On the Expressivity of ASK Queries in SPARQL

Xiaowang Zhang,[1,4] **Jan Van den Bussche,**[2] **Kewen Wang,**[3]
Heng Zhang,[1] **Xuanxing Yang,**[1,4] **Zhiyong Feng**[1]
[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Faculty of Sciences, Hasselt University, Hasselt, Belgium
[3]School of Information and Communication Technology, Griffith University, Brisbane, Australia
[4]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China

Abstract

As a major query type in SPARQL, ASK queries are boolean queries and have found applications in several domains such as semantic SPARQL optimization. This paper is a first systematic study of the relative expressive power of various fragments of ASK queries in SPARQL. Among many new results, a surprising one is that the operator UNION is redundant for ASK queries. The results in this paper as a whole paint a rich picture for the expressivity of fragments of ASK queries with the four basic operators of SPARQL 1.0 possibly together with a negation. The work in this paper provides a guideline for future SPARQL query optimization and implementation.

Introduction

The Resource Description Framework (RDF) (Cyganiak, Wood, and Lanthaler 2014), a popular data model for information on the Web, represents information in the form of directed labelled graphs called RDF graphs (e.g., YAGO (Hoffart et al. 2013)), as a currently popular graph database (Angles and Gutiérrez 2008b). The standard query language for RDF data is SPARQL (Prud'hommeaux and Seaborne 2008) with its latest version SPARQL 1.1 (Harris and Seaborne 2013) by extending essential features such as negations, subqueries, regular expressions and aggregation. A SPARQL query is usually expressed in terms of certain algebraic expressions called patterns.

As a major query type in SPARQL (Prud'hommeaux and Seaborne 2008), ASK queries are boolean queries (Gottlob, Leone, and Scarcello 2001) while a SELECT query extracts the set of all result mappings. The importance of boolean queries is well-known in databases (Abiteboul, Hull, and Vianu 1995). In fact, ASK queries have also been applied in semantic SPARQL optimisation (Schmidt, Meier, and Lausen 2010), representing nested queries (Angles and Gutiérrez 2010) and RDF data access from mobile devices with context-aware policies (Costabello, Villata, and Gandon 2012). The expressivity, as an important fundamental property for query optimization and implementation, is a recurring topic in the area of query languages (Angles and Gutiérrez 2008b; Chandra and Harel 1980; Wood 2012). The expressive power of SPARQL has been analyzed in its relationship to the relational algebra (Cyganiak 2005), SQL (Chebotko, Lu, and Fotouhi 2009), Datalog (Angles and Gutiérrez 2008a; Polleres 2007), OWL (Gottlob and Pieris 2015), and modal logics (Guido 2015). Also the relationship between expressivity, complexity and optimization of query evaluation has been studied (Pérez, Arenas, and Gutiérrez 2009; Arenas and Pérez 2011; Schmidt, Meier, and Lausen 2010; Letelier et al. 2013; Chekol 2016; Nikolaou and Koubarakis 2016). The internal expressivity of SPARQL has been investigated including regular expressions and property paths (Kostylev et al. 2015), assignment and aggregation (Kaminski, Kostylev, and Grau 2016; 2017), well-designed patterns (Pichler and Skritek 2014; Kaminski and Kostylev 2016), negation and non-monotone operators (Angles and Gutiérrez 2016; Kontchakov and Kostylev 2016), CONSTRUCT queries (Kostylev, Reutter, and Ugarte 2015), operator primitivity (Zhang and den Bussche 2014), navigational power (Zhang and den Bussche 2015), and pattern satisfiability problem (Zhang, den Bussche, and Picalausa 2016; 2016).

However, the expressivity of ASK queries has not been explicitly investigated yet, while some results on SELECT queries and query satisfiability carry forward to ASK queries. Informally, the problem of expressivity of ASK queries is to study the role of an operator in expressing ASK queries. Two SELECT queries are equivalent if they have the same answers for each RDF graph; two queries are equivalent for satisfiability if they have the same satisfiability. Thus, the following implications for the expressivity of ASK queries hold but not the vice versa in general: *Expressible for SELECT Query* $\Rightarrow$ *Expressible* for ASK queries $\Rightarrow$ *Expressible for query satisfiability*.

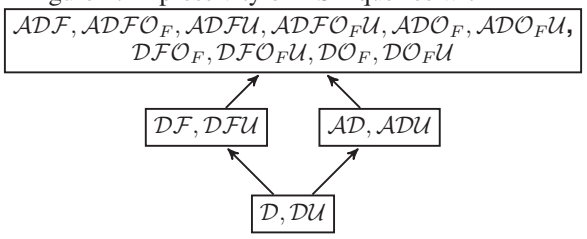
It implies that when an operator is not expressible for SELECT queries, it could be expressible for ASK queries. For instance, it is known that a SELECT query expressed by AND ($\mathcal{A}$), MINUS ($\mathcal{M}$) and UNION ($\mathcal{U}$) cannot be equivalently rewritten into a SELECT query expressed by only AND and MINUS (Zhang et al. 2018). It is interesting to know whether this will hold for ASK queries.

In this paper, we conduct a systematic study on the expressivity of ASK queries in SPARQL. Also, we show some

results on expressivity that are not reported in the literature while they hold for both SELECT queries and ASK queries. Our major contributions are briefly summarized as follows and specific results are depicted in Figure 1, Figure 2 and Figure 3, respectively. In these figures, a fragment of SPARQL is represented by a sequence of the operators. For instance, $\mathcal{ADO}_F$ denotes the fragment of ASK queries formed by three operators AND, DIFF and OPT_F. All fragments in the same rectangle box are equivalent to each other. If there is an arrow from a box B_1 to another box B_2, then every fragment in B_1 is expressible in a fragment in B_2.

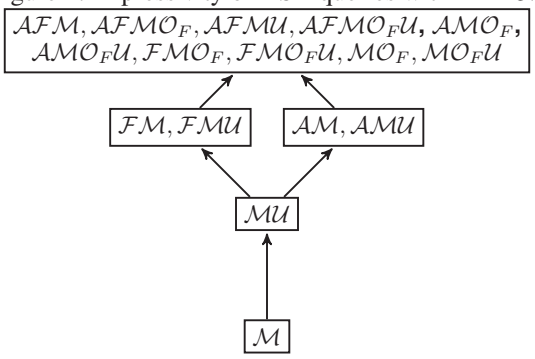
- We investigate the expressivity of 16 fragments of ASK queries with four basic SPARQL 1.0 operators together with DIFF (see Figure 1).

