fuzzy querying over ontologies by extending the SPARQL language. This extension is based on threshold query (e.g., asking for people who are tall at a degree greater than 0.7) or general fuzzy queries based on semantic functions. Ma et al. [16] proposed a flexible extension of SPARQL allowing to introduce fuzzy terms and relations into the query language. The objective is to provide users with useful results ranked according to their preferences brought by the query. However, all the aforementioned works only consider crisp RDF graph databases.

As for the fuzzy RDF graph databases, Straccia [26], presented Fuzzy RDF in a general setting where triples were annotated with a degree of truth in [0, 1]. For instance, ⟨ (*audiTT*, type, *SportsCar*), 0.8⟩ is a fuzzy triple, intending that AudiTT is almost a sports car. For the query language, it formalized conjunctive queries. Hartig [29] defined semantics for fuzzy RDF graphs formed by associating trust degree to triplets, and they proposed query evaluation algorithms for fuzzy RDF data with tSPARQL. Zimmermann *et al*. [30] described a generic framework for representing and reasoning with annotated Semantic Web data. The authors formalized the annotated language, the corresponding deductive system and address the query answering problem. FSAQL [12] resembled SPARQL and it was able to query fuzzy values assigned to RDF statements. The FURQL query language proposed in [11] extended SPARQL with navigational capabilities for querying fuzzy RDF graph data through fuzzy graph path patterns by using regular expressions. Fuzzy conditions could be also used to express fuzzy preferences on data. Thresholding via the CUT operator could be set in FURQL. This extension made it possible to query a fuzzy RDF data model, and to express fuzzy preferences on the value of the vertices and the structure of the RDF data graph. The PrefSPARQL extension of SPARQL proposed in [31] introduced 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. FSA-SPARQL [14] was an extension of SPARQL which introduced more user preferences in the FILTER clause. This query language could involve different fuzzy connectives, linguistic modifiers, fuzzy equalities/ inequalities, as well as fuzzy aggregation operations. FSA-SPARQL worked on fuzzy RDF data, handling fuzzy concept memberships, fuzzy role fulfilling, fuzzy subconcept and fuzzy subrole relationships. FSA-SPARQL also enabled to specify a threshold on queries. Although these works can address the query problems of triple-level fuzzy RDF data. However, queries against element-level fuzzy RDF data still have limitations.

The idea of using regular expressions to query graphs has been adopted by query languages such as GQ [18], SoQL [19] and SPARQL [17]. GQ supported arbitrary attributes on vertices, edges and graphs. SoQL was an SQL-like language that allowed users to retrieve paths satisfying various conditions. The W3C decided to boost SPARQL 1.1 with extensive navigational capabilities by the introduction of property paths [17]. Property

paths closely correspond to regular expressions and are a crucial tool in SPARQL if one wants to perform non-trivial navigation through RDF data.

III. PRELIMINARIES

In this section, we present notions and definitions that are necessary for our discussions later. We first give a short overview of RDF. Then we introduce the fuzzy framework to enrich standard RDF graphs with fuzzy values. We furthermore define their semantics, and Finally we will present some preliminaries on SPARQL.

A. RDF Knowledge Graph Model

Knowledge graphs are the main representation form of intelligent data. With the development of knowledge graphs, more and more intelligent data has been released in the form of the RDF. In the RDF data model, the universe is modelled as a set of resources, where a resource is anything that has a universal resource identifier (URI) reference and is described using a set of RDF triples. Formally, an RDF triple is defined as $\langle s, p, o\rangle \in (U \cup B) \times U \times (U \cup L \cup B)$, where U, B, and L are infinite sets of URI reference, Blank vertices, and RDF Literals, respectively. In a triple $\langle s, p, o\rangle$, s is called the subject, p the predicate (or property), and o the object. The interpretation of a triple statement is that subject s has property p with value o. Additionally, assume the existence of an infinite set VAR of variables disjoint from $U \cup L$. Note that any object in one triple, say o_i in $\langle s_i, p_i, o_i\rangle$, can play the role of a subject in another triple, say $\langle o_i, p_j, o_j\rangle$. A set of RDF triples consists of an RDF graph, where subjects and objects are vertices of the graph, and predicates form the oriented edges. Therefore, RDF data is a directed, labelled graph data format for representing Web resources [23]. We focus on the abstract representation of the RDF data model in knowledge graph in this paper.

Definition 1 (RDF Knowledge Graph). An RDF knowledge graph is a directed, labelled graph denoted as $G = (V, E, \Sigma, L)$, in which V is a finite set of vertices, $E \subset V \times V$ is a set of directed edges, Σ is a set of labels, and L: $V \cup E \rightarrow \Sigma$ is a function assigning respectively labels to vertices and edges.

In other words, an RDF knowledge graph is a labeled multi-digraph where the vertex labels are either URIs corresponding to resources or literals, and the edge labels are URIs corresponding to predicates.

