Hindawi
Complexity
Volume 2019, Article ID 8258197, 7 pages
https://doi.org/10.1155/2019/8258197

Research Article

pSPARQL: A Querying Language for Probabilistic RDF Data

Hong Fang

College of Arts and Sciences, Shanghai Polytechnic University, Shanghai 201209, China

Correspondence should be addressed to Hong Fang; fanghong@sspu.edu.cn

Received 19 December 2018; Accepted 19 February 2019; Published 26 March 2019

Guest Editor: Jianxin Li

More and more linked data (taken as knowledge) can be automatically generated from nonstructured data such as text and image via learning, which are often uncertain in practice. On the other hand, most of the existing approaches to processing linked data are mainly designed for certain data. It becomes more and more important to process uncertain linked data in theoretical aspect. In this paper, we present a querying language framework for probabilistic RDF data (an important uncertain linked data), where each triple has a probability, called pSRARQL, built on SPARQL, recommended by W3C as a querying language for RDF databases. pSPARQL can support the full SPARQL and satisfies some important properties such as well-definedness, uniqueness, and some equivalences. Finally, we illustrate that pSPARQL is feasible in expressing practical queries in a real world.

1. Introduction

Resource Description Framework (RDF) [1] is the standard data model in the Semantic Web. In our real world, RDF data (as a knowledge base) possibly contains some uncertainty data due to the diversity of data sources, where RDF data are automatically extracted from different sources, such as YAGO [2]. For instance, some RDF data is generated from raw data via knowledge extraction and machine learning [3]. Indeed, uncertainty is generally a basic feature of data [4–6]. However, RDF model itself provides little support for uncertain data [7]. SPARQL [8], as a querying language for RDF data officially recommended by W3C [9], is unable to process uncertain data [4].

There are many approaches to processing probabilistic RDF [10]. Reference [4] proposes a probabilistic model for SQL over relational data. Reference [5] presents a Bayesian network to represent probabilistic relations in RDF. Reference [11] develops a framework for evaluating SPARQL conjunctive queries (i.e., basic graph patterns, BGP) on RDF probabilistic databases. Reference [12] proposes answering SPARQL queries with RDFS reasoning on probabilistic models that encode statistical relationships among correlated triples, where the proposed probability models are based on either probability distribution function or a disjunctive normal form probability problem. Reference [13] presents effective pruning mechanisms, as well as structural and probabilistic pruning for query answering of SPARQL conjunctive queries (i.e., BGP) over probabilistic RDF data graphs. Reference [14] presents a RESCAL-based approach to query processing in relational data via factorization. Reference [14] presents a heuristic algorithm for query answering of SPARQL conjunctive queries (i.e., BGP) over incomplete and uncertain RDF. Reference [15] presents a framework for SPARQL query answering over probabilistic databases by extending the rich semantics offered by ontologies with probabilistic information. Reference [16] presents a probabilistic knowledge base system, ARCHIMEDESONE, for query answering with inference by scaling up the knowledge expansion and statistical inference algorithms. Reference [17] proposes a probabilistic automata-based framework of query evaluation in the presence of uncertainty efficiently.

Although those approaches can query probabilistic RDF, most of them mainly process SPARQL conjunctive queries, that is, BGP queries. However, those existing probability models have little support for expressive operators (for instance, neither [13] nor [16] discusses OPTIONAL query for RDF) such as OPTIONAL, which is the least conventional operator of SPARQL [18], and DIFF, a difference operator in SPARQL 1.1 [19], which brings more expressivity [20].

In this paper, we present an extended querying language (called *pSPARQL: probabilistic SPARQL*) for probabilistic

RDF databases with support of the full SPARQL fragment. We show that the semantics of pSPARQL can satisfy some important properties such as well-definedness, uniqueness, and some equivalences. Compared with the previous poster in ISWC 2016 [21], in this paper, we present a totally new probabilistic representation model and prove that the newly proposed model can preserve some important properties such as uniqueness and distributive law of equivalence.

The remainder of this paper is structured as follows: the next section recalls RDF and SPARQL. Section 3 introduces the syntax and semantics of pSPARQL and Section 4 discusses some important properties. Finally, we summarize our work in the last section.

2. RDF and SPARQL

In this section, we briefly recall the syntax and semantics of SPARQL. For more readings, please refer to the core SPARQL formalization in [22].

2.1. RDF Graphs. Let I and L be infinite sets of *IRIs* and *literals*, respectively, with $I \cap L = \emptyset$. Let $U = I \cup L$. A triple $(s, p, o) \in U \times I \times U$ is called an *RDF triple*. An *RDF graph* is a finite set of RDF triples.