Figure 1: Expressivity of ASK queries with DIFF



- We investigate the expressivity of 16 fragments of ASK queries with four basic SPARQL 1.0 operators together with MINUS (see Figure 2).

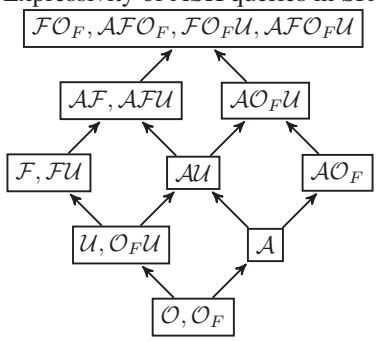
Figure 2: Expressivity of ASK queries with MINUS



- We investigate the expressivity of 16 fragments of ASK queries with all four basic operators of SPARQL 1.0 (see Figure 3). These results paint a complete picture for the expressivity of the 9 classes (see Figure 3).

This paper is further organised as follows. In the next section, we recall syntax and semantics of SPARQL patterns and ASK queries. Section 3 introduces ASK queries and ASK-expressivity. Section 4 discusses the expressivity of ASK queries with negations and Section 5 discusses the expressivity of ASK queries in SPARQL 1.0. Section 6 concludes our works.

Figure 3: Expressivity of ASK queries in SPARQL 1.0



SPARQL: Syntax and Semantics

In this section, we briefly recall some definitions and notations for the core SPARQL formalization in (Pérez, Arenas, and Gutiérrez 2009; Kontchakov and Kostylev 2016).

RDF graphs

Let U be the universe of *RDF terms* consisting of *IRIs* (i.e., I) and *literals* (i.e., L). An *RDF graph* (for short, graph) is modeled as a finite set of RDF triples $(s, p, o) \in U \times I \times U$. Following related work (Kontchakov and Kostylev 2016), the distinction between IRIs and literals, on the one hand, and *blank nodes* and RDF terms on the other hand, does not bring up fundamentally new problems about expressive power. Hence our model of RDF graph is adequate for the purpose of our investigation.

Besides the set U of constants, we also assume that V is an infinite set of *variables*, which is disjoint from U. A variable usually starts with a question mark "?" to distinguish it from a constant. We use u, v, w to denote elements that are either literals or variables; $?x$, $?y$, $?z$ to denote variables; a, b, c to denote constants.

Syntax of SPARQL

In SPARQL, a query is defined in terms of *patterns*. A triple (u, v, w) from $(U \cup V) \times (U \cup V) \times (U \cup V)$ is called a *triple pattern*. A *graph pattern* (for short, *pattern*) P is inductively defined as follows:

$$P := (u, v, w) \mid P_1 \text{ AND } P_2 \mid P_1 \text{ OPT}_F \ P_2 \mid P_1 \ \text{DIFF}_F \ P_2 \mid P_1 \text{ UNION } P_2 \mid P_1 \ \text{MINUS} \ P_2 \mid P_1 \text{ FILTER } F \mid \text{SELECT}_S(P_1).$$

A *filter condition* F (or just *filter*) is a formula constructed from (filter) atoms of the form $\mathrm{bnd}(?x)$, $?x = ?y$, and $?x = c$, using logical connectives $\wedge$, $\vee$, and $\neg$.

For a pattern P, $\mathrm{var}(P)$ and $\mathrm{con}(P)$ denote the set of all variables and the set of all constants in P respectively.

Semantics of SPARQL

Mappings The semantics of patterns is defined in terms of sets of so-called *solution mappings*. A solution mapping (simply, mapping) is a total function $\mu : S \to U$ on a finite set S of variables. The domain S of μ is denoted

$\mathrm{dom}(\mu)$. Two mappings μ_1 and μ_2 are *compatible*, denoted by $\mu_1 \sim \mu_2$, if they agree on the intersection of their domains. The *empty mapping* ($\mu_\emptyset$), whose domain is empty, is always compatible with any mapping.

Given a mapping μ and a filter F, the evaluation of F on μ, denoted by F^μ, is defined in terms of a two-valued logic. That is, $F^\mu = true$ if F is satisfied by μ and otherwise $F^\mu = false$ (Kontchakov and Kostylev 2016). The standard semantics of SPARQL is three-valued semantics (Pérez, Arenas, and Gutiérrez 2009; Prud'hommeaux and Seaborne 2008). Fortunately, the three-valued semantics of SPARQL can be simulated by the two-valued semantics of SPARQL if FILTER is allowed (Zhang, Meng, and Zou 2018).

SPARQL Algebra Let Ω, Ω_1, and Ω_2 be three sets of mappings and S be a set of variables. The SPARQL algebra for sets of mappings is composed of the operations of projection, selection, join, difference, left-join, union, and minus, respectively, defined as follows:

- $\pi_S(\Omega) = \{\mu|_{S\cap\mathrm{dom}(\mu)} \mid \mu \in \Omega\}$;
- $\sigma_F(\Omega) = \{\mu \in \Omega \mid F^\mu = true\}$;
- $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \mu_1 \sim \mu_2\}$;
- $$\Omega_1 \setminus_F \Omega_2 = \{\mu_1 \in \Omega_1 \mid \forall\, \mu_2 \in \Omega_2, \\ (\mu_1 \not\sim \mu_2) \vee (F^{\mu_1\cup\mu_2} \neq true)\};$$
- $\Omega_1 \cup \Omega_2 = \{\mu \mid \mu \in \Omega_1 \vee \mu \in \Omega_2\}$;
- $$\Omega_1 \ominus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \forall\, \mu_2 \in \Omega_2, \\ (\mu_1 \not\sim \mu_2) \vee (\mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2) = \emptyset)\};$$
- $\Omega_1 \leftmodels_F \Omega_2 = \sigma_F(\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus_F \Omega_2)$.