B. Fuzzy RDF Knowledge Graph

In knowledge graph applications, information is full of uncertainty due to the openness of the Web. Uncertainty can result from either imprecision of sources or from inconsistency between them. A shortcoming that RDF has been widely criticized for is the lack of an approach to represent vague or ambiguous information. Extending RDF models to incorporate the vagueness is important for many applications. In the paper, we give the formalization of fuzzy RDF data model, which is defined based on the data model of fuzzy RDF graph in [4].

A fuzzy graph [27] $FG = (\mu, \rho)$ is a pair of functions μ: $V \rightarrow [0, 1]$, ρ: $V \times V \rightarrow [0, 1]$ which satisfies $\forall (u_1, u_2) \in V \times V$, $\rho(u_1, u_2) \leq \mu_1(u_1) \wedge \mu_2(u_2)$, where $\wedge$ denotes the minimum.

Definition 2 (Fuzzy RDF Knowledge Graph). Fuzzy RDF knowledge graph G is represented by a 6-tuple $(V, E, \Sigma, L, \mu, \rho)$ where V is a finite set of vertices, $E \subset V \times V$ is a set of directed edges, Σ is a set of labels, L: $V \cup E \rightarrow \Sigma$ is a function assigning labels to vertices and edges respectively, μ: $V \rightarrow [0, 1]$ is a fuzzy subset, and ρ: $E \rightarrow [0, 1]$ is a fuzzy relation of on fuzzy subset μ. Note that $\rho(v_i, v_j) \leq \mu(v_i) \wedge \mu(v_j)$ $v_i, v_j \in V$.

In Definition 2, each vertex $v_i \in V$ of graph G has one label, $L(v_i)$, corresponding to either *subjects* or *objects* in RDF triples datasets. Moreover, $(v_i, v_j) \in E$ is a directed edge from vertex v_i to v_j, with an edge label $L(v_i, v_j)$ that corresponds to the *predicate* in RDF triples. A fuzzy RDF knowledge graph may contain both fuzzy vertices (resp. edges) and crisp vertices (resp. edges) as a fuzzy vertex (resp. edge) with a degree of 0 or 1 can be considered as crisp. Along the same line, a crisp RDF knowledge graph is a special case of fuzzy RDF knowledge graph (where μ: $V \rightarrow \{0, 1\}$ for all $v_i \in V$ and ρ: $V \times V \rightarrow \{0, 1\}$ for all $(v_i, v_j) \in E$), and the fuzzy RDF knowledge graph is a generalization of the crisp RDF graph.

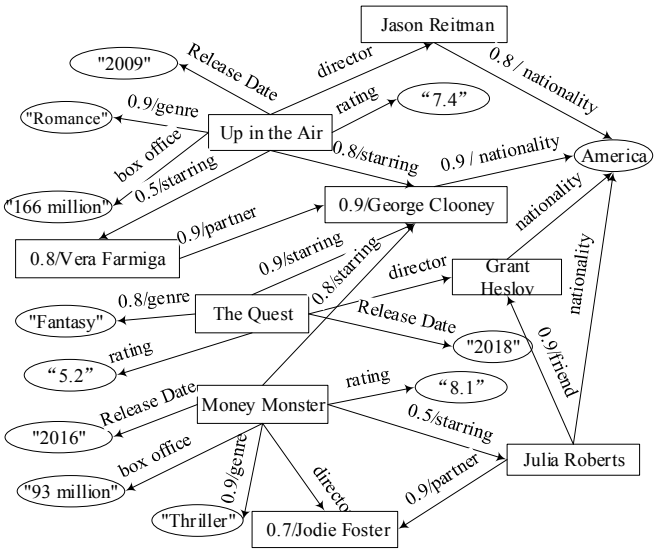

Fig. 1. An excerpt of IMDB's RDF knowledge graph.

Example 1. Let us consider the fuzzy RDF knowledge graph in Fig. 1, which represents an excerpt of the Internet Movie Database (IMDB) [25], a database containing information about movies, actors, directors, producers, among others, as well as their relationships. For readability reasons, we omit URI prefixes. In fuzzy RDF knowledge graph, vertices represent entities or property values of entities, whereas edges represent entity relationships or entity properties. In particular, if a vertex represents a real-world entity, the fuzzy degree associated with it quantifies the entity's identity possibility. For instance, we can assign the vertex "George Clooney" with possibility 0.9, to indicate the likelihood that the vertex corresponds to a real-world actor. On other hand, an edge, such as "partner" and "starring", is representing relationship. Fuzzy degree associated with it reflect the likelihood of the relationship's existence. For example, the fuzzy edge "Vera Farmiga, partner, George Clooney" with a fuzzy degree states that Vera Farmiga and

George Clooney are partners with a satisfaction degree of 0.9. Furthermore, if a vertex represents the attribute value of an entity, the fuzzy degree on the edge between the entity vertex and the property value vertex specifies the possibility of the property value. For example, the genre property of the movie "Up in the Air" is "Romance" in Figure 1, with a possibility of 0.9.

