Path discovery by Querying the federation of Relational Database and RDF Graph

Xiaowang Zhang\textsuperscript{1,3} Jiahui Zhang\textsuperscript{1,2} Muhammad Qasim Yasin\textsuperscript{2,3}
Wenrui Wu\textsuperscript{1,3} Zhiyong Feng\textsuperscript{2,3}
\textsuperscript{1}School of Computer Science and Technology, Tianjin University, Tianjin 300350, P. R. China
\textsuperscript{2}School of Computer Software, Tianjin University, Tianjin 300350, P. R. China
\textsuperscript{3}Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300350, P.R. China

ABSTRACT

The class of queries for detecting path is an important as those can extract implicit binary relations over the nodes of input graphs. Most of the path querying languages used by the RDF community, like property paths in W3C SPARQL 1.1 and nested regular expressions in nSPARQL are based on the regular expressions. Federated queries allow for combining graph patterns and relational database that enables the evaluations over several heterogeneous data resources within a single query. Federated queries in W3C SPARQL 1.1 currently evaluated over different SPARQL endpoints. In this paper, we present a federated path querying language as an extension of regular path querying language for supporting RDF graph integration with relational database. The federated path querying language is absolutely more expressive than nested regular expressions and negation-free property paths. Its additional expressivity can be used for capturing the conjunction and federation of nested regular path queries. Despite the increase in expressivity, we also show that federated path queries are still enjoy a low computational complexity and can be evaluated efficiently.

CCS Concepts

- Information systems \rightarrow Database query processing; Query languages for non-relational engines;

Keywords

Heterogeneous Database; RDF; Relational Database; Regular Path Query; Federated Path Query

1. INTRODUCTION

The Resource Description Framework (RDF) [31] recommended by World Wide Web Consortium (W3C) is a standard graph-oriented model for interchanging data on the Web [1]. RDF has implemented in a broad range of applications including the semantic web, social network, bio-informatics, geographical data, etc[1]. Graph-structured data is typical to access due its navigational nature [16] [22] [12]. Navigational path queries on graph databases return binary relations over the nodes of the graph [5]. Many existing navigational query languages for graphs are based on binary relational algebra such as XPath (a standard navigational query language for trees [26]) or regular expressions such as RPQ (regular path queries) [25].

SPARQL [32] recommended by W3C has become the standard language for querying RDF data since 2008 by inheriting classical relational languages such as SQL. However, SPARQL only provides limited navigational functionalities for RDF [29] [39]. Recently, there are several proposed languages with navigational capabilities for querying RDF graphs [27] [19] [29] [3] [33]. Roughly, Versa [27] is the first language for RDF with navigational capabilities by using XPath over the XML serialization of RDF graphs. SPARQLeR proposed by Kochut et al. [19] extends SPARQL by allowing path variables. CFSPARQL proposed by Alkhateeb et al. [3] allows constraints over regular expressions in PSPARQL where variables are allowed in regular expressions. nSPARQL proposed by Pérez et al. [29] extends SPARQL by allowing nested regular expressions in triple patterns Indeed, nSPARQL is still expressible in SPARQL if the transitive closure relation is absent [39]. In March 2013, SPARQL 1.1 [33] recommended by W3C allows property paths which strengthen the navigational capabilities of SPARQL1.0 and [10] [35] extend property paths by adding some operators such as intersection etc.

However, those regular expressions-based extensions of SPARQL are still limited in representing some more expressive navigational queries which are not expressed in regular expressions. It let us consider the RDF graph dataset (G) have information about points of longitude and latitude on the map as in Figure [1] and a relational database (D) as in Table [1].

A record in Table [1] depicts an order that at some time, a passenger placed to travel from a location to another. In the response of to the order of the passenger, a vehicle is allocated to the driver and asked to fulfill the order of the passenger. But sometime there is no vehicle at the station then the driver with already allocated vehicle near the location of the passenger is supposed to be asked to fulfill the order of the passenger by picking him/her from his location. Which can be possible by querying the federation of relational and RDF (graph data).

Assume that there are three passengers A,B and C. Passenger A has placed an order at 8:15AM that he want to hire a vehicle to travel from point P3 to point P5 as in Table [1] at 8:25, the vehicle with Passenger A is near to point P1 and this is recorded. Three minutes later, at 8:28, Passenger B asked for a vehicle and he want to go point 4 (P4) from point P1 and this message is stored in Table [1].Meanwhile, Passenger C also called for vehicle and he wants to go from point P2 to point P5 at 8:40AM and it recorded in relational database. The system receives the three queries related to same
path. By discovery of right path and having information about the vehicle type and time, with one vehicle we can accommodate all of three passenger A, B, and C. as in Figure 1 by selecting path “P1 → P2 → P4 → P5” the driver can accommodate the passenger A,B,C.

Due to the limited space, we omit all proofs in this paper but available in a TR in the link[1] for arXiv.org.

2. PRELIMINARIES

In this section, we briefly recall RDF graphs and the syntax and semantics of nested regular expressions, largely following the excellent exposition [29].

2.1 RDF graphs

An RDF statement is a subject-predicate-object structure, called RDF triple which represents properties and the resources of those resources. For the sake of simplicity similar to [29], we assume that RDF data is composed only IRIs. Formally, let U be an infinite set of IRIs. A triple (s, p, o) ∈ U × U × U is called an RDF triple. An RDF graph G is a finite set of RDF triples. We use adom(G) to denote the active domain of G, i.e., the set of all elements from U occurring in G.

For instance, a RDF graph can be modeled in an RDF graph where each labeled-edge of the form a ↪ b is directly translated into a triple (a, p, b).

Let G be an RDF graph. A path π = (c1, c2, ..., cm) in G is a non-empty finite sequence of constants from G, where, for every i ∈ {1, 2, ..., m − 1}, ci and ci+1 exactly occur in the same triple of G (i.e., (c1, c, ci+1), (ci, c, ci+1), and (ci, c1, ci+1) etc.). Note that the precedence between ci and ci+1 in a path is independent of the positions of ci and ci+1 in a triple.

To capture all binary relations on triples, three different navigation axes, namely, next, edge, and node, and their inverses, i.e., next⁻¹, edge⁻¹, and node⁻¹, are introduced to move through an RDF triple (s, p, o).

Let Σ = \(\{\text{axis}, \text{axis} :: c \mid c \in U\}\) where axis ∈ \{self, next, edge, node, next⁻¹, edge⁻¹, node⁻¹\}. Let G be an RDF graph. We use Σ(G) to denote the set of all symbols (axis, axis :: c | c ∈ adom(G)) occurring in G.