Note that we use $\setminus$ and $\leftmodels$ to denote $\setminus_\top$ and $\leftmodels_\top$ respectively. That is,

$$\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \forall\, \mu_2 \in \Omega_2, \mu_1 \not\sim \mu_2\}.$$

$$\Omega_1 \leftmodels \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2).$$

Clearly, the following equations hold:

$$\Omega_1 \setminus_F \Omega_2 = \Omega_1 \setminus \sigma_F(\Omega_1 \bowtie \Omega_2).$$

$$\Omega_1 \leftmodels_F \Omega_2 = \sigma_F(\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \sigma_F(\Omega_1 \bowtie \Omega_2)).$$

Semantics of SPARQL Given a graph G and a pattern P, the *semantics* $[\![P]\!]_G$ of P on G is defined by a set of mappings as follows:

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

  Here $\mu(c) = c$ for any $c \in U$.
- $[\![P_1 \ \mathrm{UNION}\ P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$;
- $[\![P_1 \ \mathrm{AND}\ P_2]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$;
- $[\![P_1 \ \mathrm{DIFF}_F\ P_2]\!]_G = [\![P_1]\!]_G \setminus_F [\![P_2]\!]_G$;
- $[\![P_1 \ \mathrm{MINUS}\ P_2]\!]_G = [\![P_1]\!]_G \ominus [\![P_2]\!]_G$;
- $[\![P_1 \ \mathrm{FILTER}\ F]\!]_G = \sigma_F([\![P_1]\!]_G)$;
- $[\![P_1 \ \mathrm{OPT}_F\ P_2]\!]_G = [\![P_1]\!]_G \leftmodels_F [\![P_2]\!]_G$;
- $[\![\mathrm{SELECT}_S(P_1)]\!]_G = \pi_S([\![P_1]\!]_G)$.

Clearly, the semantics of P_1 DIFF P_2 and P_1 OPT P_2 on G are equivalently defined as follows:

$$[\![P_1 \ \mathrm{DIFF}\ P_2]\!]_G = [\![P_1]\!]_G \setminus [\![P_2]\!]_G.$$

$$[\![P_1 \ \mathrm{OPT}\ P_2]\!]_G = [\![P_1]\!]_G \leftmodels [\![P_2]\!]_G.$$

Standard Equivalence: Two patterns P and P' are equivalent, denoted $P \equiv P'$, if $[\![P]\!]_G = [\![P']\!]_G$ for any graph G.

By the definition, we have the following lemma.

Lemma 1. *Let P_1 and P_2 be two patterns and F a filter. For any graph G, the following two equations hold:*

- $$P_1 \ \mathrm{DIFF}_F\ P_2 = P_1 \ \mathrm{DIFF}\ ((P_1 \ \mathrm{AND}\ P_2)\ \mathrm{FILTER}\ F);$$
- $$P_1 \ \mathrm{OPT}_F\ P_2 = ((P_1 \ \mathrm{AND}\ P_2)\ \mathrm{FILTER}\ F)\ \mathrm{UNION} \\ (P_1 \ \mathrm{DIFF}\ ((P_1 \ \mathrm{AND}\ P_2)\ \mathrm{FILTER}\ F).$$

By Lemma 1, DIFF_F and OPT_F are not necessary if AND, FILTER, UNION and DIFF are allowed.

ASK Queries and ASK-Expressivity

In this section, we define the equivalence of ASK queries and then formalise the expressivity problem of ASK queries.

ASK Query and Expressivity

An *ASK query* Q is of the form $\mathrm{ASK}(P)$, where P is a pattern. When there is no confusion, an ASK query is just called a query.

Semantically, given a graph G, the evaluation of $\mathrm{ASK}(P)^G$ is *true* if $[\![P]\!]_G \neq \emptyset$; and *false* otherwise.

Definition 2. *Two ASK queries Q_1 and Q_2 are equivalent, denoted $Q_1 \equiv_{\mathrm{ASK}} Q_2$, if for every graph G, $Q_1^G = true$ iff $Q_2^G = true$. If $Q_i = \mathrm{ASK}(P_i)$ for $i = 1, 2$ and $Q_1 \equiv_{\mathrm{ASK}} Q_2$, then we say that P_1 and P_2 are ASK-equivalent, denoted $P_1 \equiv_{\mathrm{ASK}} P_2$.*

We say that Q is *ASK-expressible*, or just *expressible*, in a fragment $\mathcal{W}$ if $Q \equiv_{\mathrm{ASK}} Q'$ for some query Q' in $\mathcal{W}$.

A fragment $\mathcal{W}_1$ is *ASK-expressible*, or just *expressible*, in another fragment $\mathcal{W}_2$, denoted by $\mathcal{W}_1 \preceq \mathcal{W}_2$, if every ASK query in $\mathcal{W}_1$ is expressible in $\mathcal{W}_2$. Otherwise, $\mathcal{W}_1 \not\preceq \mathcal{W}_2$. By $\mathcal{W}_1 \precneqq \mathcal{W}_2$, we mean $\mathcal{W}_1 \preceq \mathcal{W}_2$ and $\mathcal{W}_2 \not\preceq \mathcal{W}_1$.

An ASK query Q is *monotone* if for each pair of graphs G_1 and G_2 with $G_1 \subseteq G_2$, $Q^{G_1} = true$ implies $Q^{G_2} = true$; and *non-monotone* otherwise. A fragment $\mathcal{W}$ of ASK queries is *monotone* if all ASK queries in $\mathcal{W}$ are monotone; and *non-monotone* otherwise. An operator $\mathcal{X}$ is *monotone* if the fragment consisting of only $\mathcal{X}$ is monotone; and *non-monotone* otherwise. DIFF_F and MINUS are non-monotone, while AND, FILTER, OPT_F, and UNION are

monotone under ASK queries. Note that OPT_F is monotone in ASK queries while it is non-monotone in SELECT queries (Kontchakov and Kostylev 2016).

As pointed out in (Zhang, den Bussche, and Picalausa 2016), SELECT can be removed by a renaming of variables. Thus, we assume that ASK queries are SELECT-free.

Expressivity of ASK Queries with Negations

In this section, we discuss the expressivity of ASK queries with negations (DIFF_F, DIFF, and MINUS).

Expressivity of ASK queries with DIFF_F

It is easy to see that there are 16 fragments of SPARQL that are extensions of the fragment $\mathcal{AFOU}$ by adding DIFF_F. In this subsection, we investigate the expressive power of ASK queries in these fragments. A major result is that AND cannot be expressed by DIFF_F.

We first show that ASK queries with FILTER is expressible in ASK queries with DIFF_F.

Proposition 3. *Let P be a pattern and F be a filter. Then*

$$P \text{ FILTER } F \equiv P \text{ DIFF } (P \,\mathrm{DIFF}_F\ (?x, ?y, ?z)),$$

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

By definition, it follows that OPT_F is expressible by AND, DIFF_F and UNION while FILTER is expressible by only DIFF_F and UNION.

Proposition 4. *The following hold.*

- $\mathcal{AD}_F\mathcal{FO}_F\mathcal{U} \preceq \mathcal{AD}_F\mathcal{U}$;
- $\mathcal{D}_F\mathcal{FU} \preceq \mathcal{D}_F\mathcal{U}$.