C. Fuzzy RDF Knowledge Graph semantics

Intuitively a fuzzy interpretation represents a possible configuration of the world. We can verify if what is said on a fuzzy RDF knowledge graph is true within the framework of fuzzy logic. As described in [20], any interpretation is relative to a certain vocabulary V. A fuzzy interpretation is also relative to the vocabulary V. Formally a fuzzy simple interpretation I_f of V is a 7-tuple $I_f = (V, Ir, Ip, Lv, Is, Il, Iext)$. Here,

- V is a fuzzy set of vocabulary and some elements have membership degrees. For such an element $x \in V$, its degree $\mu_x \in [0, 1]$ gives an estimation of the belonging of x to V;
- Ir is a non-empty finite set of resources, called the domain or universe of I_f;
- Ip is a finite set of property names, not necessarily disjoint from Ir ;
- Lv is a fuzzy set of literal values;
- Is is a function URIref $\rightarrow Ir \cup Ip$, and $\mu_{Is(x)} \in [0, 1]$ indicates the degree that an entity which is mapped from $x \in$ URIrefs via Is belongs to $Ir \cup Ip$;
- Il: $L_T \rightarrow Ir$ is a function from typed literals to Ir, and $\mu_{Il(x)} \in [0, 1]$ indicates the degree that a type literal x which is mapped via Is belongs to Ir;
- $Iext$ is a function $Ip \rightarrow 2^{Ir \times Ir}$, and $\mu_{(xz)y} \in [0, 1]$ indicates the membership degree that a pair $(Is(x), Is(z))$ belongs to the set $Iext(Is(y))$, where y and z are elements of fuzzy set V.

Given a fuzzy triple, say $\langle \mu_s/s, \mu_p/p, \mu_o/o \rangle$, $I_f(\langle \mu_s/s, \mu_p/p, \mu_o/o \rangle) = true$ if $min(\mu_s, \mu_p, \mu_o, \mu_{Is(s)}, \mu_{Is(p)}, \mu_{Is(o)}) \geq \alpha$ (a given threshold); otherwise, $I_f(\langle \mu_s/s, \mu_p/p, \mu_o/o \rangle) = false$. Given a set of fuzzy triples S, $I_f(S) = false$ if there exists a fuzzy triple $\langle \mu_s/s, \mu_p/p, \mu_o/o \rangle$ and $I_f(\langle \mu_s/s, \mu_p/p, \mu_o/o \rangle) = false$, otherwise $I_f(S) = true$. I_f satisfies S, written as $I_f \models S$ if $I_f(S) = true$; in this case, we say I_f is a fuzzy simple interpretation of S.

D. SPARQL Language

Simple Protocol and RDF Query Language (SPARQL) [10], the official W3C recommendation as an RDF query language, plays the same role for the RDF model as SQL does for relational data model. A SPARQL query consists of three main parts. The argument of the SELECT statement is for specifying which variables should be returned. The clause FROM defines the datasets to be queried, and the statement WHERE contains the query graph patterns.

Roughly speaking, a graph pattern is defined as being triples where variables can occur, composed by binary operators (.) (In the W3C SPARQL syntax, a dot (.) is used as the AND operator, but we avoid it for clarity and use AND instead.), UNION, OPTIONAL and FILTER. In order to avoid ambiguities in the parsing, we present the syntax of SPARQL graph patterns in a more traditional algebraic way, using the binary operators AND, UNION, OPTIONAL and FILTER.

Definition 3 (SPARQL Graph Pattern). A SPARQL graph pattern expression P is defined recursively as follows:

- A tuple $t = \langle s', p', o' \rangle$ is a graph pattern (a triple pattern), where $s' \in \{s, ?s\}$, $p' \in \{p, ?p\}$ and $o' \in \{o, ?o\}$.
- If P_1 and P_2 are graph patterns, then expressions (P_1 AND P_2), (P_1 OPTIONAL P_2), and (P_1 UNION P_2) are graph patterns.
- If P is a graph pattern and C is a SPARQL built-in condition, then the expression (P FILTER C) is a graph pattern.

The notion of graph pattern specifies the topological and content-based constraints chosen by the user. Next, we introduce the notion of SPARQL graph pattern matching which generalizes result subgraph homomorphism [23] with evaluation of the SPARQL graph pattern.

To define the semantics of SPARQL graph pattern expressions, we need to introduce mapping terminology. A solution mapping is a partial function π: $VAR \rightarrow U \cup L$ where the domain of π, dom(π), is the subset of VAR where π is defined. The empty mapping, denoted π_0, is the mapping satisfying that dom(π_0) = Φ.

In the context of SPARQL, the evaluation of a SPARQL graph pattern P over an RDF graph G is defined as a function $[\![P]\!]_G$ which takes a pattern expression and returns a multiset of solution mappings. We follow the approach in [21] defining the semantics as the set of mappings that matches the dataset G.

