On the Primitivity of SPARQL 1.1 Operators

Xiaowang Zhang
School of Computer Science and Technology,Tianjin University, Tianjin 300350, China
Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300350, China
Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing, China
xiaowangzhang@tju.edu.cn

ABSTRACT

The paper studies the primitivity of the eleven basic operators used in the SPARQL 1.1 query language. This paper shows that the six operators BIND, FILTER, GRAPH, property path, SELECT, and VALUES are primitive while the left five operators AND, EXISTS, MINUS, OPT, and UNION, are not primitive. It is surprised that OPT and UNION which are primitive in SPARQL 1.0 become no longer primitive in SPARQL 1.1.

Keywords

RDF databases; SPARQL 1.1; primitive operator; expressive power

1. INTRODUCTION

As a recurring interest topic in the classical topic of graph databases [1], the primitivity of an operator is to determine whether the operator can be expressed in terms of the other operators. The standard query language for RDF (Resource Description Framework) data, a popular data model for information in the Web, is SPARQL 1.1 [4] by extending SPARQL 1.0 [3] with important features such as negation, subqueries, aggregation, and regular expressions, which those features will enrich the ability to represent more expressive queries [2, 6].

Zhang and Van den Bussche [5] investigated that, among five SPARQL 1.0 operators: AND, UNION, OPT, FILTER, and SELECT, only AND is not primitive where AND can be expressible by OPT and FILTER.

The main goal of this paper is to investigate the primitivity of the eleven basic operators in the SPARQL 1.1 query language. We show that the six operators BIND, FILTER, GRAPH, property path, SELECT, and VALUES are primitive while the left five operators AND, EXISTS, MINUS, OPT, and UNION, are not primitive.

This paper is further organized as follows. In the next section, we introduce syntax and semantics of SPARQL 1.1 operators. Section 3 shows the five non-primitive operators and Section 4 shows the left six primitive operators.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3054260

2. RDF AND SPARQL 1.1

In this section we recall SPARQL1.1 operators, closely following the SPARQL formalization in [4].

RDF graphs.

Let I, B, and L be infinite sets of *IRIs*, *blank nodes* and *literals*, respectively.

A triple $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an *RDF triple*. An *RDF graph* is a finite set of RDF triples.

A *dataset* DS is of the form $(G, \{(g_1, G_1), \ldots, (g_k, G_k)\})$ where $G, G_1, \ldots, G_k$ are RDF graphs and $g_1, \ldots, g_k$ are IRIs. We call G the *default graph* and $(g_1, G_1), \ldots, (g_k, G_k)$ *named graphs*. Let $\text{named}(DS) = \{g_1, \ldots, g_n\}$ and $DS(def) = G$ and $DS(g_i) = G_i$ for $i = 1, \ldots, k$.

SPARQL 1.1 operators.

SPARQL 1.1 patterns are constructed by using triple patterns (possibly adopting property paths) and operators: AND, OPT, FILTER, UNION, GRAPH, SELECT, EXISTS, BIND, MINUS, and VALUES, where SELECT (nested operation is not allowed in SPARQL 1.0), EXISTS, MINUS, BIND, and VALUES are newly added in SPARQL 1.1. The semantics of patterns is defined in terms of sets of so-called *mappings*, which are simply total functions $\mu\colon S \to U$ on some finite set S of variables. The semantics is based on a three-valued logic with truth values *true*, *false*, and *error* and the semantics in this paper is set-based while the semantics is bag-based in the practical applications [3].

3. EXPRESSIVITY OF OPERATORS

Let us abbreviate the operator AND by $\mathcal{A}$; BIND by $\mathcal{B}$; EXISTS by $\mathcal{E}$; FILTER by $\mathcal{F}$; MINUS by $\mathcal{M}$; GRAPH by $\mathcal{G}$; OPT by $\mathcal{O}$; property path by $\mathcal{P}$; subqueries by $\mathcal{S}$; UNION by $\mathcal{U}$; and VALUES by $\mathcal{V}$. We can denote any fragment of SPARQL 1.1, where only a subset of those operators is available, by the letter word formed by the operators that are available in the fragment.