The next result shows that UNION of ASK queries in $\mathcal{AD}_F\mathcal{U}$ and $\mathcal{D}_F\mathcal{U}$ is essentially redundant in terms of the expressivity of ASK queries.

A pattern P is in *UNION normal form* (UNF) if P is of the form

$$P_1 \text{ UNION} \cdots \text{UNION } P_m \tag{1}$$

where P_i is a UNION-free pattern ($i = 1, \ldots, m$).

Proposition 5. *If P is a UNF pattern in Equation (1), then*

$$P \equiv_{\mathrm{ASK}} (?x, ?y, ?z) \text{ DIFF } (((?x', ?y', ?z') \text{ DIFF } P_1) \cdots \text{DIFF } P_m) \tag{2}$$

where $?x, ?y, ?z, ?x', ?y', ?z'$ are fresh variables that do not appear in P.

In the above proof of Proposition 5, the idea is essentially based on the following statement.

Proposition 6. *Let Q, $P_1, \ldots, P_m$ are patterns. Then*

$$Q \text{ DIFF } (P_1 \text{ UNION } \cdots \text{ UNION } P_{m-1} \text{ UNION } P_m) \equiv_{\mathrm{ASK}} ((Q \text{ DIFF } P_1) \cdots \text{DIFF } P_{m-1}) \text{ DIFF } P_m.$$

Since each pattern in $\mathcal{ADU}$ is equivalent to a UNF pattern (Zhang et al. 2018), by Propositions 5 and 6, we have the following corollary.

Corollary 7. *The following hold.*

- $\mathcal{D}_F\mathcal{U} \preceq \mathcal{D}_F$;
- $\mathcal{AD}_F\mathcal{U} \preceq \mathcal{AD}_F$.

It is interesting to see whether $\mathcal{A}$ is expressible in $\mathcal{D}_F$. The answer is negative.

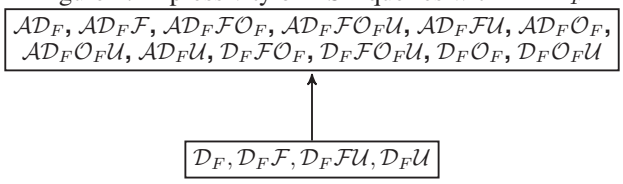
Proposition 8. $\mathcal{A} \not\preceq \mathcal{D}_F$.

Now, we can present our main theorem in this subsection.

Theorem 9. *The following statements hold for ASK queries in SPARQL (shown in Figure 4):*

1. *All fragments in $\mathbf{W}_1 = \{\mathcal{D}_F, \mathcal{D}_F\mathcal{F}, \mathcal{D}_F\mathcal{FU}, \mathcal{D}_F\mathcal{U}\}$ are equivalent to each other.*
2. *All fragments in $\mathbf{W}_2 = \{\mathcal{AD}_F, \mathcal{AD}_F\mathcal{F}, \mathcal{AD}_F\mathcal{FO}_F, \mathcal{AD}_F\mathcal{FO}_F\mathcal{U}, \mathcal{AD}_F\mathcal{FU}, \mathcal{AD}_F\mathcal{O}_F, \mathcal{AD}_F\mathcal{O}_F\mathcal{U}, \mathcal{AD}_F\mathcal{U}, \mathcal{D}_F\mathcal{FO}_F, \mathcal{D}_F\mathcal{FO}_F\mathcal{U}, \mathcal{D}_F\mathcal{O}_F, \mathcal{D}_F\mathcal{O}_F\mathcal{U}\}$ are equivalent to each other.*
3. *If $\mathcal{W}_1 \in \mathbf{W}_1$ and $\mathcal{W}_2 \in \mathbf{W}_2$, then $\mathcal{W}_1 \not\preceq \mathcal{W}_2$.*

Figure 4: Expressivity of ASK queries with DIFF_F



Expressivity of ASK queries with either DIFF or MINUS

In this subsection, we consider the 32 fragments of $\mathcal{AFO}_F\mathcal{U}$ with either DIFF or MINUS.

We show that DIFF_F is expressible in $\mathcal{MO}_F$ and $\mathcal{DO}_F$.

Proposition 10. *Let P_1 and P_2 be two patterns in $\mathcal{AD}_F$. Then P_1 DIFF_F $P_2 \equiv P$ MINUS $(?x, ?y, ?z)$ where $?x, ?y, ?z$ are fresh variables and $P = (P_1 \ \mathrm{OPT}_F\ (P_2 \text{ OPT } (?x, ?y, ?z)))$.*

It is shown that AND is expressible in $\mathcal{MO}_F$ (Zhang and den Bussche 2014). By Proposition 10, $\mathcal{AD}_F$ is expressible in $\mathcal{MO}_F$.

Now we wonder to know whether $\mathcal{AD}_F$ is expressible in $\mathcal{DO}_F$. The answer is positive.

Let P be a pattern and F a filter. $Filter(P, F)$ denotes the pattern P DIFF $P_{\neg F}$, where $P_{\neg F}$ is defined as

$$(P \,\mathrm{OPT}_{F \wedge ?x_1 = ?x_2}\ ((?x_1, ?y_1, ?z_1) \text{ OPT } (?x_2, ?y_2, ?z_2))) \text{ DIFF } Q, \tag{3}$$

$$Q = (((?x_1, ?y, ?z)\ \mathrm{OPT}_{?x_1 \neq ?x_2}\ (?x_2, ?u_1, ?v_1))\ \mathrm{OPT}_{?x_1 \neq ?x_2}\ (?u_2, ?x_2, ?v_2))\ \mathrm{OPT}_{?x_1 \neq ?x_2}\ (?u_3, ?v_3, ?x_2); \tag{4}$$

where $?y, ?z, ?x_1, ?x_2, ?x_3, ?y_1, ?y_2, ?z_1, ?z_2, ?u_i, ?v_i$ ($i = 1, 2, 3$) are fresh variables.

Proposition 11. *Let P be a pattern and F be a filter. For any graph G, if G contains at least two constants then:*

$$[\![P \text{ FILTER } F]\!]_G = [\![Filter(P, F)]\!]_G.$$

However, Proposition 11 does not hold if G consists of only one constant of the form $\{(c, c, c)\}$ (that is, G is a singleton complete graph). In this case, $[\![P \text{ DIFF } P_{\neg F}]\!]_G \neq \emptyset$ since $[\![P_F]\!]_G = \emptyset$ no matter what F is.

So, we consider all patterns in $\mathcal{ADFO}_F\mathcal{U}$ but graphs containing only one constant.