Let π = (c1, ..., cm) a path in G. A trace of path π is a string over Σ(G) written by \(T(\pi) = l_1 \ldots l_{m+1}\) where, for all \(i \in \{1, ..., m−1\}\), \((c_i, c_i+1)\) is labeled by \(l_i\) in the following manner: let axis ∈ \{next, edge, node\},

- \(l_1 = \text{self}\) if \(c_1 = c_{i+1}\);
- \(l_1 = \text{self} :: c\) if \(c_1 = c_{i+1}\);
- \(l_i = \text{next} :: c\) if \((c_i, c_{i+1}) \in G\);
- \(l_i = \text{edge} :: c\) if \((c_i, c_{i+1}, c) \in G\);
- \(l_i = \text{node} :: c\) if \((c_i, c_{i+1}, c) \in G\);
- \(l_i = \text{next} :: c\) if \((c_i, c_{i+1}) \in G\) for some \(c \in \text{adom}(G)\);
- \(l_i = \text{edge} :: c\) if \((c_i, c_{i+1}, c) \in G\) for some \(c \in \text{adom}(G)\);
- \(l_i = \text{node} :: c\) if \((c_i, c_{i+1}, c) \in G\) for some \(c \in \text{adom}(G)\);
- \(l_i = \text{axs} :: c\) if \((c_i, c_{i+1})\) is labeled by \(axis\);
- \(l_i = \text{axs} :: c\) if \((c_i, c_{i+1})\) is labeled by \(axis\).

We use Trace(π) to denote the set of all traces of π.

Note that it is possible that a path has multiple traces since any two nodes possibly occur in the multiple triples. For example, an RDF graph G = \{(a, b, c), (a, c, b)\} and given a path π = (abc), both \(\text{edge} :: c\)(node :: a) and \(\text{next} :: c\)(node⁻¹ :: a) are traces of π.

2.2 Nested regular expressions

Nested regular expressions (nre) are defined by the following formal syntax:

\[ e ::= \text{axis} | \text{axis} :: (e ∈ U) | \text{axis} :: [e/e'] | e/e' | e'^* \]

Here the nesting nre-expression is of the form axis :: [e].

Given an RDF graph G, the evaluation of e on G, denoted by \([e]_G\), is a binary relation inductively defined as follows:

\[
\begin{align*}
[\text{self}]_G &= \{(c, c) \mid c \in \text{adom}(G)\}; \\
[\text{self} :: c]_G &= [\text{self}]_G \cap \{(c, c)\}; \\
[\text{next}]_G &= \{(a, b) \mid \exists c \in (a, c, b) \in G\}; \\
[\text{next} :: c]_G &= \{(a, b) \mid (a, c, b) \in G\}; \\
[\text{edge}]_G &= \{(a, b) \mid \exists c \in (a, c, b) \in G\}; \\
[\text{edge} :: c]_G &= \{(a, b) \mid (a, c, b) \in G\}; \\
[\text{node}]_G &= \{(a, b) \mid \exists c \in (a, c, b) \in G\}; \\
[\text{node} :: c]_G &= \{(a, b) \mid (a, c, b) \in G\}; \\
[\text{axs}]_G &= \{(a, b) \mid (b, a) \in [\text{axis}]_G\}; \\
[\text{axs} :: c]_G &= \{(a, b) \mid (b, a) \in [\text{axis}]_G\}; \\
[\text{axs} :: c]_G &= \{(a, b) \mid (b, a) \in [\text{axis}]_G\}; \\
[\text{axs} :: c]_G &= \{(a, b) \mid (b, a) \in [\text{axis}]_G\}; \\
[\text{axs} :: c]_G &= \{(a, b) \mid (b, a) \in [\text{axis}]_G\}; \\
[\text{axs} :: c]_G &= \{(a, b) \mid (b, a) \in [\text{axis}]_G\}; \\
[\text{axs} :: e]_G &= \{e\} \cup [e]_G; \\
[\text{axs} :: e]_G &= \{e\} \cup [e]_G; \\
[\text{axs} :: e]_G &= \{e\} \cup [e]_G; \\
[\text{axs} :: e]_G &= \{e\} \cup [e]_G; \\
[\text{axs} :: e]_G &= \{e\} \cup [e]_G; \quad \ldots \]
\]

A standard RDF data is composed of IRIs, blank nodes, and literals. For the purposes of this paper, the distinction between IRIs and literals will not be important.
Query evaluation.

Let \( V \) be a set of variables, disjoint with \( U \). It is a SPARQL convention to prefix each variable with a question mark “?”. An nre-triple pattern is of the form \((u, e, v)\) where \(u, v \in U \cup V\) and \(e\) is an nre. Given an RDF graph \( G \), the semantics of \((u, e, v)\) is defined as follows:

\[
[(u, e, v)]_G \equiv \{ \mu : (u, v) \cap V \rightarrow U | (\mu(u), \mu(v)) \in [e]_G \}.
\]

Here, for any mapping \( \mu \) and any constant \( c \in U \), we agree that \( \mu(c) = c \) itself.

A nested regular path query (NRPQ) \( q(u, v) \) is of the form \( (u', e', v') \) where
- \( q \) is the name of NRPQ;
- \( \{u, v\} \cap V \subseteq \{u', v'\} \cap V \);
- \( \{u', e', v'\} \) is an nre-triple pattern.

Given an RDF graph \( G \), an NRPQ \( q(u, v) \), and a mapping \( \mu \), the query evaluation problem is deciding whether \( \mu \) is in \([q(u, v)]_G\). The complexity of query evaluation problem is in time \( O(|G| \cdot |q|) \)[29].

3. CONJUNCTIVE NRPQ

In this section, we introduce an extension of nested regular path queries named conjunctive nested regular path queries (CNRPQ).

3.1 Syntax and semantics

In syntax, the conjunctive NRPQ extends NRPQ in a natural way.

Formally, an CNRPQ is of the form \( q(u, v) \) defined as follows:

\[
q(u, v) := \bigwedge_{i=1}^{n} (u_i, e_i, v_i)
\]

(1)

where
- \( q \) is the name of CNRPQ;
- \( \{u, v\} \cap V \subseteq \{u_i, v_i\} \cap V \);
- each \( \{u_i, e_i, v_i\} \) for \( i \in \{1, \ldots, n\} \) is an nre-triple pattern.

Note that the first item ensures that each CNRPQ is bounded, that is, all variables in \( u, v \) also occurs in some subqueries of the CNRPQ. The second item states that all nre-triple patterns of CNRPQ are NRPQ. By default, if both \( u \) and \( v \) are constants, i.e., \( u, v \in U \), then CNRPQ returns the empty mapping \( \mu_0 \), that is, a mapping with the empty domain. In this sense, CNRPQ is taken as a boolean query, where either “true” or “false” is returned.

