On the Expressivity of ASK Queries in SPARQL

Xiaowang Zhang, Jan Van den Bussche, Heng Zhang, Xuanxing Yang, Zhiyong Feng

1College of Intelligence and Computing, Tianjin University, Tianjin, China
2Faculty of Sciences, Hasselt University, Hasselt, Belgium
3School of Information and Communication Technology, Griffith University, Brisbane, Australia
4Tianjin 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 Gand on 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; Nikolau 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 ⇒ Expressible for ASK queries ⇒ 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 (\&), MINUS (\textit{M}) and UNION (\textit{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, \( ADO_F \) 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](image)

- 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](image)

- 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.

**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 \( \text{bnd}(?x) \), \( ?x = ?y \), and \( ?x = c \), using logical connectives \( \land, \lor, \text{and} \). For a pattern \( P \), \( \text{var}(P) \) and \( \text{con}(P) \) 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 \( \mu : S \rightarrow U \) on a finite set \( S \) of variables. The domain \( S \) of \( \mu \) is denoted...
dom(μ). Two mappings μ₁ and μ₂ are compatible, denoted by μ₁ ∼ μ₂, if they agree on the intersection of their domains. The empty mapping (μₑ), 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[μ] = true if F is satisfied by μ and otherwise F[μ] = false (Kontchakov and Kontchakov 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, Ω₁, and Ω₂ 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:

- \( π_S(Ω) = \{ μ | S ⊆ \text{dom}(μ) \land μ ∈ Ω \} \);
- \( σ_F(Ω) = \{ μ | μ ∈ Ω \land F[μ] = \text{true} \} \);
- \( Ω_1 ∙ Ω_2 = \{ μ_1 \cup μ_2 | μ_1 ∈ Ω_1, μ_2 ∈ Ω_2, μ_1 ∼ μ_2 \} \);
- \( Ω_1 \setminus_F Ω_2 = \{ μ_1 ∈ Ω_1 | \forall μ_2 ∈ Ω_2, (μ_1 \neq μ_2) \lor (F[μ_1 ∪ μ_2] \neq \text{true}) \} \);
- \( Ω_1 \cup Ω_2 = \{ μ | μ ∈ Ω_1 \lor μ ∈ Ω_2 \} \);
- \( Ω_1 \setminus Ω_2 = \{ μ_1 ∈ Ω_1 | \forall μ_2 ∈ Ω_2, (μ_1 \neq μ_2) \lor (\text{dom}(μ_1) \cap \text{dom}(μ_2) = ∅) \} \);
- \( Ω_1 \setminus_F Ω_2 = σ_F(Ω_1 \setminus Ω_2) \cup (Ω_1 \setminus_F Ω_2) \).

Note that we use \( \setminus \) and \( \setminus_F \) to denote \( \setminus_T \) and \( \setminus_T \) respectively. That is,

\[ Ω_1 \setminus Ω_2 = \{ μ | μ ∈ Ω_1 | \forall μ_2 ∈ Ω_2, μ_1 \neq μ_2 \}. \]

\[ Ω_1 \setminus_F Ω_2 = (Ω_1 \setminus Ω_2) \cup (Ω_1 \setminus_F Ω_2). \]

Clearly, the following equations hold:

\[ Ω_1 \setminus_F Ω_2 = Ω_1 \setminus Ω_2 \setminus_F Ω_2 \]

\[ Ω_1 \setminus_F Ω_2 = σ_F(Ω_1 \setminus Ω_2) \cup (Ω_1 \setminus_F Ω_2). \]

**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:

- \([u,v,w]_G = \{ μ | \{ u,v,w \} \cap V \rightarrow U | (μ(u), μ(v), μ(w)) ∈ G \} \).

Here \( μ(c) = c \) for any \( c ∈ U \).

- \([P_1 \cup P_2]_G = [P_1]_G \cup [P_2]_G \);
- \([P_1 \land P_2]_G = [P_1]_G \land [P_2]_G \);
- \([P_1 \setminus_F P_2]_G = [P_1]_G \setminus_F [P_2]_G \);
- \([P_1 \setminus P_2]_G = [P_1]_G \setminus [P_2]_G \);
- \([P_1 \text{ FILTER } F]_G = σ_F([P_1]_G) \);
- \([P_1 \text{ OPT } P_2]_G = [P_1]_G \circ_P [P_2]_G \);
- \([\text{SELECT}_S(P_1)]_G = π_S([P_1]_G) \).