2.2. Syntax of SPARQL. Let V be a set of *variables*. SPARQL *patterns* are inductively defined as follows:

(i) Any triple from $(U \cup V) \times (I \cup V) \times (U \cup V)$ is a pattern (called a *triple pattern*).

(ii) If P_1 and P_2 are patterns, then so are the following: P_1 UNION P_2, P_1 AND P_2, P_1 DIFF P_2, and P_1 OPT P_2.

(iii) If P is a pattern and F is a constraint (defined next), then P FILTER F is a pattern; we call F the *filter*, which is a Boolean combination of *atomic constraints*, one of the three following forms: bound($?x$) (*bound*), $?x = ?y$ (*equality*), and $?x = c$ (constant equality), for $?x, ?y \in V$ and $c \in U$.

2.3. Semantics of SPARQL. Now, given a graph G and a pattern P, we define the semantics of P on G, denoted by $[\![P]\!]_G$, as a set of mappings (i.e., partial functions from V to U, in the following manner, where we use dom(μ) to denote the domain of μ)

(i) $[\![(u, v, w)]\!]_G := \{\mu : \{u, v, w\} \cap V \longrightarrow U \mid (\mu(u), \mu(v), \mu(w)) \in G\}$.

(ii) $[\![P_1 \text{ UNION } P_2]\!]_G := [\![P_1]\!]_G \cup [\![P_2]\!]_G$.

(iii) $[\![P_1 \text{ AND } P_2]\!]_G = \{\mu_1 \cup \mu_2 \mid \mu_1 \in [\![P_1]\!]_G \text{ and } \mu_2 \in [\![P_2]\!]_G \text{ and } \mu_1 \sim \mu_2\}$.

Here, two mappings μ_1 and μ_2 are called *compatible*, denoted by $\mu_1 \sim \mu_2$, if for any

$?x \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2) = \emptyset, \mu_1(?x) = \mu_2(?x)$.

(iv) $[\![P_1 \text{ DIFF } P_2]\!]_G = \{\mu_1 \in [\![P_1]\!]_G | \neg \exists \mu_2 \in [\![P_2]\!]_G, \mu_1 \sim \mu_2\}$.

(v) $[\![P_1 \text{ OPT } P_2]\!]_G = [\![(P_1 \text{ AND } P_2) \text{ UNION } ((P_1 \text{ DIFF } P_2)]\!]_G$.

(vi) $[\![P_1 \text{ FILTER } P_2]\!]_G := \{\mu \in [\![P_1]\!]_G \mid \mu(F) = true\}$.

Here, for any mapping μ and filter F, the evaluation of F on μ, denoted by $\mu(F)$, is defined in terms of a three-valued logic with truth values *true*, *false*, and *error*. Recall that

F is a Boolean combination of atomic constraints.

For a bound constraint bound($?x$), we define

$$\mu(\text{bound}(?x)) = \begin{cases} true & if\ ?x \in \text{dom}(\mu); \\ false & otherwise. \end{cases} \tag{1}$$

For an equality constraint $?x = ?y$, we define

$$\mu(?x = ?y) = \begin{cases} true & if\ ?x, ?y \in \text{dom}(\mu)\ and\ \mu(?x) = \mu(?y); \\ false & if\ ?x, ?y \in \text{dom}(\mu)\ and\ \mu(?x) \neq \mu(?y); \\ error & otherwise. \end{cases} \tag{2}$$

Thus, when $?x$ and $?y$ do not both belong to dom(μ), the equality constraint evaluates to *error*. Similarly, for a constant-equality constraint $?x = c$, we define

$$\mu(?x = c) = \begin{cases} true & if\ ?x \in \text{dom}(\mu)\ and\ \mu(?x) = c; \\ false & if\ ?x \in \text{dom}(\mu)\ and\ \mu(?x) \neq c; \\ error & otherwise. \end{cases} \tag{3}$$

A Boolean combination is evaluated using the truth tables given in Table 1.

3. Probabilistic RDF and pSPARQL

In this section, we present probabilistic RDF and introduce the syntax and semantics of pSPARQL.

3.1. Probabilistic RDF. *A probabilistic RDF* R is a pair (G, ρ) where G is an RDF graph and ρ is a total function from $G \longrightarrow (0, 1]$. Intuitively speaking, ρ is a probability function mapping each triple to a probability.

For instance, let $R_{\text{John}} = (G, \rho)$ be a probabilistic RDF with $G = \{t_1, t_2, t_3\}$ and ρ is a function from $G \longrightarrow [0, 1]$ defined in Table 2.