Definition 4 (SPARQL graph pattern evaluation [21]). The evaluation of a graph pattern P over an RDF graph G, denoted by $[\![P]\!]_G$, is defined recursively as follows:

- P is a *triple* pattern t, then $[\![P]\!]_G = \{\pi \mid \text{dom}(\pi) = \text{var}(t) \wedge \pi(t) \in G\}$, where $\pi(t)$ is the triple obtained by replacing the variables occurring in t according to π, var(t) denote the set of variables occurring in t.
- $[\![(P_1 \text{ AND } P_2)]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$.
- $[\![(P_1 \text{ OPTIONAL } P_2)]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$.
- $[\![(P_1 \text{ UNION } P_2)]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$.
- $[\![(P_1 \text{ FILTER } C)]\!]_G = \sigma_{f(C)}[\![P]\!]_G$

Here t is a triple pattern, P_1, P_2 are graph patterns and C is a filter constraint.

IV. THE FUZZY QUERY LANGUAGE

In this section we are ready to extend SPARQL for querying fuzzy RDF knowledge graph. We first introduce the graph pattern to enrich standard SPARQL graph pattern with regular expressions and fuzzy conditions. Then we define the evaluation of the query pattern in the proposed fuzzy RDF graph.

A. Fuzzy SPARQL Graph Pattern

Before giving the formal definition of fuzzy graph pattern, we first introduce the concepts of fuzzy regular expressions and fuzzy conditions.

A regular expression is a property path, as specified in SPARQL 1.1 [22]. Let *Rex* be a path regular expression, and it can be constructed inductively as $Rex = u \mid R_1 \cdot R_2 \mid R_1 \mid R_2 \mid R^+$. Here u denotes either an edge label or a wildcard symbol * matching any label in U, $R_1 \cdot R_2$ denotes a concatenation of expressions, $R_1 \mid R_2$ denotes disjunction and is an alternative of expressions, R^+ denotes one or more occurrences of R.

A fuzzy condition is a logical combination of fuzzy terms which can be a constant c, a variable ?x, or a fuzzy condition C defined as the form "bound(?x)", "truth(?x)", "?x *op* c", "?x *op* ?y" and "?$x = Ft$". Here ?x, ?$y \in VAR$, $c \in (U \cup L)$, truth(?x) is the truth degree of the variable ?x, *op* is a fuzzy or crisp comparator (e.g., $<, \leq, =, \geq, >, \neq$), and Ft is a predefined or user-defined fuzzy term like high, long, young and so on. If C_1 and C_2 are fuzzy condition, then $\neg C_1$, $C_1 \wedge C_2$, and $C_1 \vee C_2$ are fuzzy conditions. When C_1 and/or C_2 are fuzzy atomic value constraints, the $\neg$, $\wedge$ and $\vee$ need equally to be extended to fuzzy logic. To simplify the discussion, we focus on fuzzy conditions in the simple form given above. Many other fuzzy terms, fuzzy operators and fuzzy relationship that may be used for expressing different kinds of fuzzy conditions and user preferences, are described in [11, 14, 16].

We introduce the notion of fuzzy SPARQL graph pattern, which is a fuzzy extension of the graph pattern notion introduced in [11, 21]. A fuzzy graph pattern allows to express fuzzy preferences on the vertices and edges of a fuzzy RDF graph (through fuzzy conditions) and on the edge (or path) structure of the graph (through fuzzy regular expressions).

Definition 5 (Fuzzy SPARQL Graph Pattern). Let α, τ_1, τ_2 be fuzzy degree variables, β a fuzzy degree value and *Rex* a path regular expression, then a fuzzy SPARQL graph pattern expression is defined recursively as follows:

- A triple pattern $t = \langle s'{:}\ \tau_1,\ p', o'{:}\ \tau_2 \rangle$ associated to a truth degree variable $\alpha \in [0, 1]$ is a fuzzy SPARQL graph pattern, written as $\langle t \rangle{:}\ \alpha$, where $s' \in \{s, ?s\}$, $p' \in \{Rex, ?p\}$ and $o' \in \{o, ?o\}$.
- If P_1 and P_2 are fuzzy SPARQL graph patterns, then expressions (P_1 AND P_2), (P_1 OPTIONAL P_2), and (P_1 UNION P_2) are fuzzy SPARQL graph patterns.
- If P is a fuzzy SPARQL graph pattern and C is a fuzzy condition, then the expression (P FILTER C [WITH β]) is a fuzzy SPARQL graph pattern.

Formally, a fuzzy RDF triple pattern has the form $\langle t \rangle{:}\ \alpha$. Here, α represents the degree in which subject s' has property p' with value o' or subject s' and object o' have a relationship p'. τ_1 and τ_2 represent the truth degree of subject s' and object o'. Although they do not provide any additional information, we allow users to query and use these truth degree variables. Furthermore, the optional parameter [WITH β] indicates the condition that must be satisfied as the minimum membership degree in [0, 1]. It is required in [6] that users need to choose an appropriate value of β to express their requirements. If not specified, 1 is used as default.