For the moment we consider the fragment $\mathcal{ADF}_0$ where all triple patterns are constant-free and all filters are atomic.

Given a pattern P in $\mathcal{ADF}_0$, its *mapping schema*, denoted by $\Gamma(P)$, is defined as a set of variables by

- $\Gamma((?u, ?v, ?w)) = \{?u, ?v, ?w\}$;
- $\Gamma(P_1 \text{ AND } P_2) = \Gamma(P_1) \cap \Gamma(P_2)$;
- $\Gamma(P_1 \text{ DIFF } P_2) = \Gamma(P_1)$;
- $\Gamma(P_1 \text{ FILTER } F) = \Gamma(P_1)$.

A pattern P in $\mathcal{ADF}_0$ is said to be *safe* if $\text{var}(F) \subseteq \Gamma(P_1)$ for every subpattern Q of the form P_1 FILTER F. We remark that this definition extends the definition of safe pattern in (Pérez, Arenas, and Gutiérrez 2009) for supporting negated patterns.

Firstly, we consider the safe fragment $\mathcal{ADF}_0^{\text{safe}}$ where all patterns are safe and all filters are atomic.

Let $?x_0$ be a fixed variable and P be a pattern in $\mathcal{ADF}_0^{\text{safe}}$. We define $\delta(P)$ as an $\mathcal{AD}$-pattern in an inductive way.

- $\delta(?u, ?v, ?w) = (?x_0, ?x_0, ?x_0)$;
- $\delta(P_1 \text{ AND } P_2) = \delta(P_1) \text{ AND } \delta(P_2)$;
- $\delta(P_1 \text{ DIFF } P_2) = \delta(P_1) \text{ DIFF } \delta(P_2)$;
- $\delta(P \text{ FILTER } \mathit{bound}(?y)) = \delta(P)$;
- $\delta(P \text{ FILTER } \neg \mathit{bound}(?y)) = \delta(P) \text{ DIFF } (?x_0, ?x_0, ?x_0)$;
- $\delta(P \text{ FILTER } ?y = ?z) = \delta(P)$;
- $\delta(P \text{ FILTER } ?y \neq ?z) = \delta(P) \text{ DIFF } (?x_0, ?x_0, ?x_0)$;
- $\delta(P \text{ FILTER } ?y = c) = \delta(P) \text{ AND } (c, c, c)$;
- $\delta(P \text{ FILTER } ?y \neq c) = \delta(P) \text{ DIFF } (c, c, c)$.

The definition of $\delta(P)$ can be extended in $\mathcal{ADFU}$.

Lemma 12. *Let P be a patten in $\mathcal{ADFO}_F\mathcal{U}$. For any RDF graph G of the form $\{(c, c, c)\}$ with $c \in U$, the following hold:*

$$[\![P]\!]_G \text{ is nonempty iff } [\![\delta(P)]\!]_G \text{ is nonempty.}$$

By Proposition 11 and Lemma 12, we can show that patterns in $\mathcal{ADFO}_F\mathcal{U}$ are expressible in $\mathcal{DO}_F$.

We first prove a proposition.

Given a pattern P in $\mathcal{ADFO}_F\mathcal{U}$, we define

$$P^* = P^F \text{ UNION } (\delta(P) \text{ AND } P_\top); \quad (5)$$

where P^F is a FILTER-free pattern by substituting $\mathit{Filter}(P', F)$ for all sub-patterns of the form P' FILTER F in P, $P_\top$ is defined by

$$P_\top = ((?x_3, ?y_3, ?z_3) \text{ OPT}_{?x_1=?x_2} ((?x_1, ?y_1, ?z_1) \text{ OPT } (?x_2, ?y_2, ?z_2))) \text{ DIFF } Q. \quad (6)$$

Here Q is defined in Equation (4) and $?x_i, ?y_i, ?z_i$ $(i = 1, 2, 3)$ are fresh variables.

Proposition 13. *Let P be a pattern in $\mathcal{ADFO}_F\mathcal{U}$ and the pattern P^* is defined as above. Then $P \equiv_{\text{ASK}} P^*$.*

Now, we show that $\mathcal{ADFO}_F\mathcal{U}$ is expressible in $\mathcal{DO}_F$.

Theorem 14. $\mathcal{ADFO}_F\mathcal{U} \preceq \mathcal{DO}_F$.

By Theorems 9, 13 and 14, we conclude the following.

Corollary 15. *The following hold.*

- $\mathcal{AFMO}_F\mathcal{U} \preceq \mathcal{MO}_F$.
- $\mathcal{ADFO}_F\mathcal{U} \preceq \mathcal{DO}_F$.

Next, we show that OPT_F is necessary to express FILTER when DIFF_F is absent. This is due to the fact that equality-constraints of FILTER are not expressible in any equality-free fragment (Casanovas, Dellunde, and Jansana 1996).

Since DIFF can be rewritten by MINUS syntactically in $\mathcal{ADMU}$, we have the following result.

Proposition 16. $\mathcal{ADMU} \preceq \mathcal{AMU}$.

In the proof of Proposition 16, in a pattern P in $\mathcal{ADMU}$, for any subpattern Q of the form P_1 DIFF P_2, we can rewrite it into P_1 MINUS $(P_1$ AND $P_2)$.

We will show that FILTER is not expressible in $\mathcal{ADMU}$. By Proposition 16, we consider only the fragment $\mathcal{AMU}$ instead of $\mathcal{ADMU}$.

It is in order to introduce the notion of *surjective strict homomorphism* for graphs.

Definition 17. *Let G and G' be two graphs and C be a set of constants. A* surjective strict C-homomorphism *from G to G' is a surjective function $h : \mathit{const}(G) \to \mathit{const}(G')$ such that $(s, p, o) \in G$ iff $(h(s), h(p), h(o)) \in G'$ for all constants $s, p, o \in \mathit{const}(G)$, and $h(c) = c$ for all $c \in C$.*

Lemma 18. *Let P be a pattern in $\mathcal{AMU}$ and $C = \text{con}(P)$. Let G and G' be two graphs such that there is a surjective strict C-homomorphism h from G onto G'. Then*

1. *$\mu \in [\![P]\!]_G$ iff $\mu \circ h \in [\![P]\!]_{G'}$;*
2. *If $\mu \in [\![P]\!]_{G'}$, then $\mu = \mu' \circ h$ for some $\mu' \in [\![P]\!]_G$.*

Note that OPT_F is needed for expressing FILTER.

By Lemma 18 and Proposition 16, we can show the following theorem.