Note that we assign a triple to a probability so that we could take triples as atoms in our scenario analogously treated in [13, 16]. This treatment is not direct to characterize the probability of subjects/objects in triples.

3.2. pSPARQL: A Probabilistic SPARQL. In this section, we introduce a probabilistic SPARQL (for short, pSPARQL).

The Syntax of pSPARQL. The syntax of pSPARQL is slightly different from the syntax of SPARQL [22] in filters, where we

newly introduce a fixed variable ?p to express constraints of probability.

A probabilistic atomic filter is one of the four following forms: $?p \geq s, ?p \leq s, ?p = s$ and $?p \neq s$, where $s \in (0, 1]$. The filter of pSPARQL is a Boolean combination of atomic filters and probabilistic atomic filters.

All patterns are called probabilistic patterns in pSPARQL.

The Semantics of pSPARQL. The semantics of probabilistic patterns are defined in terms of sets of pairs of the form (μ, p) (called a solution (with probability), denoted by τ), where μ is a solution of probabilistic patterns and $p \in (0, 1]$. Note that we only consider pairs of form (μ, p) where $p > 0$.

Now, given a probabilistic RDF graph $R = (G, \rho)$ and a probabilistic pattern P, we define the semantics of P on R, denoted by $[\![P]\!]_R$, as a set of solutions with probability, in the following manner:

$$\begin{aligned}
&[\![(u, v, w)]\!]_R \coloneqq \{(\mu, p) : \{u, v, w\} \cap V \\
&\quad \longrightarrow U \mid (\mu(u), \mu(v), \mu(w)) \in G \text{ and } p \\
&\quad = \rho(\mu(u), \mu(v), \mu(w))\}. \\
&[\![P_1 \text{ UNION } P_2]\!]_R \coloneqq \{(\mu, p) \mid (\mu, p_1) \\
&\quad \in [\![P_1]\!]_R \textit{ or } (\mu, p_2) \in [\![P_2]\!]_R \textit{ and } p \\
&\quad = \max\{p_1, p_2\}\}.
\end{aligned} \tag{4}$$

By default, we set $p = \max\{p\}$.

$$\begin{aligned}
&[\![P_1 \text{ AND } P_2]\!]_R \coloneqq \{\mu_1 \cup \mu_2, p \mid (\mu_i, p_i) \\
&\quad \in [\![P_i]\!]_R \ (i = 1, 2) \\
&\textit{and } p = \max_{\mu_1 \cup \mu_2 = \mu_i \cup \mu_j} \min\{p_i, p_j \mid (\mu_i, p_i) \\
&\quad \in [\![P_1]\!]_R \textit{ and } (\mu_j, p_j) \in [\![P_2]\!]_R \textit{ and } \mu_i \sim \mu_j\} \\
&[\![P_1 \text{ DIFF } P_2]\!]_R \coloneqq \{(\mu_1, p_1) \in [\![P_1]\!]_R \mid \neg\exists(\mu_2, p_2) \\
&\quad \in [\![P_2]\!]_R \text{ s.t. } \mu_1 \sim \mu_2\}. \\
&[\![P_1 \text{ OPT } P_2]\!]_R \\
&\quad = [\![(P_1 \text{ AND } P_2) \text{ UNION } (P_1 \text{ DIFF } P_2)]\!]_R. \\
&[\![P_1 \text{ FILTER } F]\!]_G \coloneqq \{\tau = (\mu, p) \in [\![P_1]\!]_G \mid \tau(F) \\
&\quad = \textit{true}\}.
\end{aligned} \tag{5}$$

(i) For a nonprobabilistic filter F, $\tau(F) = \mu(F)$.

(ii) For a Boolean combination F, $\tau(F) = \mu(F)$.

(iii) For a probabilistic filter $?p \geq s$, we define

$$\tau(?p \geq s) = \begin{cases} \textit{true} & \textit{if } p \geq s; \\ \textit{false} & \textit{otherwise}. \end{cases} \tag{6}$$

Table 1: Truth tables for the three-valued semantics.