For instance, let \( Q(?, \lambda) = (\text{next} :: \text{father}, \text{next} :: \text{father}, \text{next} :: \text{father}, \text{next} :: \text{father}) \) a CNRPQ. Clearly, \( Q \) represents the “grandfather” relationship.

Semantically, let \( q(u, v) \) be a CNRPQ of the form (1) and \( G \) be an RDF graph, \([q(u, v)]_G\) returns a set of mappings with the domain \( \{u, v\} \cap V \) defined as follows:

\[
\{ \mu | (u, v) \cap V | \mu = \mu_1 \cup \mu_2 \cup \ldots \cup \mu_n \text{ and } \forall i \in \{1, \ldots, n\}, \mu_i \in [(u_i, e_i, v_i)]_G \}.
\]

Intuitively, each mapping \( \mu \) of \( q(u, v) \) on \( G \) is the restriction of \( \mu_1 \cup \mu_2 \cup \ldots \cup \mu_n \) where each \( \mu_i \) on \( (u, v) \cap V \) is a mapping of a subquery \( q_i(u_i, v_i) = (u_i, e_i, v_i) \) for \( i = 1, 2, \ldots, n \).

**Example 1.** Let \( G = \{(a, p, b), (b, q, c), (a, r, c)\} \) (shown in Fig. 2) and \( H = \{(a, p, b), (b, q, c), (a, r, d)\} \) (shown in Fig. 3) be two RDF graphs.

In this paper, we simply write a conjunctive query as a Datalog rule (1).

Consider a CNRPQ \( q(?, x, ?, y) = (\text{next} :: ?, p)/\text{next} :: q, ?) \) and \( (\text{next} :: ?, p)/\text{next} :: q, ?) \). We have \([q(?, x, ?, y)]_G = \{(?x = a, ?y = c)\} \) while \([Q(?, x, ?, y)]_H = \emptyset\).

In other words, Example 1 shows that the query \( Q \) can distinguish graph \( G \) from \( H \). However, we find that there exists no any CNRPQ to distinguish graph \( G \) from \( H \) in the following subsection.

3.2 CNRPQ is not expressible in NRPQ

In this subsection, we theoretically show that CNRPQ has more expressive power than NRPQ. Firstly, we define the notion of expressiveness between two query languages.

Let \( L_1 \) and \( L_2 \) be two query languages on RDF graphs. We say \( L_1 \) is expressible in \( L_2 \) if for any query \( q \), there exists some query \( q' \) for any RDF graph \( G \) such that \([q']_G = [q]_G\).

Secondly, we introduce an extension of nre nre(\( \cap \)) by adding the intersection operator \( \cap \) in nre and then we will show that nre(\( \cap \)) can express the intersection of nre-expressions. Finally, we show that the intersection of nre-expressions is not expressed by any nre-expression.

Let \( e_1 \) and \( e_2 \) be two nre-expressions. We use \( e_1 \cap e_2 \) to denote the intersection of \( e_1 \) and \( e_2 \). The evaluation of \( e_1 \cap e_2 \) is defined as follows: let \( G \) be an RDF graph,

\[
[e_1 \cap e_2]_G := [e_1]_G \cap [e_2]_G.
\]

Analogously, we could define NRPQ\( ^{\cap} \) corresponding to nre(\( \cap \)).

Next, we will show that nre(\( \cap \)) is not expressible in nre.

An RDF graph \( G \) is called p-RDF graph if all predicates in all triplles of \( G \) are p and neither subject nor object is p. Let \( G \) be a p-RDF graph. An induced graph of \( G \) written by index(\( G \)) is a node-labeled undirected graph obtained from \( G \) in the following way:

let \( index(G) = (V, E, \lambda) \),
- \( V(G) = V_1 \cup V_2 \) where \( V_1 = \{u_a, w_b | (a, p, b) \in G\} \) and \( V_2 = \{u_{ab} | (a, p, b) \in G\} \);
- \( E(G) = \{(u_{ab}, w_b), (u_{ab}, w_b) | (a, p, b) \in G\} \);
- \( \lambda(v_a) = a, \lambda(u_{ab}) = p, \text{ and } \lambda(w_b) = b \);
- \( \lambda(v_1) = \lambda(v_2) \) implies \( v_1 = v_2 \) for \( v_1, v_2 \in V_1 \).

Clearly, for every p-RDF graph, its all induced graphs are isomorphic.

A p-RDF graph \( G \) is called strongly acyclic if index(\( G \)) is acyclic. For instance, the p-RDF graph \( (a, p, b) \) is strongly acyclic.

We use nre\( ^{cf} \) to denote the constant-free nre, that is, axis :: c is free.

**Lemma 2.** For any nre\( ^{cf} \) expression \( e, \) if \( (a, b) \in [e]_G \) for some p-RDF graph \( G \) and some pair \( (a, b) \) with \( a, b \in \text{const}(G) \) then there exists some strongly acyclic p-RDF graph \( H \) such that \( (a, b) \in [e]_H \).

The following property shows that the intersection of nre-expressions cannot be expressed by any nre-expression.

**Proposition 3.** nre(\( \cap \)) is not expressible in nre.

By Proposition 3 we can conclude an important result.

**Theorem 4.** CNRPQ is not expressible in NRPQ.
4. FEDERATED PATH QUERIES

In this section, we introduce two extensions of conjunctive nested regular path queries named \textit{federated conjunctive nested regular path queries (FCNRPQ)} and \textit{union of federated conjunctive nested regular path queries (UF CNRPQ)} for heterogeneous databases with RDF graphs and relational databases.

4.1 FCNRPQ

Let \( R \) be a set of relation names. An FCNRPQ is of the form \( q(u, v) \) defined as follows:

\[
q(u, v) := \mathcal{F} \land \bigwedge_{i=1}^{n} (u_i, e_i, v_i);
\]

where
- \( q \) is the name of FCNRPQ;
- \( \{u, v\} \cap V \subseteq \{u_1, \ldots, u_n, v_1, \ldots, v_m\} \cup \text{vars}(\mathcal{F}) \cap V \);
- each \( (u_i, e_i, v_i) \) for \( i \in \{1, \ldots, n\} \) is an nre-triple pattern;
- \( \mathcal{F} \) is a conjunction combination of literals \( R(w_1, \ldots, w_m) \) defined as follows:

\[
\mathcal{F} := R(w_1, \ldots, w_m) \mid \mathcal{F}_1 \land \mathcal{F}_2.
\]

Here
- \( R \) is a relation name;
- \( \{w_1, \ldots, w_m\} \subseteq V \cup U \);
- \( \text{vars}(\mathcal{F}) \) is the collection of all variables occurring in \( \mathcal{F} \).