An operator can not contribute the expressivity of a fragment if it is expressible by the fragment. Formally, we should define what we mean when we say that some operator $\mathcal{X}$ is "expressible" in some fragment $\mathcal{W}$. We will simply take this to mean here that *for every pattern P in the fragment $\mathcal{W}\mathcal{X}$* (i.e., adding $\mathcal{X}$ to $\mathcal{W}$) *there exists a pattern Q in the given fragment $\mathcal{W}$ such that for any graph G, we have*

$[\![P]\!]_G = [\![Q]\!]_G$ *[5]. In this sense, we say that P and Q are* equivalent*, denoted by $P \equiv Q$.*

We know that EXISTS is already expressible in $\mathcal{AFS}$ [2] and MINUS is already expressible in $\mathcal{OFS}$ [5] as follows:

$$P \text{ FILTER EXISTS}(Q) \equiv \text{SELECT}_{\text{var}(P)}(P \text{ AND } Q).$$

$$P \text{ MINUS } Q \equiv \text{SELECT}_{\text{var}(P)}(((P \text{ OPT } (?x, ?y, ?z))\text{OPT} \\ (Q \text{ OPT } (?x, ?y, ?u))) \text{ FILTER} \neg \text{bound}(?u)),$$

where $?x, ?y, ?z, ?u$ are fresh variables.

Firstly, we can express OPT in $\mathcal{AMSU}$.

Proposition 1. *Let P and Q be two patterns.*

$$P \text{ OPT } Q \equiv (P \text{ AND } Q) \text{ UNION SELECT}_{\text{var}(P)}(P \text{AND} \\ (?x, ?y, ?z) \text{ MINUS } (Q \text{ AND } (?x, ?y, ?z))$$

where $?x, ?y, ?z$ are three fresh variables.

Finally, we show that UNION is also expressible in $\mathcal{AOSV}$.

Proposition 2. *let P and Q be two patterns, we consider a pattern Q' in $\mathcal{AOSV}$ s.t. P UNION $Q \equiv Q'$ as follows:*

$$Q' = \text{SELECT}_S((((P_1 \text{ OPT } P_2) \text{ OPT } P_3) \text{ OPT } P_4) \text{ AND } P_5)$$

where $?x, ?y, ?z$ are fresh variables and a, b, c, d are fresh constants; and

$$S = \text{var}(P) \cup \text{var}(Q)$$
$$P_1 = (\text{VALUES}\,(?x)\,\{(a), (b)\})$$
$$P_2 = (P \text{ OPT } (\text{VALUES}\,(?x, ?y)\,\{(a, c)\})$$
$$P_3 = (Q \text{ OPT } (\text{VALUES}\,(?x, ?y)\,\{(b, c)\})$$
$$P_4 = (\text{VALUES}\,(?y)\,\{(d)\})$$
$$P_5 = (\text{VALUES}\,(?y)\,\{(c)\})$$

4. PRIMITIVITY OF OPERATORS

An operator $\mathcal{X}$ is "primitive" if $\mathcal{X}$ cannot be expressible by other operators.

Clearly, based on the discussions in Section 3, the five operators, namely, AND, EXISTS, MINUS, OPT, and UNION, are not primitive since AND is not primitive [5].

In the rest of this paper, we will discuss the primitivity of the left six operators.

As we well known, the transitivity is not expressible in first-order logic and SPARQL 1.0 has the same expressivity of first-order logic [3]. It is clear that path property is not expressible in SPARQL1.0. Then property path is primitive since other newly added operators are still expressible in first-order logic [4].

To show the primitivity of BIND, we need a lemma.