(a)

p	q	$p \wedge q$	$p \vee q$
true	*true*	*true*	*true*
true	*false*	*false*	*true*
true	*error*	*error*	*true*
false	*true*	*false*	*true*
false	*false*	*false*	*false*
false	*error*	*false*	*error*
error	*true*	*error*	*true*
error	*false*	*false*	*error*
error	*error*	*error*	*error*

(b)

p	$\neg p$
true	*false*
false	*true*
error	*error*

(iv) For a probabilistic filter $?p \leq s$, we define

$$\tau(?p \leq s) = \begin{cases} \textit{true} & \textit{if } p \leq s; \\ \textit{false} & \textit{otherwise}. \end{cases} \tag{7}$$

(v) For a probabilistic filter $?p = s$, we define

$$\tau(?p = s) = \begin{cases} \textit{true} & \textit{if } p = s; \\ \textit{false} & \textit{otherwise}. \end{cases} \tag{8}$$

(vi) For a probabilistic filter $?p \neq s$, we define

$$\tau(?p \neq s) = \begin{cases} \textit{true} & \textit{if } p \neq s; \\ \textit{false} & \textit{otherwise}. \end{cases} \tag{9}$$

Example 1. Given a pattern $P = (?x, \textit{sufferFrom}, ?y)$FILTER$?\ p \geq 0.5$ (i.e., we query those persons who have suffered from some illness with probability over 0.5), we can compute that $[\![P]\!]_{R_{\text{John}}} = \{(\mu, p)\}$ where $\mu = \{?x \longrightarrow \text{John}, ?y \longrightarrow \textit{Schizophrenia}\}$ and $p = 0.84$. However, let $\tau = (\{?x \longrightarrow \text{John}, ?y \longrightarrow \textit{Schizophrenia}\}, 0.32)$, $\tau \in [\![P]\!]_{R_{\text{John}}}$ since $\tau(?p \geq 0.5) = \textit{false}$.

Example 2. Given a pattern $P = (?x, \textit{sufferFrom}, ?y)$ AND $(?x, \textit{Treatedby}, ?z)$ (i.e., we query those who have suffered from some illness and have been treated), we can compute that $[\![P]\!]_{R_{\text{John}}} = \{\tau_1, \tau_2\}$, where

(i) $\tau_1 = (\{?x \longrightarrow \text{John}, ?y \longrightarrow \textit{Schizophrenia}, ?z \longrightarrow \textit{Psychiatrist}\}, 0.32)$;

(ii) $\tau_2 = (\{?x \longrightarrow \text{John}, ?y \longrightarrow \textit{MentalDisorder}, ?z \longrightarrow \textit{Psychiatrist}\}, 0.84)$.

Table 2

No	Triple: (t)	Probability: ($\rho(t)$)
t_1	(*John*, *sufferedFrom*, *Schizophrenia*)	0.32
t_2	(*John*, *sufferedFrom*, *MentalDisorder*)	0.84
t_3	(*John*, Treatedby, *Psychiatrist*)	0.95

Example 3. Given a pattern P = (?x, *sufferFrom*, *Schizophrenia*)UNION (?x, *Treatedby*, *Psychiatrist*) (i.e., we query those who have suffered from schizophrenia or those who are treated by psychiatrists); we can compute that $[\![P]\!]_{R_{\text{John}}} = \{(?x \longrightarrow \text{John}, 0.95)\}\}$.

Note that *?p* is slightly different from variables where the value of *?p* is variable via probability computation, while the value of other variables is fixed. Moreover, we disallow the comparison of probability in filters.

4. Well-Definedness, Uniqueness, and Equivalence of pSPARQL

In this section, we discuss some important properties of pSPARQL.

Firstly, we introduce a property called well-definedness, which can ensure that the semantics of pSPARQL are well defined.

Proposition 4 (well-definedness). *For any pSPARQL pattern P, for any probabilistic RDF R, for any solution* $(\mu, p) \in [\![P]\!]_R$, *we have* $p \in (0, 1]$.

Proof. By induction on the structure of P,

if P is a triple pattern (u, v, w), then $p = ((\mu(u), \mu(v), \mu(w))) \in (0, 1]$;

if P is of the form P_1 UNION P_2, then let us discuss the three cases:

(i) if $(\mu, p) \in [\![P_1]\!]_R$ but $(\mu, p) \notin [\![P_2]\!]_R$, then $p \in (0, 1]$;

(ii) if $(\mu, p) \in [\![P_2]\!]_R$ but $(\mu, p) \notin [\![P_1]\!]_R$, then $p \in (0, 1]$;

(iii) if $(\mu, p) \in [\![P_1]\!]_R$ and $(\mu, p) \in [\![P_2]\!]_R$, then $p = \max\{p_1, p_2\} \in (0, 1]$ by induction.