Intuitively speaking, FCNRPQ is an extension of CNRPQ by introducing the conjunctive queries on relations. By default, we allow \( q(u, v) = \mathcal{F} \), that is, CNRPQ is absent. In this case, FCNRPQ is taken as a fragment of FCNRPQ.

Semantically, let \( q(u, v) \) be an FCNRPQ of the form \( 3 \) and \( D = (G, D) \) be a heterogeneous database where \( G \) is an RDF graph and \( D \) is a set of relations, \( \{q(u, v)\}_G \) returns a set of mappings defined as follows:

\[
\{\mu \mid \{u, v\} \cap V \subseteq \{\mu_1, \ldots, \mu_n\} \cup \text{vars}(\mathcal{F}) \cap V \land \forall i \in \{1, \ldots, n\}, \mu_i \in \{(u_i, e_i, v_i)\}_G \}.
\]

Here \( \{\mathcal{F}\}_D \) is defined in the following inductive way:
- Basically, let \( R^D \) be a relation of \( D \) mapped to \( R \):
  \[
  [R(w_1, \ldots, w_m)]_D = \{\mu \mid \text{dom}(\mu) = \{w_1, \ldots, w_m\} \cap V \land \{\mu(w_1), \ldots, \mu(w_m)\} \in R^D \};
  \]
- Inductively, \( [\mathcal{F}_1 \land \mathcal{F}_2]_D = [\mathcal{F}_1]_D \times [\mathcal{F}_2]_D \), where \( \Omega_1 \times \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \mu_1 \sim \mu_2\} \) for any two sets of mappings \( \Omega_1 \) and \( \Omega_2 \). Here, two mappings \( \mu_1 \) and \( \mu_2 \) are compatible \( 25 \), written by \( \mu_1 \sim \mu_2 \), if for every variable \( ?x \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2), \mu_1(?x) = \mu_2(?x) \).

In the following, we will show that FCNRPQ has more expressive power than CNRPQ.

To do so, we introduce the following lemma.

**Lemma 5.** For any CNRPQ \( q(?x, ?y) \), for any RDF graph \( G \) with \( G = \{a, a, a\} \), if \( a \) does not occur in \( q(?x, ?y) \) then \([q(?x, ?y)]_G \neq \emptyset \).

By Lemma 5 we can conclude an important result.

**Theorem 6.** FCNRPQ is not expressible in CNRPQ.

4.2 Union of FCNRPQ

A UFCNRPQ is of the form \( Q(u, v) \) defined as follows:

\[
Q(u, v) := \bigvee_{i=1}^{n} q_i(u, v);
\]

where
- \( q_i \) is the name of UFCNRPQ;
- each \( q_i(u, v) \) is an ECNRQ for \( i \in \{1, 2, \ldots, n\} \).

Semantically, let \( q(u, v) \) be a UFCNRPQ of the form \( 4 \) and \( D = (G, D) \) be a heterogeneous database where \( G \) is an RDF graph and \( D \) is a set of relations, \( \{q(u, v)\}_D \) returns a set of mappings defined as follows:

\[
[q(u, v)]_D = \bigcup_{i=1}^{n} [q_i(u, v)]_D.
\]

In the following, we will show that UFCNRPQ has more expressive power than FCNRQP.

**Lemma 7.** For any FCNRQ \( q(?x, ?y) \), for any heterogeneous database \( D = (G, \emptyset) \), if \( G \) is a singleton then \([q(?x, ?y)]_D \) contains at most one mapping.

**Theorem 8.** UFCNRPQ is not expressible in FCNRQ.

5. EXPRESSIONINESS OF FPQ

In previous sections, our proposed path queries CNRPQ, CNRPQ, FCNRQ, and UFCNRQ are called \textit{federated path queries (FPQ)}.

In this section, we investigate the expressiveness of FPQ.

5.1 Expressiveness of variants of RPQ

To discuss subtly, we introduce some interesting fragments of nre as follows \(37\):
- \( \text{nre}_0: \text{basic nre, i.e., nre only consisting of } "\text{axis}"", "\text{/}" and "\text{\text{as}}s"");
- \( \text{nre}_0()\): basic nre by adding the operator "\text{\text{as}}x"");
- \( \text{nre}_0(n)\): basic nre by adding nesting nre \( axis :: [e] \).

According to the three fragments of nre, namely, \( \text{nre}_0, \text{nre}_0() \), \( \text{nre}_0(n) \), we can introduce the following three fragments of NRPQ as follows:
- \( \text{RPQ}: \) an NRPQ with \( \text{nre}_0 \)-expressions;
- \( \text{RPQ}()\): an NRPQ with \( \text{nre}_0()\)-expressions;
- \( \text{RPQ}(n)\): an NRPQ with \( \text{nre}_0(n)\)-expressions.

In this sense, FPQ can be denoted as \( \text{RPQ}(\lambda, \mathcal{X}) \) \( \lambda \) is a set of operators as \( \lambda, \Lambda, \mathcal{N}, \mathcal{A}, \mathcal{R}, \mathcal{V} \) as follows:
- \( \Lambda \): the conjunctive operator;
- \( \mathcal{N} \): the federated operator;
- \( \mathcal{V} \): the union operator.

Thus we can denote CNRPQ, FCNRQ, and UFCNRQ as follows:
- \( \text{CNRPQ}: \text{RPQ}[\lambda, \mathcal{N}, \mathcal{A}]\);
- \( \text{FCNRQ}: \text{RPQ}[\lambda, \mathcal{N}, \mathcal{A}, \mathcal{R}]\);
- \( \text{UFCNRQ}: \text{RPQ}[\lambda, \mathcal{N}, \mathcal{A}, \mathcal{R}, \mathcal{V}]\).

By the proofs of Theorem 4, Theorem 6 and Theorem 8, we can show that the conjunctive operator \( \lambda \), the federated operator \( \mathcal{N} \), and the union operator \( \mathcal{V} \) are primitive. So we can conclude that each fragment with the operator is not expressible in any fragment without the operator \( 38 \).

That is, \( \text{RPQ}(\mathcal{X} \cup \{\text{axis}\}) \) is not expressible in \( \text{RPQ}(\mathcal{X} = \{\text{\text{as}}\}) \) where \( \text{\text{as}} \) is the placeholder of "\text{\text{axis}}", "\text{\text{\text{as}}x}"", or "\text{\text{as}}s".