Theorem 19. $\mathcal{F} \not\preceq \mathcal{ADMU}$.

Proof. Consider the pattern $P = (?x, p, ?y)$ FILTER $?x \neq ?y$ with $C = \{p\}$. Take $G = \{(a, p, a), (a, p, b), (b, p, a), (b, p, b)\}$ and $G' = \{(a, p, a)\}$. Let h be a function such that

$$h(a) = h(b) = a \text{ and } h(p) = p.$$

Clearly, h is a surjective strict C-homomorphism h from G to G'. It is not difficult to see that $[\![P]\!]_G \neq \emptyset$. On the other hand, by Lemma 18, $[\![P]\!]_{G'} = \emptyset$, a contradiction. □

By Theorem 19, we have $\mathcal{F} \not\preceq \mathcal{AMU}$ and $\mathcal{F} \not\preceq \mathcal{ADU}$.

We note that $\mathcal{DMU}$ is expressible in GF (Zhang et al. 2018) and cyclic join queries are not expressible in GF (Flum, Frick, and Grohe 2002; Gottlob, Grädel, and Veith 2002) while cyclic join queries can be expressed by AND in the proof of Proposition 8. Also, we have the following.

Proposition 20. $\mathcal{A} \not\preceq \mathcal{DMU}$.

By Proposition 20, we have that $\mathcal{A} \not\preceq \mathcal{MU}$ and $\mathcal{A} \not\preceq \mathcal{DU}$.

Next, we would like to know whether ASK queries in a fragment of $\mathcal{AFDU}$ are expressible in the corresponding fragment with UNION. The answer is positive.

To do so, we need an important lemma.

Lemma 21. *Let P be a pattern in UNF as Equation (1) and P' be the UNION-free pattern*

$$(?x, ?y, ?z) \text{ DIFF } (((?x', ?y', ?z') \text{ DIFF } (P_1)) \cdots \text{ DIFF } (P_m)),$$

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

Since each pattern in $\mathcal{AFDU}$ is equivalent to an $\mathcal{AFD}$ pattern in UNF, by Lemma 21, we conclude the following.

Proposition 22. *The following hold.*

- $\mathcal{ADU} \preceq \mathcal{AD}$;
- $\mathcal{FDU} \preceq \mathcal{FD}$;
- $\mathcal{DU} \preceq \mathcal{D}$.

Finally, we show that ASK queries in a fragment of $\mathcal{AFMU}$ are expressible in the corresponding fragment without UNION.

Proposition 23. $\mathcal{AMU} \preceq \mathcal{AM}$.

Let P be a pattern in $\mathcal{FMU}$. We use $\text{pos}(P)$ to denote the set of triple patterns in P as follows.

- $\text{pos}((u, v, w)) = \{(u, v, w)\}$;
- $\text{pos}(P_1 \text{ UNION } P_2) = \text{pos}(P_1) \cup \text{pos}(P_2)$;
- $\text{pos}(P_1 \text{ MINUS } P_2) = \text{pos}(P_1)$;
- $\text{pos}(P_1 \text{ FILTER } F) = \text{pos}(P_1)$.

We are ready to show that UNION is expressible by FILTER and MINUS.

Proposition 24. $\mathcal{FMU} \preceq \mathcal{FM}$.

Surprisingly, different from DIFF, MINUS cannot express UNION.

Proposition 25. $\mathcal{U} \not\preceq \mathcal{M}$.

Proof. (Sketch) Consider the pattern $Q = \text{ASK}(a, a, a) \text{ UNION } (b, b, b)$ where $a, b \in U$.

Claim 26. *There exists no pattern P in $\mathcal{M}$ such that for every RDF graph G, $Q^G = true$ iff $\text{ASK}(P)^G = true$.*

In order to prove this claim, consider three graphs:

- $G_1 = \{a, b\} \times \{a, b\} \times \{a, b\} \setminus \{(a, a, a), (b, b, b)\}$.
- $G_2 = \{a, b\} \times \{a, b\} \times \{a, b\} \setminus \{(a, a, a)\}$.
- $G_3 = \{a, b\} \times \{a, b\} \times \{a, b\} \setminus \{(b, b, b)\}$.

Note that $Q^{G_1} = false$ but $Q^{G_2} = true$ and $Q^{G_3} = true$. However, there exists no pattern P in $\mathcal{M}$ such that $P^{G_1} = false$ but $P^{G_2} = true$ and $P^{G_3} = true$.

Therefore, we can conclude that $\mathcal{U} \not\preceq \mathcal{M}$. □

Clearly, by Proposition 25, it is immediate to conclude that $\mathcal{MU} \not\preceq \mathcal{M}$.

In the proof of $\mathcal{U} \not\preceq \mathcal{M}$, we consider patterns with only constants. So we wonder whether a pattern in $\mathcal{U}$ is expressible in $\mathcal{M}$ if each of triple patterns in the given pattern contains a variable. Fortunately, the answer is positive.

Proposition 27. *Let P be a pattern in $\mathcal{MU}$. If $\text{pos}(P)$ contains no variable-free triple patterns, then $\text{ASK}(P)$ can be expressed in $\mathcal{M}$.*

Expressivity of ASK queries in SPARQL 1.0

In this section, we discuss the expressivity of ASK queries in $\mathcal{AFO}_F\mathcal{U}$ (i.e., SPARQL 1.0). We study the expressivity of each operator from FILTER, UNION, OPT_F and AND. So, there are four cases.

The case of FILTER is easy. We show that an ASK query containing FILTER is not expressible in $\mathcal{AO}_F\mathcal{U}$. That is, $\mathcal{F} \not\preceq \mathcal{AO}_F\mathcal{U}$. To see this, we consider pattern $P = (?x, ?y, ?z) \text{ FILTER } ?x \neq ?y$ and graph $G = \{(c, c, c)\}$. Then $[\![P]\!]_G = \emptyset$. We will show that if a pattern P' is in $\mathcal{AO}_F\mathcal{U}$, then $[\![P']\!]_G \neq \emptyset$. Thus, the pattern P cannot be expressed in $\mathcal{AO}_F\mathcal{U}$. It is sufficient to show that the UNION, OPT_F and AND of two triple patterns (u, v, w) and (u', v', w') is always nonempty.

Case 1 $[\![(u, v, w) \text{ UNION } (u', v', w')]\!]_G$ contains two mappings $\mu : \text{var}((u, v, w)) \to \{c\}$ and $\mu' : \text{var}((u', v', w')) \to \{c\}$.