If P is of the form P_1 AND P_2, then this claim holds by induction, since there exists some solution $(\mu_1, p_1) \in [\![P_1]\!]_R$ and some solution $(\mu_2, p_2) \in [\![P_2]\!]_R$ with $\mu_1 \sim \mu_2$ such that

$$\begin{aligned} &\mu = \mu_1 \cup \mu_2 \\ &\text{and } p \geq \min\{p_1, p_2\} \\ &\text{and } p = \max_{\mu_1 \cup \mu_2 = \mu_i \cup \mu_j} \min\{p_i, p_j \mid (\mu_i, p_i) \\ &\quad \in [\![P_1]\!]_R \text{ and } (\mu_j, p_j) \in [\![P_2]\!]_R \text{ and } \mu_i \sim \mu_j\} \\ &\quad \in (0, 1]. \end{aligned} \tag{10}$$

If P is of the form P_1 DIFF P_2 or P_1 FILTER F, then this claim holds by induction, since

$$[\![P]\!]_R \subseteq [\![P_1]\!]_R. \tag{11}$$

Finally, if P is of the form P_1 OPT P_2, then this claim holds by the cases of P_1 AND P_2, P_1 UNION P_2, and P_1 DIFF P_2 by induction. □

Proposition 5 (uniqueness). *For any pSPARQL pattern P, for any probabilistic RDF R, for any two solutions* $(\mu, p), (\mu', p') \in [\![P]\!]_R$, *if* $\mu = \mu'$, *then* $p = p'$.

Proof. By induction on the structure of P, we have the following.

If P is a triple pattern (u, v, w), then this claim directly holds by definition, since $p_1 = p_2 = \rho(\mu(u), \mu(v), \mu(w)))$.

If P is of the form P_1 UNION P_2, then let us discuss the three cases:

(i) If $(\mu, p) \in [\![P_1]\!]_R$ but $(\mu, p) \notin [\![P_2]\!]_R$, then this claim holds by induction.

(ii) If $(\mu, p) \in [\![P_2]\!]_R$ but $(\mu, p) \notin [\![P_1]\!]_R$, then this claim holds by induction.

(iii) If there exist $(\mu, p_1) \in [\![P_1]\!]_R$ and $(\mu, p_2) \in [\![P_2]\!]_R$, then this claim holds by induction, since $p = \max\{p_1, p_2\}$.

If P is of the form P_1 AND P_2, then this claim holds by induction, since there exists some solution $(\mu_1, p_1) \in [\![P_1]\!]_R$ and some solution $(\mu_2, p_2) \in [\![P_2]\!]_R$ with $\mu_1 \sim \mu_2$ such that $\mu = \mu_1 \cup \mu_2$ and $= \max_{\mu_1 \cup \mu_2 = \mu_i \cup \mu_j} \min\{p_i, p_j \mid (\mu_i, p_i) \in [\![P_1]\!]_R \text{ and } (\mu_j, p_j) \in [\![P_2]\!]_R \text{ and } \mu_i \sim \mu_j\}$. Therefore, p is unique.

If P is of the form P_1 DIFF P_2 or P_1 FILTER F, then this claim holds by induction, since

$$[\![P]\!]_R \subseteq [\![P_1]\!]_R. \tag{12}$$

Finally, we discuss the equivalence of patterns in pSPARQL. Let P and Q be two patterns in pSPARQL. We say that P is *equivalent* to Q, denoted by $P \equiv_p Q$, if for any probabilistic RDF R, $[\![P]\!]_R = [\![Q]\!]_R$. □

Next, we show that pSPARQL satisfies the distributive law of equivalence, which is proven to be important in SPARQL.

Proposition 6 (distributive law). *Let P_1, P_2, and P_3 be three patterns in pSPARQL and let F be a filter. The following holds:*

(1) (P_1 UNION P_2) FILTER $F \equiv_p$ (P_1 FILTER F) UNION (P_2 FILTER F);

(2) P_1 AND (P_2 UNION P_3) $\equiv_p$ (P_1 AND P_2) UNION (P_1 AND P_3);

(3) $(P_1$ UNION $P_2)$ AND $P_3 \equiv_p (P_1$ AND $P_3)$ UNION $(P_2$ AND $P_3)$;

(4) $(P_1$ UNION $P_2)$ DIFF $P_3 \equiv_p (P_1$ DIFF $P_3)$ UNION $(P_2$ DIFF $P_3)$;

(5) $(P_1$ UNION $P_2)$ OPT $P_3 \equiv_p (P_1$ OPT $P_3)$ UNION $(P_2$ OPT $P_3)$.

Proof (sketch). The first claim directly holds by the definition.

Now, we show the second item. Let R be a probabilistic RDF of the form (G, ρ). If $(\mu, p) \in [\![P_1$ AND $(P_2$UNION $P_3)]\!]_R$, then there must exist some solution $(\mu, p') \in [\![(P_1$ AND $P_2)$ UNION $(P_1$ AND $P_3)]\!]_R$. By Proposition 5, we can conclude that $p = p'$. Then $[\![P_1$ AND $(P_2$ UNION $P_3]\!]_R \subseteq [\![(P_1$ AND $P_2) \cup (P_1$ UNION $P_3]\!]_R$.