Finally, Figure 4 provides the implication of the known results on RDF graphs for the general relations between some interesting fragments of FPQ where \( \mathcal{L}_1 \rightarrow \mathcal{L}_2 \) to denote that \( \mathcal{L}_1 \) is expressible in \( \mathcal{L}_2 \). Note that this paper does not discuss all fragments of FPQ such as \( \text{RPQ}(\lambda) \) while the left fragments leave open.
5.2 Expressiveness of property paths in FPQ

In syntax, property paths (PP) in SPARQL 1.1 are inductively defined as follows [29]:

- Any IRI in $P$ is a property path.
- If $elt_1$ and $elt_2$ are property paths, then so are the followings: $elt_1 / elt_2$ and $elt_1 | elt_2$.
- If $elt$ is a property path, then so are the followings: $elt?$, $elt*$, and $elt\+$.
- If $iri_i \in I$ for $1 \leq i \leq n + m$, then $elt$ is a property path where $elt = (iri_1 | ... | iri_n | iri_{n+1} | ... | iri_{n+m})$.

Semantically, let $P$ be a property path pattern of the form $(u, elt, v)$ where $elt$ is a property path, then the evaluation of $P$ over an RDF graph $G$ is defined as follows:

\[
[P]_G := \{ \mu | \text{dom}(\mu) = \text{vars}\{u, v\} \text{ and } \langle \mu(u), \mu(v) \rangle \in [elt]_G \},
\]

where $[elt]_G$ is inductively defined as follows:

- $[iri_i]_G := \{(a, b) | (a, iri, b) \in G\}$.
- $[elt_1 / elt_2]_G := \{(a, b) \mid \text{there exists } c \text{ such that } (a, c) \in [elt_1]_G \text{ and } (c, b) \in [elt_2]_G\}$.
- $[elt_1 | elt_2]_G := [elt_1]_G \cup [elt_2]_G$.
- $[(iri_1 | ... | iri_n)]_G := \{(a, b) \mid \exists c, (a, c, b) \in G \text{ and } \forall i \in \{1, ..., n\}, (a, b) \not\in [iri_i]_G\}$.
- $[elt?]_G := \{(a, b) | (b, a) \in [elt]_G\}$.
- $[elt*]_G := \{(a, b) | a \in \text{dom}(G) \cup [elt]_G\}$.
- $[elt+]_G := [elt]_G \cup [elt / elt]_G \cup [elt / elt / elt]_G \cup \cdots$.
- $[elt*]_G := \{(a, a) | a \in \text{dom}(G) \cup [elt]_G \cup [elt / elt]_G \cup [elt / elt / elt]_G \cup \cdots\}$.

A PP query is of the form $Q_P(x, elt, ?y)$ where $elt$ is a PP. Let $G$ be an RDF graph. $([Q_P(x, pp, ?y)]) = \{(x \rightarrow a, y \rightarrow b) | (a, b) \in [pp]_G\}$. For simplification, we still use PP to denote the PP query language, where each query is a PP query.

Since nre is not expressible in PP, we conclude the following proposition.

**Proposition 9.** **NRQ** is not expressible in PP.

Moreover, since PP allows the negation of (atomic) property, PP is not expressible in nre [39]. To prove that PP is not expressible in UFCNRPQ, we first introduce the following property named monotonicity.

A UFCNRPQ query $q$ is **monotone** if for any two datasets $D = (G, D)$ and $D' = (G', D')$, $D \subseteq D'$ implies $[q]_D \subseteq [q]_{D'}$. Here $D \subseteq D'$ is defined as follows:

- $G \subseteq G'$;
- for any $D \in D$, there exists some $D' \in D'$ such that $D \subseteq D'$.

Since each UFCNRPQ query can be rewritten a conjunctive first-order query (CQ) which is monotone [1], we conclude the following result.

**Lemma 10.** All UFCNRPQ queries are monotone.

**Proposition 11.** PP is not expressible in UFCNRPQ.

Since the negation-free PP can be expressible in nre [39], it is clear that the negation-free PP queries are also expressible in NRQ. Theoretically, it is feasible to introduce nre with negation [39] to extend our proposed FPQ.

5.3 Expressiveness of FPQ in SPARQL

To compare FPQ with SPARQL in expressiveness, we recall briefly nSPARQL [29].

In syntax, nSPARQL (graph) patterns are defined in an inductive way:

- Each nre-triples are nSPARQL patterns;
- $P_1 \cup P_2$ and $P_1 \cap P_2$ are nSPARQL patterns if $P_1$ and $P_2$ are patterns;
- $\text{SELECT } \varnothing(P)$ is an nSPARQL pattern if $P$ is an nSPARQL pattern and $S \subseteq V$;
- $P_1 \cup P_2$ is an nSPARQL pattern if $P_1$ is an nSPARQL pattern and $C$ is a constraint.

Semantically, the evaluation of general nSPARQL patterns is defined as follows:

- $[P_1 \cup P_2]_G = [P_1]_G \cup [P_2]_G$.
- $[P_1 \cap P_2]_G = [P_1]_G \cap [P_2]_G$.
- $[\text{SELECT } \varnothing(P)]_G = \{ \mu | \forall \Omega \in \text{vars}(\Omega) \} \subseteq [P]_G$.
- $[\text{FILTER } C]_G = \{ \mu | \mu(C) = \text{true} \}$.

Since the Kleene star $*$ is not expressible in SPARQL [29], let nre be the Kleene star-free nre. We use RPQ$^d$ to denote RPQ by only allowing nre$^d$-expressions.

Since nSPARQL does not support querying on relations, we conclude the inexpressivity of RPQ$^d$ in nSPARQL.

**Proposition 12.** RPQ$^d$ is not expressible in nSPARQL.

We use nSPARQL$^d$ to denote an extension of SPARQL by allowing the Kleene star-free nre$^d$-triple patterns.

**Proposition 13.** RPQ$^d$, $\langle n, \land, \lor \rangle$ is expressible in nSPARQL$^d$.

**Theorem 14.** The following properties hold:

- RPQ$^d$, $\langle n, \land, \lor \rangle$ is not expressible in SPARQL.
- RPQ$^d$, $\langle n, \land, \lor \rangle$ is expressible in SPARQL.

In short, the Kleene star $*$ in nre and the federated operator $\otimes$ are indeed beyond the expressiveness of SPARQL.
At the end of this section, we will discuss the complexity of the query evaluation problem in FPQ.

Let $D = (G, D)$ be a heterogeneous database. Given a FCNRPQ $q(u, v)$ and a mapping $\mu$, the query evaluation problem is deciding whether $\mu \in [q(u, v)]_G$, that is, whether the tuple $\mu$ is in the result of the query $q$ on the heterogeneous database $D$.

There are two kinds of computational complexity in the query evaluation problem \cite{1, 2}:

- the data complexity refers to the complexity w.r.t. the size of the heterogeneous database $D$, given a fixed query $q$; and
- the combined complexity refers to the complexity w.r.t. the size of query $q$ and the heterogeneous database $D$.

As a result, we can conclude the following proposition.