Clearly, the semantics of \( P_1 \setminus_F P_2 \) and \( P_1 \text{ OPT } P_2 \) on G are equivalently defined as follows:

\([P_1 \setminus_F P_2]_G = [P_1]_G \setminus [P_2]_G \).

\([P_1 \text{ OPT } P_2]_G = [P_1]_G \circ_P [P_2]_G \).

Standard Equivalence: Two patterns P and \( P' \) are equivalent, denoted \( P ≡ P' \), 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:

- \( P_1 \setminus F P_2 = P_1 \setminus F ((P_1 \land P_2) \text{ FILTER } F) \);
- \( P_1 \text{ OPT } P_2 = ((P_1 \land P_2) \text{ FILTER } F) \cup (P_1 \setminus_F P_2 \text{ FILTER } F) \).

By Lemma 1, \( \setminus_F \) and \( \text{OPT}_F \) are not necessary if \( \text{AND}, \text{FILTER}, \text{UNION} \) and \( \setminus_F \) 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 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 \( \text{ASK}(P)_G \) is true if \([Q]_G \neq ∅ \); and false otherwise.

**Definition 2.** Two ASK queries \( Q_1 \) and \( Q_2 \) are equivalent, denoted \( Q_1 ≡ \text{ASK} Q_2 \), if for every graph G, \( Q_1^G = Q_2^G \) is true i f \( Q^G \) is true. If \( Q_1 = \text{ASK}(P_1) \) for \( i = 1, 2 \) and \( Q_1 = \text{ASK} Q_2 \), then we say that \( P_1 \) and \( P_2 \) are ASK-equivalent, denoted \( P_1 ≡ \text{ASK} P_2 \).

We say that \( Q \) is ASK-expressible, or just expressible, in a fragment \( W \) if \( Q ≡ \text{ASK} Q' \) for some query \( Q' \) in \( W \).

A fragment \( W_1 \) is ASK-expressible, or just expressible, in another fragment \( W_2 \), denoted by \( W_1 ⊆ W_2 \), if every ASK query in \( W_1 \) is expressible in \( W_2 \). Otherwise, \( W_1 \not ⊆ W_2 \).

By \( W_1 \not ⊆ W_2 \), we mean \( W_1 \not ⊆ W_2 \) and \( W_2 \not ⊆ W_1 \).

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

\[ P \; \text{FILTER} \; F \equiv_P P \; \text{DIFF} \; (P \; \text{DIFF} \; (?x, ?y, ?z)), \]

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.

- \( \text{DIFF}_F \; \text{FO}_U \leq \text{DIFF}_F \; \text{U} \)
- \( \text{DIFF}_F \; \text{U} \leq \text{DIFF}_F \; \text{U} \)

The next result shows that UNION of ASK queries in \( \text{DIFF}_F \; \text{U} \) and \( \text{DIFF}_F \; \text{U} \) 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

\[ P_1 \; \text{UNION} \cdots \; \text{UNION} \; P_m \quad (1) \]

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

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

\[ P \equiv_{\text{ASK}} (?x, ?y, ?z) \; \text{DIFF} \; (((?x', ?y', ?z') \; \text{DIFF} \; P_1) \cdots \; \text{DIFF} \; P_m) \quad (2) \]

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

\[ Q \; \text{DIFF} \; (P_1 \; \text{UNION} \cdots \; \text{UNION} \; P_{m-1} \; \text{UNION} \; P_m) \equiv_{\text{ASK}} ((Q \; \text{DIFF} \; P_1) \cdots \; \text{DIFF} \; P_{m-1}) \; \text{DIFF} \; P_m. \]

Since each pattern in \( \text{DIFF}_F \) 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.

- \( \text{DIFF}_F \; \text{U} \preceq \text{DIFF}_F \)
- \( \text{DIFF}_F \; \text{U} \preceq \text{DIFF}_F \)

It is interesting to see whether \( \text{A} \) is expressible in \( \text{DIFF}_F \).

The answer is negative.

**Proposition 8.** \( \text{A} \not\preceq \text{DIFF}_F \).

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 \( W_1 = \{ \text{DIFF}_F, \text{DIFF}_F \; \text{FO}_U, \text{DIFF}_F \; \text{U} \} \) are equivalent to each other.
2. All fragments in \( W_2 = \{ \text{DIFF}_F, \text{DIFF}_F \; \text{FO}_U, \text{DIFF}_F \; \text{FO}_U, \text{DIFF}_F \; \text{U} \} \) are equivalent to each other.
3. If \( W_1 \in W_1 \) and \( W_2 \in W_2 \), then \( W_1 \not\preceq W_2 \).