On the other hand, $(\mu, p) \in [\![(P_1$ AND $P_2) \cup (P_1$ UNION $P_3]\!]_R$; then there must exist some solution $(\mu, p') \in [\![P_1$ AND $(P_2$ UNION $P_3)]\!]_R$. By Proposition 5, we can conclude that $p = p'$. Then $[\![(P_1$ AND $P_2) \cup (P_1$ UNION $P_3]\!]_R \subseteq [\![P_1$ AND $(P_2$ UNION $P_3]\!]_R$.

Analogously, we can prove the third item and the fourth item.

Finally, we could prove the fifth item by using the third item and the fourth item. □

5. A Practical Example

In this section, we illustrate the application of pSPARQL in a real world via a practical example, where a probabilistic RDF is introduced in [11] shown in Figure 1.

Consider the following four queries (Q_1, Q_2, Q_3, Q_4) in pSPARQL.

(1) Q_1: What causes fatigue associated with some illness over 0.65 probability?

Q_1 is formally expressed in pSPARQL as follows:

SELECT *?x* ((*Fatigue*, *CauseOf*, *?x*) AND ((*?x*, *AssociatedWith*, *?z*) FILTER *?p* >0.65).

The solution of Q_1 is as follows:

$$[\![Q_1]\!]_{Rrsv} = \{(\{?x \longrightarrow Flu\}, 0.6)\}. \quad (13)$$

Note that $[\![(?x, AssociatedWith, ?z)]\!]_{Rrsv} = \{(\{?x \longrightarrow Flu, ?z \longrightarrow Cough\}, 0.7), (\{?x \longrightarrow Pneumonia, ?z \longrightarrow Bronchitis\}, 0.6)\}$. Thus $[\![(?x, AssociatedWith, ?z)$ FILTER$?p > 0.65]\!]_{Rrsv} = \{(\{?x \longrightarrow Flu, ?z \longrightarrow Cough\}, 0.7)\}$. Then $[\![Q_1]\!]_{Rrsv} = \{(\{?x \longrightarrow Flu\}, 0.6)\}$, since $[\![(Fatigue, CauseOf, ?x)]\!]_{Rrsv} = \{(\{?x \longrightarrow Flu\}, 0.6), (\{?x \longrightarrow Pneumonie\}, 0.6)\}$.

(2) Q_2: What are associated with cough over 0.7 probability?

Q_2 is formally expressed in pSRARQL as follows:

SELECT *?x* ((*?x*, *AssociatedWith*, *?y*) FILTER *?y* = '*Cough*' ∧*?p* >0.7).

The solution of Q_2 is as follows:

$$[\![Q_2]\!]_{Rrsv} = \{(\{?x \longrightarrow Bronchitis, ?y \longrightarrow Cough\}, 0.8)\}. \quad (14)$$

Table 3

?x	*?y*	*?p*
Bronchitis	*Cough*	0.8
Bronchitis	*RSV*	0.6
RSV	*Cough*	0.7
Flu	*Cough*	0.7
Pneumonia	*Bronchitis*	0.6

Note that $[\![(?x, AssociatedWith, ?y)]\!]_{Rrsv}$ is shown in Table 3.

Thus $[\![(?x, AssociatedWith, ?y)$ *FILTER*$? y =$ '*Cough*'$]\!]_{Rrsv} = \{(\{?x \longrightarrow$ Bronchitis, $?y \longrightarrow$ Cough$\}, 0.8), (\{?x \longrightarrow$ RSV, $?y \longrightarrow$ Cough$\}, 0.7), (\{?x \longrightarrow$ Flu, $?y \longrightarrow$ Cough$\}, 0.7)\}$.

Then $[\![Q_2]\!]_{Rrsv} = \{(\{?x \longrightarrow Bronchitis, ?y \longrightarrow Cough\}, 0.8)\}$.

(3) Q_3: What is probability of bronchitis associated with cough directly or indirectly?

Q_3 is formally expressed as follows:

SELECT *?x* ((*?x*, *AssociatedWith*, *?y*) UNION ((*?x*, *AssociatedWith*, *?z*) AND (*?z*, *AssociatedWith*, *?y*)) FILTER *?x* = 'Bronchitis'∧*?y* = '*Cough*').