B. Fuzzy Extension of SPARQL Query Language

In order to query fuzzy RDF knowledge graphs, we extend the declarative query language SPARQL. Syntactically the extension naturally extends the SPARQL one, by allowing the occurrence of fuzzy graph patterns in the WHERE clause and the occurrence of fuzzy conditions in the FILTER clause. Its simple syntax is given as follow:

```
SELECT... # Result variables
FROM ... #Fuzzy RDF Dataset
WHERE ... #Fuzzy RDF Graph patterns
FILTER … [WITH value] … #Value-constraints
[THRESHOLD value]
```

The language is an extension of standard SPARQL in which triple patterns subject-property-object: $\langle s, p, o \rangle{:}\ \alpha$ are allowed. The query asks about a set of subgraphs that match the fuzzy RDF query patterns in the WHERE statement. The argument of the FROM statement is a fuzzy RDF repository. The argument of the SELECT statement is a list of variables (e.g. ?x, ?y) and truth degree variables (e.g. α). Here, the truth degree variable can be the truth degree of the triple pattern or the truth degree of the variable (vertex). The argument of the FILTER statement is a list of value constraints that should be verified by the query result with a degree equal to a value defined as an argument of the WITH statement. In other words, the WITH statement gives the truth degree of the corresponding fuzzy value constraint. When we omit the WITH part, we obtain boolean value constraints. Moreover, a threshold of truth degree can be associated to a given variable in the FILTER statement, which allows to perform a cut on a particular variables of the answers. The argument of the THRESHOLD statement performs an alpha-cut on the answers (only those having a degree of truth greater or equal to threshold are kept). The THRESHOLD clause also is of course optional. Standard SPARQL is a sublanguage of this extension language and thus triple patterns can still be $\langle s, p, o \rangle$. This is because not all the predicates have to be fuzzy.

Example 2. The following query looks for the recent (importance 0.6) thriller movies in which American actors be the leading role.

```
SELECT ?Movie ?Actor ?l
WHERE {
?Movie Release Date ?Date .
(?Movie starring · nationality "America"): ?l.
?Movie genre "thriller".
FILTER (?Date = recent) WITH 0.6}
THRESHOLD 0.6
```

Here, ?Movie and ?Date are variables, "staring · nationality" is a regular expression, and ?l represents the degree in which the American actor (?Actor) is the leading role of the movie (?Movie). Furthermore, "?Date = recent" is a fuzzy condition expression.

C. Fuzzy SPARQL Graph Pattern Evaluation

A fuzzy SPARQL query defines a fuzzy graph pattern to be matched against a given fuzzy RDF graph. Intuitively, given a fuzzy RDF data graph G, the semantics of a graph pattern P defines a set of matching, where each matching (from variable of P to URIs and literals of G) matches the pattern to a homomorphism subgraph of G [11]. For introducing such a concept, the notion of matching of a regular expression must first be defined.

Definition 6 (Regular expression matching of a path): Let $pa = (\langle s_1, p_1, o_1\rangle, \cdots, \langle o_n, p_n, o_n\rangle) \subseteq G$ be a path of a fuzzy RDF knowledge graph G and Rex be a regular expression. pa matches a regular expression Rex with a degree of truth, $\delta_{Rex}(pa)$, defined as follows, according to the form of Rex (in the following, R, R_1 and R_2 are regular expressions):

- Rex is of the form u with $u \in U$ (resp. "*"). If p_i is u (resp. any $u \in U$) then $\delta_{Rex}(pa) = \rho(p_i)$ else $\delta_{Rex}(pa) = 0$.
- • Rex is of the form $R_1 \cdot R_2$. We denote by P the set of all pairs of paths (p_1, p_2) such that p is of the form $p_1 p_2$. One has $\delta_{Rex}(pa) = \max_P(\min(\delta_{R_1}(p_1), \delta_{R_2}(p_2)))$.
- Rex is of the form $R_1 \mid R_2$. One has $\delta_{Rex}(pa) = \max(\delta_{R_1}(p_1), \delta_{R_2}(p_2))$.
- Rex is of the form R^+. Let PA be the set of all tuples of paths $(p_1, \cdots, p_n)$ $(n > 0)$ such that p is of the form $p_1 \cdots p_n$. One has $\delta_{Rex}(pa) = \max_P(\min(\delta_R(p_1), \cdots, \delta_R(p_n)))$.

As we can see when a regular expression matches with a path of fuzzy RDF knowledge graph, we consider the degree of truth associated to edge of fuzzy RDF graph. Next, we will discuss the issue of interpretation of fuzzy conditions. And we will consider the degree of truth associated to the vertex in the fuzzy RDF graph. In fact, we define conditions on these degrees of truth with value constraints in the FILTER statement.

Definition 7 (Interpretation of fuzzy condition): Let π be a mapping and C be a fuzzy condition. Then π satisfies the fuzzy condition with a degree of truth δ_{co} defined as follows, according to the form of C:

- if C is of the form "truth(?x) $\geq \gamma$", then π satisfies the condition C with a degree of $\delta_{co} = \min(\mu(\pi(?x)), \gamma)$. Here $\gamma \in [0, 1]$, is a threshold of truth degree. Users need to choose an appropriate value of γ to express their requirements.
- if C is of the form "?x *op* c", then π satisfies the condition C with a degree of $\delta_{co} = \min(\mu(\pi(?x)), \mu_{op}(\pi(?x), c))$. Here μ_{op} is membership function of the fuzzy or crisp comparator. In particular, crisp comparison operators have a Boolean semantics, if the condition is evaluated to true, then the degree of truth is 1, otherwise 0.
- if C is of the form "?x *op* ?y", then $\pi(?x)$ and $\pi(?y)$ satisfy the condition C with a degree of $\delta_{co} = \min(\mu(\pi(?x)), \mu_{op}(\pi(?x), \pi(?y)), \mu(\pi(?y)))$.
- if C is of the form "?x = *Ft*", then $\pi(?x)$ satisfies the condition C to the degree $\delta_{co} = \min(\mu(\pi(?x)), \mu_{Ft}(\pi(?x)))$. Here μ_{Ft} is fuzzy membership function of the fuzzy term *Ft*.
- if C is of the form $\neg C_1$, $C_1 \wedge C_2$ or $C_1 \vee C_2$, then the degrees of truth can be defined as: $\delta_{co}(\neg C_1) = 1 - \delta_{co}(\neg C_1)$, $\delta_{co}(C_1 \wedge C_2) = \min(\delta_{co}(C_1), \delta_{co}(C_2))$ and $\delta_{co}(C_1 \vee C_2) = \max(\delta_{co}(C_1), \delta_{co}(C_2))$, respectively.

Definition 8 (Evaluation of a fuzzy graph pattern): The fuzzy RDF graph evaluation of a fuzzy SPARQL graph pattern over G, denoted by $[\![P]\!]_G$ is recursively defined by:

- if P is of the form of a fuzzy triple graph pattern $\langle t\rangle$: α, denoted by $\langle s, p, o\rangle$: α, then $[\![P]\!]_G = \{\pi \mid \mathrm{dom}(\pi) = \mathrm{var}(t) \wedge \pi(t) \in G\}$ and $\alpha = \rho(p)$.
- if P is of the form of a fuzzy triple graph pattern $\langle t\rangle$: α, denoted by $\langle ?s, Rex, ?o\rangle$: α, then $[\![P]\!]_G = \{\pi \mid \mathrm{dom}(\pi) = \mathrm{var}(t) \wedge \pi(t) \in G\}$ and $\alpha = \delta_{Rex}(Rex)$, $\tau_1 = \mathrm{truth}(\pi(?s))$, τ_2 $\mathrm{truth}(\pi(?o))$.
- if P is of the form P_1 AND P_2, then $[\![P]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$.
- if P is of the form P_1 OPTIONAL P_2 then $[\![P]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$.
- if P is of the form P_1 UNION P_2 then $[\![P]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$.
- if P is of the form P_1 FILTER C then $[\![P]\!]_G = \{\pi \in [\![P]\!]_G \mid \pi \models_n C\}$ which denotes the set of mappings in $[\![P]\!]_G$ that satisfy C with a degree $\geq \beta$.

Example 3. Let us consider the fuzzy SPARQL query of Example 2. We evaluate this query according to the fuzzy RDF graph G of Fig.1. The query also specifies a threshold δ_t ($\delta_t = 0.6$ in the example), to indicate that only matches with possibility larger than δ_t should be returned. The matching process is depicted as follows.

Intuitively, this pattern retrieves the list of movies in G, and the matching value of ?Movies is potentially *Up in the Air*, *The Quest* and *Money Monster*. The actors in the three movies are Vera Farmiga, George Clooney and Julia Roberts respectively. Because four paths p_1 = *Up in the Air* − *starring* – George Clooney − *nationality* – America, p_2 = *The Quest* − *starring* – George Clooney − *nationality* – America, p_3 = *Money Monster* − *starring* – George Clooney − *nationality* – America, and p_4 = *Money Monster* − *starring* – Julia Roberts − *nationality* – America match the regular expression "staring · nationality". The degrees of truth are $\delta_{re}(p_1) = 0.8$, $\delta_{re}(p_2) = 0.9$, $\delta_{re}(p_3) = 0.8$, and $\delta_{re}(p_4) = 0.5$. However, the genre of movies *Up in the Air* and *The Quest* are "Romance" and "Fantasy" respectively. Moreover, if we suppose that $\mu_{recent}(2016) = 0.65$, the degree of truth of "?Date = recent" is 0.65. So, *Money Monster* is the only movie which is a thriller movie with degree of truth $\delta_u(''\mathrm{Thriller}'') = 0.9$. Thus, we obtain the following answers:

π_1 = {?Movie → Money Monster, ?Actor → George Clooney, ?l → 0.8}

π_2 = {?Movie → Money Monster, ?Actor →Julia Roberts, ?l → 0.5}

As the degree of truth of the final query result is the minimum of degrees of truth induced by the results described above. There are $\delta_{1P}(G) = 0.65$ and $\delta_{2P}(G) = 0.5$ in this example. So, π_1 satisfies the minimum degree of truth threshold constraint and is the only answer.