**Proposition 15.** The followings hold:

1. The data complexity of the query evaluation of FCNRPQ is in polynomial time;
2. The combined complexity of the query evaluation of FCNRPQ is in NP-complete time.

Note that the query evaluation of UFCNRPQ has the same complexity as the evaluating of FCNRPQ since we can simply evaluate a number (linear in the size of a FCNRPQ) of FCNRPQ in isolation \cite{2}.

6. EXPERIMENTS

All experiments are carried out on a machine with operating system WINDOWS 7 (professional version) having following specifications like CPU with four cores of 3.30GHz, 4GB memory and 450 GB storage. MySQL is used as relational database tool. Our code is an extension of RPL \cite{36} which evaluates RPQs on RDF graphs \cite{20}. Firstly we construct a relational database as in Figure 5.

![Figure 5: Relational database.](image)

We assessed our federated path queries on relational databases and a real RDF data set of total size of 14000 lines. It provides the information about a map in longitude and latitude points for different locations as in Figure 6.

![Figure 6: RDF Graph.](image)

Four federated queries are planned for experiments these are as followed:

**Query 1.**
On specific Date, at what location passengers get on vehicle and get off.

Let $q_1$ be an FPQ query defined as follows:

$$q_1(x, y) = (x, \exp_1(x, y)) \land R(\bar{u}),$$

where

- $\exp_1 : \next^{-1} :: \text{lon} / [\next :: \text{tag} / \edge :: \text{tourism}] / [\self :: \text{tourism}] / [\next :: \text{lat}]$;
- $R(\bar{u}) := \text{Orders}(\text{Date}, x, y)$.

The $\exp_1$ is to query the location of the road attraction points in the RDF graph. The $R(\bar{u})$ is to query the relational database which points that the passengers have gone through the order. Finally, by joining the parts we get $Q_1$.

**Query 2.**
On specific Date, Did the passengers visit a tourist attraction place on a map?

Let $q_2$ be an FPQ query defined as follows:

$$q_2(x, y) = (x, \exp_2(x, y)) \land R(\bar{u}),$$

where

- $\exp_2 : \next^{-1} :: \text{lat}$;
- $\exp_3 : \next^{-1} :: \text{lat}$;
- $\exp_4 : \self :: \next^{-1} :: \text{ref} / \next :: \text{lat}$;
- $R(\bar{u}) := \text{Orders}(\text{Date}, x, y)$.

The $\exp_2$ is to query the latitude and longitude of the tourist attraction points in the map. The $R(\bar{u})$ is to query the relational database which points that the passengers have gone through the order. Finally, by joining the parts we get $Q_2$.

**Query 3.**
On specific and unique Date, on which location of the road passengers get down from the taxi.

Let $q_3$ be an FPQ query defined as follows:

$$q_3(x, y) = (x, \exp_1(x, y)) \land (x, \exp_3(x, y)) \land R(\bar{u}),$$

where

- $\exp_1 : \next^{-1} :: \text{lon} / [\next :: \text{lat}]$;
- $\exp_2 : \next^{-1} :: \text{lat}$;
- $\exp_3 : \next^{-1} :: \text{lat}$;
- $R(\bar{u}) := \text{Orders}(\text{Date}, x, y)$.

The $\exp_1$ is the same as above in $q_1$. The $\exp_3$ is to query the latitude and longitude of points on the road in RDF data set. Sometimes some points are not on the road. The $R(\bar{u})$ is to query that on exact date to which points that the passengers have placed the order. Finally, by joining three parts we have the result.

**Query 4.**
On specific Date, Can a passenger take a ride when No vehicle
### 7. CONCLUSIONS

We have proposed federated path queries to navigate through RDF graphs integrated with relational databases. Some investigation about some fundamental properties of those federated path queries. We prove that FPQ strictly expresses nested regular expression and we also give a complete Hasse diagram of fragments of FPQ. Finally, we show that the query evaluation of FPQ maintains the polynomial time data complexity and NP-complete combined complexity as the same as conjunctive first-order queries. These results provides a starting point for further research on expressiveness of federated path languages for heterogeneous databases such as RDF graphs integrated with relational databases. Besides, we show that federated path queries can be evaluated efficiently in our experiments.

There are a number of practical open problems like more complex queries on larger heterogeneous datasets of database, to formulate relationships between within heterogeneous RDFs and with heterogeneous relational databases in different scenarios ultimately toward an optimized query manager. In this paper, we restrict that RDF data does not contain blank nodes as the same treatment in nSPARQL. We have to admit that blank nodes do make RDF data more expressive since a blank node in RDF is taken as an existentially quantified variable. An interesting future work is to extend our proposed federated path queries for general RDF data with blank nodes by allowing path variables which are already valid in some extensions of SPARQL such as SPARQL-eR and CPSPARQL, which are popular in querying over general RDF data with blank nodes.

### 8. ACKNOWLEDGMENTS

This work is supported by the programs of the National Key Research and Development Program of China (2016YFB1000603), the National Natural Science Foundation of China (NSFC) (61502336), and the open funding project of Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education (K93-9-2016-05). Xiaowang Zhang is supported by Tianjin Thousand Young Talents Program.
9. REFERENCES