**Figure 4: Expressivity of ASK queries with DIFF_F**

- \( \text{DIFF}_F, \text{DIFF}_F \; \text{FO}_U, \text{DIFF}_F \; \text{U} \)
- \( \text{DIFF}_F \; \text{U} \)

### Expressivity of ASK queries with either DIFF or MINUS

In this subsection, we consider the 32 fragments of \( \text{AFOU} \) with either DIFF or MINUS.

We show that DIFF_F is expressible in \( \text{MO}_F \) and \( \text{DO}_F \).

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

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

Now we wonder to know whether \( \text{AD}_F \) is expressible in \( \text{DO}_F \). The answer is positive.

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

\[ (P \; \text{OPT}_{F \land ?z_1 = ?z_2} \; (?x_1, ?y_1, ?z_1) \; \text{OPT} \; (?x_2, ?y_2, ?z_2)) \; \text{DIFF} \; Q, \quad (3) \]

\[ Q = ((?x_1, ?y, ?z) \; \text{OPT}_{?z_1 \neq ?z_2} \; (?x_2, ?u_1, ?v_1)) \; \text{OPT}_{?x_1 \neq ?x_2} \; (?u_2, ?x_2, ?v_2)) \]

\( \text{OPT}_{?z_1 \neq ?z_2} \; (?u_3, ?v_3, ?x_2); \quad (4) \]

where \(?y, ?z, ?x_1, ?x_2, ?z, ?y_1, ?y_2, ?z_1, ?z_2, ?u_i, ?v_i, ?u_i \) (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:

\[ [P \; \text{FILTER} \; F]_G = [\text{FILTER}(P, F)]_G. \]
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, $[P]_G \neq \emptyset$ since $[F]_G = \emptyset$ no matter what $F$ is.

So, we consider all patterns in $AFEOFU$ but graphs containing only one constant.

For the moment we consider the fragment $AFEO$ where all triple patterns are constant-free and all filters are atomic.

Given a pattern $P$ in $AFEO$, its mapping schema, denoted by $\Gamma (P)$, is defined as a set of variables by

\[
\Gamma (\{u_0, v_0, w_0\}) = \{u_0, v_0, w_0\};
\]

\[
\Gamma (P_1 \text{ AND } P_2) = \Gamma (P_1) \cap \Gamma (P_2);
\]

\[
\Gamma (P_1 \text{ DIFF } P_2) = \Gamma (P_1);
\]

\[
\Gamma (P_1 \text{ FILTER } F) = \Gamma (P_1);
\]

A pattern $P$ in $AFEO$ is said to be safe if $\text{var}(F) \subseteq \Gamma (P_1)$ for every subpattern $Q$ of the form $P_1 \text{ FILTER } F$. 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 $AFEO\text{safe}$ where all patterns are safe and all filters are atomic.

Let $v_0$ be a fixed variable and $P$ be a pattern in $AFEO\text{safe}$. We define $\delta (P)$ as an $AD$-pattern in an inductive way.

\[
\delta (u_0, v_0, w_0) = \{u_0, v_0, w_0\};
\]

\[
\delta (P_1 \text{ AND } P_2) = \delta (P_1) \text{ AND } \delta (P_2);
\]

\[
\delta (P_1 \text{ DIFF } P_2) = \delta (P_1) \text{ DIFF } \delta (P_2);
\]

\[
\delta (P_1 \text{ FILTER } F) = \delta (P_1);
\]

\[
\delta (\text{ FILTER } \text{ ? } y = \text{ } z) = \delta (P);
\]

\[
\delta (P_1 \text{ FILTER } \text{ ? } y \neq \text{ } z) = \delta (P) \text{ DIFF } \{x_0, x_0, x_0\};
\]

\[
\delta (P_1 \text{ FILTER } \text{ ? } y = \text{ } c) = \delta (P) \text{ AND } (c, c, c);
\]

\[
\delta (P_1 \text{ FILTER } \text{ ? } y \neq \text{ } c) = \delta (P) \text{ DIFF } (c, c, c);
\]

The definition of $\delta (P)$ can be extended in $DOF$.

**Lemma 12.** Let $P$ be a pattern in $AFEOFU$. For any RDF graph $G$ of the form $\{c, c, c\}$ with $c \in U$, the following hold:

$[P]_G$ is nonempty if and only if $[\delta (P)]_G$ is nonempty.

By Proposition 11 and Lemma 12, we can show that patterns in $AFEOFU$ are expressible in $DOF$.