V. IMPLEMENTATION ISSUES

The most important and difficult part in the fuzzy SPARQL query language is the fuzzy RDF evaluation of the query pattern P over a fuzzy RDF graph G. Actually, each SPARQL query can be represented by a graph pattern. RDF graph pattern matching in SPARQL is essentially enumerating all PRDF homomorphism from the patterns graph into the data graph G. PRDF homomorphisms extend graph homomorphisms to deal with vertices connected with regular expression patterns, that can be mapped to vertices connected by paths, rather than edge-

to-edge mappings. As a result, any SPARQL query can be equivalently transformed into a subgraph query problem, which locates the subgraph in RDF data graph matching with the query graph. We propose in Algorithm 1 a backtracking technique Q for the processing of a fuzzy query pattern P over a fuzzy RDF graph G. The method generates each possible map from the current one by traversing the parse tree in a depth-first manner. In particular, we need to produce answers with truth degree of the query patterns.

Algorithm 1: Pattern-Match (P, G, μ_p).

Data: an RDF graph pattern P, a fuzzy RDF graph G, and a partial map π_p.
Output: extends the partial map to a set of RDF homomorphism.
1. if Complete (π_p) // $|\mu_p| = =|V_p|$
2. return solution-Found (π_p);
3. $u \leftarrow$ ChooseTerm(V_p); // pick a vertex u of V_p;
4. for each $\langle v, \pi \rangle \in$ *Candidates* (π_p, u, G, P) do
5. *Pattern-Match* (P, G, $\pi_p \bowtie \{(\langle u, v \rangle, \delta)\} \bowtie \pi$);

Algorithm 1 describes the framework for a pattern match query Q over a fuzzy RDF graph G, which is a recursive version of the basic backtrack algorithm [32]. The input of this algorithm is: an RDF graph pattern Q, an RDF graph G, and a partial map μ_p, which includes a set of pairs $\{(\langle u, v \rangle, \delta)\}$ such that u is a term of Q, v is the image of u in G and δ is a truth degree associated with the mapping. If we call this algorithm with (Q, G, $\mu_\emptyset$), where $\mu_\emptyset$ is the map with the empty domain, then it can output all homomorphisms from the pattern graph Q into the fuzzy RDF graph G. Specifically, we define in the following the operations used in the algorithm:

Complete (π_p) checks if each term $u \in V_p$ is mapped to a term in G. It returns TRUE if all $u \in V_P$ are mapped, and FALSE otherwise.

ChooseTerm(V_p) chooses a term $u \in V_p$ to obtain a possible homomorphism.

Candidates (π_p, u, G, P) calculates all possible candidate maps in G for the current term u satisfying the partial map π_p. It returns all sets of pairs $\langle v, \pi \rangle$, where v is a possible map of u, and π is the possible map from a term of regular expression pattern R_i appearing in a triple with u to a term in V_p already mapped in π_p.

After that, the procedure takes each candidate v of the current term $u \in V_p$ and the possible map π, puts v in the mapping pairs and tries to generate the possible candidates of v. This is done recursively in depth-first by calling function Pattern-Match (note that π_p, $\{(\langle u, v \rangle, \delta)\}$, and π are compatible since the set $\langle v, \pi \rangle$ is calculated with respect to π_p). Finally, we have a tree that contains one level with a term from P, i.e., a vertex from P, and one level with the possible images of that term in G. The input to each vertex of each level is the current map. Each possible path in the tree from the root to a leaf labeled by a term of G represents a possible homomorphism.

Proposition 1: Algorithm 1 is correct and complete for enumerating all RDF homomorphism from a given SPARQL graph pattern into a fuzzy RDF graph.

Proof: We can prove this by means of induction. The set of all homomorphism is complete for the empty set at the beginning of the algorithm. Because Candidates [24] is complete, and the number of vertices being finite, the partial homomorphism, i.e., μ_p, are completely extended for the current vertex at each step. Finally, the procedure ends having a homomorphism mapping for each vertex in P.

VI. CONCLUSION

The uncertainty of knowledge graph found today as well as the flexibility of representation offered by RDF raises challenging issues for querying fuzzy RDF knowledge graph. In this paper we have presented an extension of SPARQL to query fuzzy RDF knowledge graph. The extension is able to express fuzzy queries making use of regular expressions and fuzzy conditions. We have provided syntax and semantics to the extension of SPARQL graph pattern. On this basis, we have presented a query evaluation algorithm to subgraph query for processing fuzzy RDF queries.

There are a number of directions for future research. We are currently working on a prototype showing the feasibility of the approach for large practical applications. In the future work, we will continue to work on the extension of SPARQL. Different possibilities exist for extending the graph pattern with more sophisticated fuzzy conditions. One may think for instance of conditions involving fuzzy quantifiers, and of fuzzy preferences on the data and structure.

ACKNOWLEDGMENT

The work was supported by the Natural Science Foundation of Ningxia, China (2019AAC03033) and the key research and development program of Ningxia, China (2018BEB04002).

REFERENCES

[1] W3C: Resource Description Framework (RDF). Retrieved July 3, 2016, from https://www.w3.org/RDF/

[2] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," Fuzzy Sets Systems, 1 (1), 3-28, 1978.

[3] Z. Ma, & L. Yan, "Modeling fuzzy data with RDF and fuzzy relational database models," International Journal of Intelligent Systems, 33(7), 1534-1554, 2018.