[1] S. Abiteboul, P. Buneman, and D. Suciu. Data on the Web: From relations to semistructured data and XML. Morgan Kaufmann, 2000.
[2] S. Abiteboul, R. Hull, and V. Vianu. Foundations of databases. Addison-Wesley, 1995.
[3] P. Alkhateeb and J. Euzenat. Constrained regular expressions for answering RDF-path queries modulo RDF8. Inter. J. Web Infor. Sys., 10(1):24–50, 2014.
[4] R. Angles and C. Gutierrez. Survey of graph database models. ACM Comput. Surv., 40(1):article 1, 2008.
[5] P. Barceló. Querying graph databases. In: Proc. of PODS’13, pp. 175–188, 2013.
[6] M. Benedikt, W. Fan, and G. M. Kuper. Structural properties of XPath fragments. Theor. Comput. Sci., 336(1):3–31, 2005.
[7] S. Bischof, C. Martin, A. Polleres, and P. Schneider. Collecting, integrating, enriching and republishing open city data as linked data, In: Proc. of ISWC’15, pp.57–75, 2015.
[8] S. Böttcher, S. Link, and L. Zhang. Pulling conjunctive query equivalence out of the bag. In: Proc. of CIKM’14, pp.41–50, 2014.
[9] C. Buil Aranda, M. Arenas, Ó. Corcho, and A. Polleres. Federating queries in SPARQL 1.1: Syntax, semantics and evaluation. J. Web Sem., 18(1): 1–17, 2013.
[10] V. Fionda, G. Pirrò, and M. P. Consens. Extended property paths: Writing more SPARQL queries in a succinct way. In: Proc. of AAAI’15, pp.102–108, 2015.
[11] Guo Y., Pan Z., Heflin J.(2015). LUBM: A benchmark for OWL knowledge base systems. J. Web Sem., 3(2):158–182.
[12] G. H. L. Fletcher, M. Gyssens, D. Leinders, D. Surinx, J. V. den Bussche, D. V. Gucht, S. Vansummeren, and Y. Wu. Relative expressive power of navigational querying on graphs. Inf. Sci., 298:390–406, 2015.
[13] J. Hayes and C. Gutiérrez. Bipartite graphs as intermediate model for RDF. In: Proc. of ISWC’04, pp. 47–61, 2004.
[14] J. Hellings. Conjunctive context-free path queries. In: Proc. of ICDT’14, pp.119–130, 2014.
[15] J. Hellings, G. H. L. Fletcher, H. J. Havercort. Efficient external-memory bisimulation on DAGs. In: Proc. of SIGMOD’12, pp.553–564, 2012.
[16] J. Hellings, B. Kuippers, J. Van den Bussche, and X. Zhang. Walk logic as a framework for path query languages on graph databases. In: Proc. of ICDT’13, pp.117–128, 2013.
[17] A. Hogan, M. Arenas, A. Mallea, and A. Polleres. Everything you always wanted to know about blank nodes. J. Web Sem., 27:42–69, 2014.
[18] J. Hopcroft and J. Ullman. Introduction to automata theory, languages, and computation. Addison-Wesley, 1979.
[19] K. Kochut and M. Janik. SPARQleR: Extended SPARQL for semantic association discovery. In: Proc. of ESWC’07, pp.145–159, 2007.
[20] A. Koschmieder and U. Leser. Regular path queries on large graphs. Scientific and Statistical Database Management, Springer Berlin Heidelberg, 2012:177-194.
[21] M. Lange. Model checking propositional dynamic logic with all extras. J. Applied Logic, 4(1):39–49, 2006.
[22] L. Libkin, J. L. Reutter, and D. Vrgoc. Trial for RDF: Adapting graph query languages for RDF data. In Proc. of PODS’13, pp. 201–212, 2013.
[23] P. Linz. An Introduction to formal languages and automata (The fifth edition). Jones & Bartlett Publishers, 2012.
[24] K. Losemann and W. Martens. The complexity of regular expressions and property paths in SPARQL. ACM Trans. Database Syst., 38(4):24, 2013.
[25] J. L. Reutter, M. Romero, and M.Y. Vardi. Regular queries on graph databases. In: Proc. of ICDT’15, pp.177–194, 2015.
[26] M. Marx and M. de Rijke. Semantic characterizations of navigational XPath. SIGMOD Record, 34(2):41–46, 2005.
[27] M. Olson and U. Ogbuij. The Versa specification. October 2001.
[28] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. ACM Trans. Database Syst., 34(3):article 16, 2009.
[29] J. Pérez, M. Arenas, and C. Gutierrez. nSPARQL: A navigational language for RDF. J. Web Sem., 8(4):255–270, 2010.
[30] A. Polleres and J. P. Wallner. On the relation between SPARQL1.1 and answer set programming. J. Applied Non-Classical Logics, 23(1–2):159–212, 2013.
[31] RDF primer. W3C Recommendation, Feb. 2004.
[32] SPARQL query language for RDF. W3C Recommendation, Jan. 2008.
[33] SPARQL 1.1 query language. W3C Recommendation, Mar. 2013.
[34] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In: Proc. of SIGMOD’08, pp.567–580, 2008.
[35] E. V. Kostylev, J. L. Reutter, M. Romero, and D. Vrgoc. SPARQL with property paths. In: Proc. of ISWC’15, LNCS 9366, pp.3–18, 2015.
[36] H. Zauner, B. Linse, T. Furche, and F. Bry. A RPL through RDF: Expressive navigation in RDF graphs. In: Proc. of RR’10, pp.251–257, 2010.
[37] X. Zhang, Z. Feng, X. Wang, G. Rao, and W. Wu. Context-free path queries on RDF graphs. arXiv:1506.00743, 2016.
[38] X. Zhang and J. V. den Bussche. On the primitivity of operators in SPARQL. Inf. Process. Lett., 114(9): 480–485, 2014.
[39] X. Zhang and J. V. den Bussche. On the power of SPARQL in expressing navigational queries. Comput. J., 58(11): 2841–2851, 2015.
[40] X. Zhang and J. V. den Bussche. On the satisfiability problem for SPARQL patterns. J. Artif. Intell. Res., accepted, 2016.
Appendix: Proofs

Proof of Lemma 2

By induction on the structure of $e$.

- If $e$ is of the form $\text{axis}$ or $\text{axis}^{-1}$ and $(a,b) \in \{e\}_G$ for some $p$-RDF graph $G$ and some pair $(a,b)$ with $a,b \in \text{const}(G)$ then let consider seven cases of $\text{axis}$ as follows:

  - If $\text{axis}$ is self then $G \neq \emptyset$. Let $H = \{(a,b) \mid (a,b) \in G\}$ be a strongly acyclic $p$-RDF graph. $\{e\}_H \neq \emptyset$.
  - If $\text{axis}$ is next then $(a,b) \in G$. Let $H = \{(a,b)\}$ be a strongly acyclic $p$-RDF graph. $\{e\}_H \neq \emptyset$.
  - If $\text{axis}$ is next $^{-1}$ then $(b,a) \in G$. Let $H = \{(b,a)\}$ be a strongly acyclic $p$-RDF graph. $\{e\}_H \neq \emptyset$.
  - If $\text{axis}$ is edge then $(a,b,c) \in G$ (in this case $b = p$). Let $H = \{(a,b,c)\}$ be a strongly acyclic $p$-RDF graph. $\{e\}_H \neq \emptyset$.
  - If $\text{axis}$ is edge $^{-1}$ then $(b,a,c) \in G$ (in this case $a = p$). Let $H = \{(b,a,c)\}$ be a strongly acyclic $p$-RDF graph. $\{e\}_H \neq \emptyset$.
  - If $\text{axis}$ is node then $(c,a,b) \in G$ (in this case $a = p$). Let $H = \{(c,a,b)\}$ be a strongly acyclic $p$-RDF graph. $\{e\}_H \neq \emptyset$.

- If $e$ is of the form $e_1 e_2$ then this claim readily holds by induction.

- If $e$ is of the form $e_1 e_2$ and $(a,b) \in \{e\}_G$ for some $p$-RDF graph $G$ then there exists some $e \in \text{const}(G)$ such that $(a,c) \in \{e\}_G$. By induction, let $H_1$ be a strongly acyclic $p$-RDF graph. $(c,b) \in \{e\}_H$. Therefore $G_1$ is desired.