Lemma 3. *Let P be a* BIND*-free pattern and DS a dataset. For each mapping $\mu \in [\![P]\!]_G$, the image of μ is in $const(DS) \cup const(P)$.*

Proposition 4. BIND *is primitive.*

Proof (Sketch). Consider the following pattern: $P = (?x, r, ?y)\,\text{BIND CONCAT}(?x?y)\text{ AS }?z$ and the dataset $DS = (\{(a, r, b)\})$ where $\{a, b, ab\} \cap const(Q) = \emptyset$. □

Let us now turn to the question of primitivity of FILTER. Analogously, we can conclude the following lemma [5].

Lemma 5. *Let $DS = (G)$ where G is the complete graph on two constants $a, b \in I$ (i.e., $G = \{a, b\} \times \{a, b\} \times \{a, b\}$). Let P be any* FILTER*-free pattern with $\{a, b\} \cap const(P) = \emptyset$. If there exists some mappings $\mu \in [\![P]\!]_{DS}$ and $M \subseteq \text{dom}(\mu)$ such that $\mu(?x) \in \{a, b\}$ for all $?x \in M$ and $\mu(?y) \notin \{a, b\}$ for all $?y \in \text{dom}(\mu) - M$ then for all mapping $\mu' \colon M \to \{a, b\}$, we can conclude that $\mu' \cup \mu|_{\text{dom}(\mu) - M} \in [\![P]\!]_{DS}$.*

Proposition 6. FILTER *is primitive.*

Proof (Sketch). Consider the following pattern: $P = (?x, ?y, ?z)\,\text{FILTER}\,?x\ = ?y$ and DS be the dataset from Lemma 5. In [5], by Lemma 5, we can conclude that $\mu = (?x \to a, ?y \to b, ?z \to a)$ is a mapping in $[\![P]\!]_{DS}$. Therefore, we have arrived at a contradiction. □

Proposition 7. GRAPH *is primitive.*

Proof (Sketch). Consider the following pattern: $P = \text{GRAPH}\,c\,(?x, ?y, ?z)$ and the dataset $DS = (\emptyset, G)$ where $G = \{(c, c, c)\}$ with $c \notin const(Q)$. □

Finally, we will show that SELECT is primitive.

Lemma 8. *Let $DS = (G)$ be a dataset. For any* SELECT*-free pattern P, if $const(P) \cap const(G) = \emptyset$ then $[\![P]\!]_G$ does not contain the empty mapping $\mu_\emptyset$ (i.e., $\text{dom}(\mu_\emptyset) = \emptyset$).*

Proposition 9. SELECT *is primitive.*

Proof (Sketch). Consider the following pattern: $P = \text{SELECT}_\emptyset(?x, ?y, ?z)$ and a dataset $DS = (G)$ where $G = \{(a, a, a)\}$ with $a \notin const(Q) = \emptyset$. □

Proposition 10. VALUES *is primitive.*

Proof (Sketch). Consider the following pattern: $P = (\text{VALUES}\,(?x), \{(c)\})$ and the empty dataset $DS_\emptyset$. □

Finally, we can conclude the most important result.

Theorem 11. *Only operators* BIND*,* FILTER*,* GRAPH*, property path,* SELECT*, and* VALUES *are primitive.*

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2016YFB1000603), the National Natural Science Foundation of China (61502336), the Key Technology Research and Development Program of Tianjin (16YFZCGX00210), and the open funding project of Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education (K93-9-2016-05).

5. REFERENCES

[1] R. Angles and C. Gutierrez. Survey of graph database models, ACM Comput. Surv., 40(1)(2008): article 1.

[2] R. Kontchakov and E. Kostylev On expressibility of non-monotone operators in SPARQL In: *Proc. of AAAI'16*, pp. 369–379.

[3] J. Pérez, M. Arenas, and C. Gutierrez, Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3)(2009): article 16.

[4] SPARQL 1.1 query language, *W3C Recommendation*, March 2013.

[5] X. Zhang and J. Van den Bussche, On the primitivity of operators in SPARQL, *Inf. Process. Lett.*, 114(9):480–485, 2014.

[6] X. Zhang, J. Van den Bussche, and F. Picalausa, On the satisfiability problem for SPARQL patterns, *J. Artif. Intell. Res.*, 56: 403-428, 2016.