We first prove a proposition.

Given a pattern $P$ in $AFEOFU$, we define

\[
P^* = P^F \text{ UNION } \delta (P) \text{ AND } P_1;
\]

where $P^F$ is a FILTER-free pattern by substituting $\text{FILTER}(P', F)$ for all sub-patterns of the form $P' \text{ FILTER } F$ in $P$, $P_1$ is defined by

\[
P_1 = (((x_0, y_0, z_0) \text{ OPT } x_2 = \text{ } z_2) \text{ OPT } x_1 = \text{ } y_1 \text{ OPT } (x_2, y_2, z_2)) \text{ DIFF } Q.
\]

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 $AFEOFU$ and the pattern $P^*$ is defined as above. Then $P \equiv_{ASK} P^*$.

Now, we show that $AFEOFU$ is expressible in $DOF$.

**Theorem 14.** $AFEOFU \preceq DOF$.

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

**Corollary 15.** The following hold.

$AF \text{- } MOFU \preceq MOF$.

$AFEOFU \preceq DOF$.

Next, we show that OPT is necessary to express FILTER when DIFF 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 $ADMU$, we have the following result.

**Proposition 16.** $ADMU \preceq AMU$.

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

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

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 $h : \text{const}(G) \rightarrow \text{const}(G')$ such that $(s, p, o) \in G$ iff $(h(s), h(p), h(o)) \in G'$ for all constants $s, p, o \in \text{const}(G)$, and $h(c) = c$ for all $c \in C$.

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

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

Note that OPT is needed for expressing FILTER.

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

**Theorem 19.** $F \not\preceq ADMU$.

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

$h(a) = h(b) = a$ and $h(p) = p$.

Clearly, $h$ is a surjective strict $C$-homomorphism 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 $F \not\preceq AMU$ and $F \not\preceq ADU$.

We note that $DAMU$ 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. $A \not\subseteq DMU$.

By Proposition 20, we have that $A \not\subseteq MU$ and $A \not\subseteq DU$. Next, we would like to know whether ASK queries in a fragment of $AFDU$ 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 $AFDU$ is equivalent to an $AFD$ pattern in UNF, by Lemma 21, we conclude the following.

Proposition 22. The following hold.

- $ADU \subseteq AD$;
- $FDU \subseteq FD$;
- $DU \subseteq D$.

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

Proposition 23. $AMU \subseteq AM$.

Let $P$ be a pattern in $FMU$. 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. $FMU \subseteq FM$.

Surprisingly, different from DIFF, MINUS cannot express UNION.

Proposition 25. $U \not\subseteq M$.

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 $M$ 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 $M$ such that $P^{G_1} = false$ but $P^{G_2} = true$ and $P^{G_3} = true$.

Therefore, we can conclude that $U \not\subseteq M$.

Clearly, by Proposition 25, it is immediate to conclude that $MU \not\subseteq M$.

In the proof of $U \not\subseteq M$, we consider patterns with only constants. So we wonder whether a pattern in $U$ is expressible in $M$ 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 $MU$. If $\text{pos}(P)$ contains no variable-free triple patterns, then $\text{ASK}(P)$ can be expressed in $M$.

Expressivity of ASK queries in SPARQL 1.0

In this section, we discuss the expressivity of ASK queries in $AFOFU$ (i.e., SPARQL 1.0). We study the expressivity of each operator from FILTER, UNION, OPT, 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 $AOFU$.

That is, $F \not\subseteq AO$. To see this, we consider pattern $P = (?x, ?y, ?z)$ FILTER $?x \neq ?y$ and graph $G = \{(c, c, c)\}$. Then $[P]_G = \emptyset$. We will show that if a pattern $P'$ is in $AOFU$, then $[P']_G \neq \emptyset$. Thus, the pattern $P$ cannot be expressed in $AOFU$. It is sufficient to show that the UNION, OPT, 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 $\mu : \text{var}((u, v, w)) \rightarrow \{e\}$ and $\mu' : \text{var}((u', v', w')) \rightarrow \{\}$.

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

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

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

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 is absent, each ASK query with UNION can be expressed by some ASK query with FILTER and AND.

Proposition 28. The following hold.

- $AFU \subseteq AF$;
- $FU \subseteq F$.

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 $AOF$, 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 $c \notin \text{conn}(P)$.
Lemma 33. If \( \text{AND} \) is proven the result.

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 \( \text{OPT}_F \) and \( \text{FILTER} \) 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 \( \text{SELECT} \) 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 \( \text{EXISTS} \) 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