Case 2 $[\![(u, v, w) \text{ AND } (u', v', w')]\!]_G$ contains one mapping $\mu : \text{var}((u, v, w)) \cup \text{var}((u', v', w')) \to \{c\}$.

Case 3 $[\![(u, v, w) \text{ OPT}_F (u', v', w')]\!]_G$ contains one mapping $\mu : \text{var}((u, v, w)) \cup \text{var}((u', v', w')) \to \{c\}$.

Given $G = (c, c, c)$ and P' in $\mathcal{AOU}$, two mappings in $[\![P']\!]_G$ are always compatible. Therefore, $[\![P']\!]_G \neq \emptyset$ for each P' in $\mathcal{AOU}$. This implies P is not expressible in $\mathcal{AOU}$.

In the rest of this section, we investigate the expressivity of the other cases.

Expressivity of ASK queries with UNION

It is known that UNION is a primitive operator under select queries. It would be interesting to see whether this holds for ASK queries. We will demonstrate that this is not the case for ASK queries. Specifically, when OPT_F is absent, each ASK query with UNION can be expressed by some ASK query with FILTER and AND.

Proposition 28. *The following hold.*

- $\mathcal{AFU} \preceq \mathcal{AF}$;
- $\mathcal{FU} \preceq \mathcal{F}$.

From Proposition 28, we see that FILTER can be used to express ASK queries with UNION. Moreover, the next proposition shows that FILTER is necessary. We need a lemma before proving the proposition.

Lemma 29. *Let $G_1 = \{(a, a, a)\}$ and $G_2 = \{(b, b, b)\}$ be two graphs. For every pattern P in $\mathcal{AO}_F$, if both $[\![P]\!]_{G_1}$ and $[\![P]\!]_{G_2}$ are nonempty then $[\![P]\!]_{G_3}$ is nonempty for any graph $G_3 = \{(c, c, c)\}$ with $c \notin \text{con}(P)$.*

This lemma can be shown by a simple induction on the structure of patterns.

Proposition 30. $\mathcal{U} \not\preceq \mathcal{AO}_F$.

When OPT_F is involved, the conclusion of Proposition 28 still holds.

Proposition 31. $\mathcal{AFO}_F\mathcal{U} \preceq \mathcal{FO}_F$.

Expressivity of ASK queries with OPT

In this subsection we show that OPT is necessary in SPARQL 1.0 if UNION is absent. However, UNION is strong enough for expressing OPT. This result can be proven by the (non)-monotonicity of fragment of ASK queries.

We note that both $\mathcal{AFU}$ and $\mathcal{O}_F\mathcal{U}$ of ASK queries are monotone. But $\mathcal{AO}_F$ is non-monotone. To see this, consider the pattern P as follows:

$$P = ((?x, p, ?y) \text{ OPT } (?x, q, ?z)) \text{ AND } (?y, r, ?z).$$

Let $G_1 = \{(a, p, b), (b, r, c)\}$ and $G_2 = G_1 \cup \{(a, q, b)\}$. Then $[\![P]\!]_{G_1} = \{\{?x \to a, ?y \to b, ?z \to c\}\}$ while $[\![P]\!]_{G_2} = \emptyset$. Thus, P is not monotone. This implies $\mathcal{AO}_F$ is non-monotone.

Proposition 32. *The following hold.*

- $\mathcal{AO}_F \not\preceq \mathcal{AFU}$;
- $\mathcal{AO}_F \not\preceq \mathcal{O}_F\mathcal{U}$.

By Proposition 32, it is clear that $\mathcal{AO}_F\mathcal{U} \not\preceq \mathcal{AU}$.

Next, we show that OPT can be expressed by UNION. We first recall the notion of *principal subpattern*. Given a pattern P, the principal subpattern $\mathrm{ps}(P)$ of P is an OPT_F-free subpattern of P defined as follows (Zhang et al. 2018):

- $\mathrm{ps}((u, v, w)) = (u, v, w)$;
- $\mathrm{ps}(P_1 \text{ UNION } P_2) = \mathrm{ps}(P_1) \text{ UNION } \mathrm{ps}(P_2)$;
- $\mathrm{ps}(P_1 \text{ AND } P_2) = \mathrm{ps}(P_1) \text{ AND } \mathrm{ps}(P_2)$;
- $\mathrm{ps}(P_1 \text{ OPT}_F P_2) = \mathrm{ps}(P_1)$; and
- $\mathrm{ps}(P_1 \text{ FILTER } F) = \mathrm{ps}(P_1) \text{ FILTER } F$.

Lemma 33. *If P is a pattern in $\mathcal{O}_F\mathcal{U}$, then* $\mathrm{ASK}(P) \equiv_{\mathrm{ASK}} \mathrm{ASK}(\mathrm{ps}(P))$.

By the above lemma, it is easy to see the following result.

Proposition 34. *The following hold.*

- $\mathcal{O}_F\mathcal{U} \preceq \mathcal{U}$;
- $\mathcal{O}_F \preceq \mathcal{O}$.

Expressivity of ASK queries with AND

It is proven that AND can be expressed by OPT and FILTER (Zhang and den Bussche 2014, Proposition 4).

Moreover, we show that both OPT_F and FILTER are necessary for expressing AND. We need a lemma before proving the result.

Lemma 35. *Let P be a pattern in $\mathcal{FU}$. If there exists some graph G such that $[\![P]\!]_G$ is nonempty, then there must be a singleton graph G' such that $[\![P]\!]_{G'}$ is nonempty.*

It is in order to state the main result in this subsection.

Proposition 36. $\mathcal{A} \not\preceq \mathcal{FU}$.

By Propositions 34 and 36, $\mathcal{U}$ cannot be expressed by $\mathcal{O}_F$ alone.

Corollary 37. $\mathcal{O}_F\mathcal{U} \not\preceq \mathcal{O}_F$.

Based on Proposition 36 and Corollary 37, we can assert that both OPT_F and FILTER are necessary to express AND for ASK queries.

So far, we have investigated the expressivity of ASK queries in all 16 fragments of $\mathcal{AFO}_F\mathcal{U}$ in Figure 3.

Conclusion

In this paper, we have conducted a systematic study of relative expressivity for various fragments of SPARQL. Besides new results on SELECT queries, we have shown several interesting expressivity results for ASK queries. These results provide a guideline for SPARQL query optimisation and implementation. As a future work, we are going to investigate expressivity of fragments for ASK queries with the presence of some other operators in SPARQL 1.1 such as EXISTS and NOT EXISTS.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2017YFC0908401) and the National Natural Science Foundation of China (61972455, 61672377). Xiaowang Zhang is supported by the Peiyang Young Scholars at Tianjin University (2019XRX-0032).