The solution of Q_3 is as follows:

$$[\![Q_3]\!]_{R_{rsv}} = \{(\{?x \longrightarrow Bronchitis, ?y \longrightarrow Cough\}, 0.8)\}. \quad (15)$$

Note that $[\![((?x, AssociatedWith, ?y)]\!]_{R_{rsv}}$ is shown in Table 4.

Note that $[\![?x, AssociatedWith, ?z)$ AND (?z, $AssociatedWith, ?y)]\!]_{R_{rsv}}$ is shown in Table 5.

Thus $[\![(?x, AssociatedWith, ?y)$ UNION $((?x, AssociatedWith, ?z)$ AND $(?z, AssociatedWith, ?y))]\!]_{Rrsv}$ is shown in Table 6.

Then

$$[\![Q_3]\!]_{Rrsv} = \{(\{?x \longrightarrow Bronchitis, ?y \longrightarrow Cough\}, 0.8)\}. \quad (16)$$

(4) Q_4: What are associated with cough excluding bronchitis?

Q_4 is expressed in pSPARQL as follows:

SELECT *?x* (((*?x*, *AssociatedWith*, *?y*) FILTER *?y* = '*Cough*') DIFF ((*?x*, *AssociatedWith*, *?y*) FILTER *?x* = '*Bronchitis*' ∧*?y* = '*Cough*')).

The solution of Q_4 is as follows:

$$[\![Q_4]\!]_{Rrsv} = \{(\{?x \longrightarrow Flu, ?y \longrightarrow Cough\}, 0.7)\}. \quad (17)$$

Note that $[\![(?x, AssociatedWith, ?y)$FILTER$?\ y =$ '*Cough*'$]\!]_{R_{rsv}}$ is shown in Table 7.

Note that $[\![(?x, AssociatedWith, ?y)$FILTER$?\ x =' Bronchitis' \wedge ?y =$ '*Cough*'$]\!]_{R_{rsv}}$ is shown in Table 8.

Then $[\![Q_4]\!]_{Rrsv} = \{(\{?x \longrightarrow Flu, ?y \longrightarrow Cough\}, 0.7)\}$.

In short, we could express many interesting queries with respect to probabilistic RDF via pSPARQL, which are useful

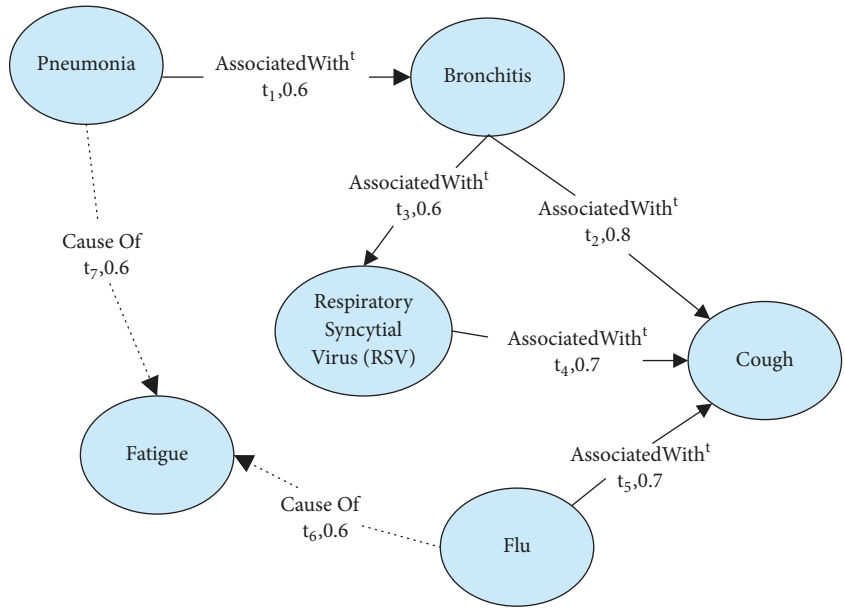

Figure 1: A virus RDF graph R_{rsv} [11].

Table 4

?x	*?y*	*?p*
Bronchitis	*Cough*	0.8
Bronchitis	*RSV*	0.6
RSV	*Cough*	0.7
Flu	*Cough*	0.7
Pneumonia	*Bronchitis*	0.6

Table 5

?x	*?y*	*?z*	*?p*
Bronchitis	*Cough*	*RSV*	0.6
Pneumonia	*Cough*	*Bronchitis*	0.6
Pneumonia	*RSV*	*Bronchitis*	0.6

Table 6

?x	*?y*	*?z*	*?p*
Bronchitis	*Cough*		0.8
Bronchitis	*RSV*		0.6
RSV	*Cough*		0.7
Flu	*Cough*		0.7
Pneumonia	*Bronchitis*		0.6
Bronchitis	*Cough*	*RSV*	0.6
Pneumonia	*Cough*	*Bronchitis*	0.6
Pneumonia	*RSV*	*Bronchitis*	0.6

in a practical world. Compared with SPARQL, where we obtain only connection via SPARQL querying, we could quantize the connection via pSPARQL, so that we could obtain more specific solutions.