[4] Z. Ma, G. Li, & L. Yan, "Fuzzy data modeling and algebraic operations in RDF," Fuzzy Sets and Systems, 351: 41-63, 2018.

[5] Y. Lv, Z. M. Ma, & L. Yan, "Fuzzy RDF: A data model to represent fuzzy metadata," 2008 IEEE International Conference on Fuzzy Systems. IEEE, 2008, pp.1439-1445.

[6] F. Alkhateeb, J.-F. Baget, and J. Euzenat. "Extending SPARQL with regular expression patterns (for querying RDF)," Web Semantics: Science,Services and Agents on the World Wide Web, 7(2):57–73, 2009.

[7] K. Anyanwu, A. Maduko, and A. Sheth, "SPARQ2L: towards support for subgraph extraction queries in RDF databases", In Proc. of the Intl. conference on World Wide Web. ACM, 2007, pp 797-806.

[8] K. J. Kochut and M. Janik, "SPARQLeR: Extended SPARQL for semantic association discovery," The Semantic Web: Research and Applications, pages 145-159. Springer, 2007.

[9] J. Pérez, M. Arenas, & C. Gutierrez, "nSPARQL:A navigational language for RDF," J. Web Sem., 8(4):255-270, 2010.

[10] E. Prud and A. Seaborne, "SPARQL query language for RDF," W3C Recomm., 2008.

[11] O. Pivert, O. Slama, and V. Thion, "An extension of SPARQL with fuzzy navigational capabilities for querying fuzzy RDF data," In Proceedings of the 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, 2016, pp. 2409-2416.

[12] A. Bahri, R. Bouaziz, and F. Gargouri, “Querying fuzzy RDFS semantic annotations,” 2010 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, 2010, pp.1–8.

[13] O. Pivert, O. Slama, and V. Thion, “SPARQL extensions with preferences: a survey,” in Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM, 2016, pp. 1015-1020.

[14] J. M. Almendros-Jiménez, A. Becerra-Terón and G. Moreno, "A fuzzy extension of SPARQL based on fuzzy sets and aggregators," 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Naples, 2017, pp.1-6.

[15] J. Cheng Z. Ma and L. Yan, "f-SPARQL: a flexible extension of SPARQL," International Conference on Database and Expert Systems Applications (DEXA) pp.487-494, 2010.

[16] R. Ma, X. Jia, J. Cheng, and R. Angryk, “SPARQL queries on RDF with fuzzy constraints and preferences,” Journal of Intelligent & Fuzzy Systems, 202:1-13, 2015.

[17] S. Harris, & A. Seaborne, “SPARQL 1.1 query language,” Tech. report, World Wide Web Consortium (W3C), January 2012.

[18] H. He, & A. K. Singh, “Query language and access methods for graph databases,” ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, Bc, Canada, June. DBLP.

[19] R. Ronen, & O. Shmueli, “SoQL: A language for querying and creating data in social networks,” 2009 IEEE 25th International Conference on Data Engineering. IEEE, 2009, pp.1595-1602.

[20] P. Hayes, RDF Semantics, http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[21] M. Arenas and J. P´erez, “ Querying semantic web data with SPARQL,” In Proc. of ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), ACM, 2011, pp 305–316.

[22] S. Harris and A. Seaborne, SPARQL 1.1 Query Language. W3C Recommendation, 2013. http://www.w3.org/TR/sparql11-query.

[23] R. Angles, & C. Gutierrez, “The multiset semantics of SPARQL patterns,” In International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, pp. 20-36.

[24] G. Li, L. Yan, & Z. Ma, “Pattern match query over fuzzy RDF graph,” Knowledge-Based Systems, 165, 460-473, 2019.

[25] M. Hafner, Internet Movie Database, https://www.imdb.com/.

[26] U. Straccia, Foundations of Fuzzy Logic and Semantic Web Languages. CRC Press, 2013.

[27] A. Rosenfeld, Fuzzy Graphs, Fuzzy Sets and Their Applications to Cognitive and Decision Processes (Eds., L.A. Zadeh, K.S. Fu, K. Tanaka, M. Shimura), Academic Press, New York, 1975, pp.77-95.

[28] F. Alkhateeb, J. Euzenat, “Constrained regular expressions for answering RDF-path queries modulo RDFS,” Intl. Journal of Web Information Systems, 10(1):24-50, 2014.

[29] O. Hartig, "Querying Trust in RDF Data with tSPARQL," Semantic Web: Research & Applications, European Semantic Web Conference, Eswc, Heraklion, Crete, Greece, May 31-june Springer-Verlag, 2009.

[30] A. Zimmermann, N. Lopes, A. Polleres, and U. Straccia. “A general framework for representing, reasoning and querying with annotated semantic web data,” J. Web Semantics, 11, 72-95, 2012.

[31] M. Gueroussova, A. Polleres, and S. McIlraith, “SPARQL with qualitative and quantitative preferences,” in Proceedings of the 2nd International Conference on Ordering and Reasoning-Volume 1059. CEURWS. org, 2013, pp. 2-8.