- If $e$ is of the form next :: $[e_1]_G$ (in this case $a = b$) and there exists some $e \in \text{const}(G)$ such that $(c,b) \in \{e\}_G$. By induction, let $H_1$ be a strongly acyclic $p$-RDF graph. $(c,b) \in \{e\}_H$. Therefore $G_1$ is desired.

- If $e$ is of the form next $^{-1}$ :: $[e_1]_G$ then $(b,a) \in G$ and there exists some $e \in \text{const}(G)$ such that $(c,b) \in \{e\}_G$. By induction, let $H_1$ be a strongly acyclic $p$-RDF graph. $(c,b) \in \{e\}_H$. Therefore $G_1$ is desired.

Therefore, there always exists some strongly acyclic $p$-RDF graph $H$ such that $(a,b) \in \{e\}_H$.

Proof of Proposition 3

Consider an $\text{nre}(\cdot)$-expression $e$ of the form $\text{next} \cap \text{next}$. Suppose, for the sake of contradiction, that $e$ is expressible as $e'$ for some $\text{nre}$-expression $e'$. Moreover, we can assume that $e'$ is consistent-free. (Otherwise, assume that $e$ occurs in $e$, let us consider an RDF graph $G = \{(a,b,p,d) \mid (a,p,b,d) \in G\}$ without $c$ and such that $\{e\}_G \neq \emptyset$. Since $e$ is expressible as $e'$, $\{e\}_G \neq \emptyset$. In other words, the constant $c$ can be removed from $e'$.

Consider a $p$-RDF graph $G = \{(a,b,p,d) \mid (a,p,b,d) \in G\}$, $(a,c) \in \{e\}_G$. Since $e$ is expressible as $e'$, we have $(a,c) \in \{e'\}_G$. By Lemma 2 there exists some strongly acyclic $p$-RDF graph $H$ such that $(a,c) \in \{e'\}_H$. Therefore, $H$ is desired.

Claim 16. For any $p$-RDF graph $G$, if $\{e\}_G \neq \emptyset$ then $\text{index}(G)$ is not strongly acyclic.
Assume that \((a, c) \in \mathcal{E} \mathcal{G}\). Then \(G\) must contain some subgraph \(G' = \{(a, p, b), (b, p, c), (a, p, c)\}\). However, \(\text{index}(G)\) is not strongly acyclic. Therefore, \(\text{index}(G)\) is not strongly acyclic since \(G' \subseteq G\).

Proof of Theorem 4

Let \(G\) be an RDFS graph. Consider a CNRPQ \(q(\exists x, ?y) = \exists (x, e_1, ?y) \land (x, e_2, ?y)\). Suppose, for the sake of contradiction, that \(q(\exists x, ?y)\) is expressible as \(q'(\exists x, ?y)\) for some NRQP \(q' \neq q\). Without loss of generality, we assume that \(q(\exists x, ?y) = (\exists y, e, ?y)\) where \(e\) is an nre-expression. Since \(q(\exists x, ?y)\) is expressible as \(q'(\exists x, ?y)\), we have that \(\mathcal{G}(q(\exists x, ?y)) = \mathcal{G}(q'(\exists x, ?y))\). That is, \(\{x, e, ?y\}\) is expressible as \(a\), however, we have arrived a contradiction.

Proof of Lemma 5

By induction on the structure of \(q\).

- If \(q(\exists x, ?y) = (\exists y, e, ?y)\) where \(e\) is an nre-expression then it follows definitions.

- If \(q(\exists x, ?y)\) is of the form \(\text{R}(\exists x, ?y)\) then, by definition, we conclude that \(\mathcal{G}(q(\exists x, ?y)) = \mathcal{G}(\text{R}(\exists x, ?y))\).

Consider an FCNRQP \(q(\exists x, ?y) = \exists (x, e_1, ?y) \land (x, e_2, ?y)\). Suppose, for the sake of contradiction, that \(q(\exists x, ?y)\) is expressible as \(q'(\exists x, ?y)\) for some CNRPQ \(q'\). Let \(D = (G, D)\) be a heterogeneous database where \(G = \{(a, a, a)\}\) where \(a\) does not occur in \(q(\exists x, ?y)\) and \(D = \{D\}\) with relation \(D = \{(a, b)\}\). We have \(\mathcal{G}(q(\exists x, ?y))_D = \emptyset\). By Lemma 5 \([q(\exists x, ?y)]_G\) is not empty, however, we have arrived a contradiction.

Proof of Lemma 7

By induction on the structure of \(q\).

- If \(q(\exists x, ?y)\) is of the form \(R(w_1, \ldots, w_m)\) then \([q(\exists x, ?y)]_D\) is empty.

- If \(q(\exists x, ?y)\) is of the form \(\exists (x, e, ?y)\) where \(e\) is an nre-expression then it follows definitions.

- If \(q(\exists x, ?y)\) is of the form \(\text{R}(\exists x, ?y)\) then, by definition, we conclude that \(\mathcal{G}(q(\exists x, ?y)) = \mathcal{G}(\text{R}(\exists x, ?y))\).

Therefore, \([q(\exists x, ?y)]_D\) contains at most one mapping by induction.

Proof of Theorem 8

Consider an FCNRQP \(q(\exists x, ?y) = (\exists y, e_1, ?y) \lor (\exists y, e_2, ?y)\). Suppose, for the sake of contradiction, that \(q(\exists x, ?y)\) is expressible as \(q'(\exists x, ?y)\) for some FNCRQ \(q'\). We have \([q(\exists x, ?y)]_D = \{(x \rightarrow a, y \rightarrow b), (x \rightarrow b, y \rightarrow a)\}\). By Lemma 7 \([q(\exists x, ?y)]_G\) contains at most one mapping, however, we have arrived a contradiction.

Proof of Proposition 12

Consider an RQ \(q(\exists x, ?y) = R(w_1, \ldots, w_m)\). Clearly, there exists no RSPARQL pattern \(P\) such that \(P\) expresses \(q(\exists x, ?y)\).

Proof of Proposition 13

Let \(q(u, v)\) be an RQ \([q(u, v)]_G = [P]_G\) by induction on the structure of \(q\).

- If \(q(u, v)\) is an RQ \([q(u, v)]_G = [P]_G\) for any RDF graph \((G, D)\).

- If \(q(u, v)\) is of the form \(\text{R}(u, v)\) then, by definition, we conclude that \(q(u, v)\) is expressible as \(q'(u, v)\) for any RDF graph \((G, D)\).

Finally, if \(q(u, v)\) is of the form \(\text{R}(u, v)\) then, by definition, we conclude that \(q(u, v)\) is expressible as \(q'(u, v)\) for any RDF graph \((G, D)\).