6. Conclusions

In this paper, we extended SPARQL to support querying over probabilistic RDF. In the future, we will discuss some

Table 7

?x	?y	?p
Bronchitis	*Cough*	0.8
RSV	*Cough*	0.7
Flu	*Cough*	0.7

Table 8

?x	?y	?p
Bronchitis	*Cough*	0.8

foundational properties of pSPARQL and implement it in a prototype to provide the full SPARQL query answering services for probabilistic RDF. As a future work, we are interested in presenting probabilistic semantics of RDF graphs in a unified framework, where many applications could be supported.

Data Availability

No data were used to support this study.

Disclosure

An earlier version of this work was presented at "International Conference on Big Scientific Data Management 2018."

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the program of the key discipline "Applied Mathematics" of Shanghai Polytechnic University (XXKPY1604).

References

[1] "RDF primer, W3C Recommendation, February 2004".

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, pp. 697–706, Alberta, Canada, May 2007.

[3] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, no. 3, pp. 37–53, 1996.

[4] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," in *Proceedings of the VLDB'04*, pp. 864–875, 2004.

[5] Y. Fukushige, "Representing probabilistic relations in RDF in," in *Proceedings of the ISWC-URSW'05*, pp. 106-107, 2005.

[6] D. Suciu, *Probabilistic Databases, Encyclopedia of Database Systems*, Springer, 2009.

[7] O. Udrea, V. Subrahmanian, and Z. Majkic, "Probabilistic RDF," in *Proceedings of the 2006 IEEE International Conference on Information Reuse & Integration*, pp. 172–177, Waikoloa Village, HI, USA, September 2006.

[8] "SPARQL query language for RDF, W3C Recommendation, January 2008".

[9] P. T. Wood, "Query languages for graph databases," *SIGMOD Record*, vol. 41, no. 1, pp. 50–60, 2012.

[10] A. Khan and L. Chen, "On uncertain graphs modeling and queries," in *Proceedings of the PVLDB Endowment*, vol. 8, pp. 2042-2043, 2015.

[11] H. Huang and C. Liu, "Query evaluation on probabilistic RDF databases," in *Proceedings of the WISE'09*, pp. 307–320, 2009.

[12] C. Szeto, E. Hung, and Y. Deng, "SPARQL query answering with RDFS reasoning on correlated probabilistic data," in *Proceedings of the WAIM'11*, pp. 56–67, 2011.

[13] X. Lian and L. Chen, "Efficient query answering in probabilistic RDF graphs," in *Proceedings of the the 2011 international conference*, p. 157, Athens, Greece, June 2011.

[14] D. Krompaß, M. Nickel, and V. Tresp, "Querying factorized probabilistic triple databases," in *Proceedings of the ISWC'14*, pp. 114–129, 2014.

[15] J. Schoenfisch, "Querying probabilistic ontologies with SPARQL," in *Proceedings of the KI'14*, pp. 2245–2256, 2014.

[16] X. Zhou, Y. Chen, and D. Z. Wang, "ArchimedesOne: Query processing over probabilistic knowledge bases," *Proceedings of the VLDB Endowment*, vol. 9, no. 13, pp. 1461–1464, 2016.

[17] T. Andronikos, A. Singh, K. Giannakis, and S. Sioutas, "Computing probabilistic queries in the presence of uncertainty via probabilistic automata," in *Proceedings of the ALGOCLOUD'17*, pp. 106–120, 2017.

[18] X. Zhang and J. Van den Bussche, "On the primitivity of operators in SPARQL," *Information Processing Letters*, vol. 114, no. 9, pp. 480–485, 2014.

[19] "SPARQL 1.1 query language, W3C Recommendation, March 2013".

[20] X. Zhang, J. Van den Bussche, K. Wang, and Z. Wang, "On the satisfiability problem of patterns in SPARQL 1.1," in *Proceedings of the AAAI'18*, pp. 2054–2061, 2018.

[21] H. Fang and X. Zhang, "pSPARQL: a querying language for probabilistic RDF (extended abstract)," in *Proceedings of the ISWC'16*, Posters, 2016.

[22] J. Pérez, M. Arenas, and C. Gutierrez, "Semantics and complexity of SPARQL," *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 3, pp. 1–45, 2009.