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Angles, R., and Gutiérrez, C. 2008a. The expressive power of SPARQL. In *Proceedings of th 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany*, 114–129. Springer.

Angles, R., and Gutiérrez, C. 2008b. Survey of graph database models. *ACM Computing Survey* 40(1):1:1–1:39.

Angles, R., and Gutiérrez, C. 2010. SQL nested queries in SPARQL. In *Proceedings of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management, Buenos Aires, Argentina*. CEUR-WS.org.

Angles, R., and Gutiérrez, C. 2016. Negation in SPARQL. In *Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panama*. CEUR-WS.org.

Arenas, M., and Pérez, J. 2011. Querying semantic web data with SPARQL. In *Proceedings of the 30th ACM SIGMOD Symposium on Principles of Database Systems, PODS 2011, Athens, Greece*, 305–316. ACM.

Casanovas, E.; Dellunde, P.; and Jansana, R. 1996. On elementary equivalence for equality-free logic. *Notre Dame Journal of Formal Logic* 37(3):506–522.

Chandra, A. K., and Harel, D. 1980. Computable queries for relational data bases. *Journal of Computing System Science* 21(2):156–178.

Chebotko, A.; Lu, S.; and Fotouhi, F. 2009. Semantics preserving sparql-to-sql translation. *Data Knowledge Engineering* 68(10):973–1000.

Chekol, M. W. 2016. On the containment of SPARQL queries under entailment regimes. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA*, 936–942. AAAI Press.

Costabello, L.; Villata, S.; and Gandon, F. 2012. Context-aware access control for RDF graph stores. In *Proceedings of the 20th European Conference on Artificial Intelligence, ECAI 2012, Montpellier, France*, 282–287. IOS Press.

Cyganiak, R.; Wood, D.; and Lanthaler, M. 2014. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation.

Cyganiak, R. 2005. A relational algebra for SPARQL. Technical report, HP Laboratories Bristol.

Flum, J.; Frick, M.; and Grohe, M. 2002. Query evaluation via tree-decompositions. *Journal of ACM* 49(6):716–752.

Gottlob, G., and Pieris, A. 2015. Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina*, 2999–3007. AAAI Press.

Gottlob, G.; Grädel, E.; and Veith, H. 2002. Datalog LITE: a deductive query language with linear time model checking. *ACM Transactions Computational Logic* 3(1):42–79.

Gottlob, G.; Leone, N.; and Scarcello, F. 2001. The complexity of acyclic conjunctive queries. *Journal of ACM* 48(3):431–498.

Guido, N. 2015. On the static analysis for SPARQL queries using modal logic. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina*, 4367–4368. AAAI Press.

Harris, S., and Seaborne, A. 2013. *SPARQL 1.1 Query Language*. W3C Recommendation.

Hoffart, J.; Suchanek, F. M.; Berberich, K.; and Weikum, G. 2013. YAGO2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence* 194:28–61.

Kaminski, M., and Kostylev, E. V. 2016. Beyond well-designed SPARQL. In *Proceedings of the 19th International Conference on Database Theory, ICDT 2016, Bordeaux, France*, 5:1–5:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Kaminski, M.; Kostylev, E. V.; and Grau, B. C. 2016. Semantics and expressive power of subqueries and aggregates in SPARQL 1.1. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada*, 227–238. ACM.

Kaminski, M.; Kostylev, E. V.; and Grau, B. C. 2017. Query nesting, assignment, and aggregation in SPARQL 1.1. *ACM Transactions Database Systems* 42(3):17:1–17:46.

Kontchakov, R., and Kostylev, E. V. 2016. On expressibility of non-monotone operators in SPARQL. In *Proceedings of the Fifteenth International Conferenceon Principles of Knowledge Representation and Reasoning: , KR 2016, Cape Town, South Africa*, 369–379. AAAI Press.

Kostylev, E. V.; Reutter, J. L.; Romero, M.; and Vrgoc, D. 2015. SPARQL with property paths. In *Proceedings of the 14th International Semantic Web Conference, ISWC 2015, Bethlehem, PA, USA*, 3–18. Springer.

Kostylev, E. V.; Reutter, J. L.; and Ugarte, M. 2015. CONSTRUCT queries in SPARQL. In *18th International Conference on Database Theory, ICDT 2015, Brussels, Belgium*, 212–229. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Letelier, A.; Pérez, J.; Pichler, R.; and Skritek, S. 2013. Static analysis and optimization of semantic web queries. *ACM Transactions on Database Systems* 38(4):25:1–25:45.

Nikolaou, C., and Koubarakis, M. 2016. Querying incomplete information in RDF with SPARQL. *Artificial Intelligence* 237:138–171.

Pérez, J.; Arenas, M.; and Gutiérrez, C. 2009. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems* 34(3):16:1–16:45.

Pichler, R., and Skritek, S. 2014. Containment and equivalence of well-designed SPARQL. In *Proceedings of the 33rd ACM SIGMOD Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA*, 39–50. ACM.

Polleres, A. 2007. From SPARQL to rules (and back). In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada*, 787–796. ACM.

Prud'hommeaux, E., and Seaborne, A. 2008. *SPARQL Query Language for RDF*. W3C Recommendation.

Schmidt, M.; Meier, M.; and Lausen, G. 2010. Foundations of SPARQL query optimization. In *Proceedings of the 13th International Conference, ICDT 2010, Lausanne, Switzerland*, 4–33. ACM.

Wood, P. T. 2012. Query languages for graph databases. *SIGMOD Record* 41(1):50–60.

Zhang, X., and den Bussche, J. V. 2014. On the primitivity of operators in SPARQL. *Information Processing Letter* 114(9):480–485.

Zhang, X., and den Bussche, J. V. 2015. On the power of SPARQL in expressing navigational queries. *The Computer Journal* 58(11):2841–2851.

Zhang, X.; den Bussche, J. V.; Wang, K.; and Wang, Z. 2018. On the satisfiability problem of patterns in SPARQL 1.1. In *Proceedings of the 32 AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA*, 2054–2062. AAAI Press.

Zhang, X.; den Bussche, J. V.; and Picalausa, F. 2016. On the satisfiability problem for SPARQL patterns. *Journal Artificial Intelligence Research* 56:403–428.

Zhang, X.; Meng, C.; and Zou, L. 2018. Expressivity issues in SPARQL: monotonicity and two-versus three-valued semantics. *SCIENCE CHINA Information Sciences* 61(12):129102:1–3.