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; Nikolaoú 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 (\(\land\)), MINUS (\(\setminus\)) and UNION (\(\cup\)) 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

\[ \text{ADF, ADF}_{OF}, \text{ADF}_U, \text{ADF}_{OF_1}, \text{ADF}_{OF_2}, \text{DF_{OF}}, \text{DF_{OF_1}}, \text{DF_{OF_2}}, \text{DF_{OF_3}}, \]  

- 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

\[ \text{AFM, AFMO}_U, \text{AFMO}_{OF}, \text{AMO}_U, \text{AMO}_{OF}, \text{AMO}_{OF_1}, \text{AMO}_{OF_2}, \text{FMO}, \text{FMO}_{OF}, \text{FMO}_{OF_1}, \text{FMO}_{OF_2}, \text{MO}_{OF}, \text{MO}_{OF_1}, \text{MO}_{OF_2} \]  

- 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:

\[
P := (u, v, w) \mid P_1 \text{ AND } P_2 \mid P_1 \text{ OPT}_f \ P_2 \mid \]

\[
P_1 \text{ DIFF } P_2 \mid P_1 \text{ UNION } P_2 \mid P_1 \text{ MINUS } P_2 \mid \]

\[
P_1 \text{ FILTER } F \mid \text{ SELECT}_S(P_1). \]

A filter condition \( F \) (or just filter) is a formula constructed from (filter) atoms of the form \( \text{bud}(?x) \), \( ?x = ?y \), and \( ?x = c \), using logical connectives \( \land, \lor, \text{ and } \neg \).

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.

**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 (μ∅), 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 Fμ, is defined in terms of a two-valued logic. That is, Fμ = true if F is satisfied by μ and otherwise Fμ = 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, 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:

- **πS(Ω) = {μ|S ∈ dom(μ) | μ ∈ Ω};**
- **σF(Ω) = {μ ∈ Ω | Fμ = true};**
- **Ω1 \* Ω2 = {μ1 ∪ μ2 | μ1 ∈ Ω1, μ2 ∈ Ω2, μ1 ¨= μ2};**
- **Ω1 \ F Ω2 = {μ1 ∈ Ω1 | ∀ μ2 ∈ Ω2, (μ1 ¨= μ2) ∨ (Fμ1∪μ2 ̸= true)};**
- **Ω1 ≈ Ω2 = {μ | μ ∈ Ω1 ∨ μ ∈ Ω2};**
- **Ω1 \* Ω2 = {μ1 ∈ Ω1 | ∀ μ2 ∈ Ω2, (μ1 ¨= μ2) ∨ (dom(μ1) ∩ dom(μ2) = ∅)};**
- **Ω1 \* F Ω2 = σF(Ω1 \* Ω2) ∪ (Ω1 \ F Ω2).**

Note that we use \ and △ to denote \ and \* respectively. That is,

- **Ω1 \ Ω2 = {μ1 ∈ Ω1 | ∀ μ2 ∈ Ω2, μ1 ¨= μ2}.
- Ω1 \* Ω2 = (Ω1 \* Ω2) ∪ (Ω1 \ \ F Ω2).**

Clearly, the following equations hold:

- **Ω1 \ F Ω2 = Ω1 \ σF(Ω1 \* Ω2).**
- **Ω1 \* F Ω2 = σF(Ω1 \* Ω2) ∪ (Ω1 \ \ σF(Ω1 \* Ω2)).**

**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} \ V → U | (μ(u), μ(v), μ(w)) \ G}.**

Here μ(c) = c for any c ∈ U.

- **[P1 UNION P2]G = [P1]G ∪ [P2]G;**
- **[P1 AND P2]G = [P1]G ∩ [P2]G;**
- **[P1 DIFF F P2]G = [P1]G \ F [P2]G;**
- **[P1 MINUS P2]G = [P1]G \ [P2]G;**
- **[P1 FILTER F]G = σF([P1]G);**
- **[P1 OPT F P2]G = [P1]G OPT F [P2]G;**
- **[SELECT S(P1)]G = πS([P1]G).**

Clearly, the semantics of P1 DIFF P2 and P1 OPT P2 on G are equivalently defined as follows:

- **[P1 DIFF P2]G = [P1]G \ [P2]G.**
- **[P1 OPT P2]G = [P1]G OPT F [P2]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 P1 and P2 be two patterns and F a filter. For any graph G, the following two equations hold:

- **P1 DIFF F P2 = P1 DIFF ((P1 AND P2) FILTER F);**
- **P1 OPT F P2 = ((P1 AND P2) FILTER F) UNION (P1 DIFF ((P1 AND P2) 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 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 ASK(P)G is true if [P]G ̸= ∅; and false otherwise.

**Definition 2.** Two ASK queries Q1 and Q2 are equivalent, denoted Q1 ≡ ASK Q2, if for every graph G, Q1G = true iff Q2G = true. If Q1 = ASK(P1) for i = 1, 2 and Q1 ≡ ASK Q2, then we say that P1 and P2 are ASK-equivalent, denoted P1 ≡ ASK P2.

We say that Q is ASK-expressible, or just expressible, in a fragment W if Q ∈ ASK Q′ for some query Q′ in W.

A fragment W1 is ASK-expressible, or just expressible, in another fragment W2, denoted by W1 ≲ W2, if every ASK query in W1 is expressible in W2. Otherwise, W1 ̸≲ W2. By W1 ≲ W2, we mean W1 ⊆ W2 and W2 ̸⊆ W1.

An ASK query Q is monotone if for each pair of graphs G1 and G2 with G1 ⊆ G2, QG1 = true implies QG2 = 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. 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 Kontchakov 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 $P$ FILTER $F$ \( \equiv P \) 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.
- \( AD_F \cdot FO \cdot U \preceq AD_U \).
- \( DP \cdot F \cdot U \preceq DP \cdot U \).

The next result shows that UNION of ASK queries in \( AD_F \cdot U \) and \( DP \cdot U \) 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

\[ P_1 \text{ UNION } \ldots \text{ UNION } P_m \]

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

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

\[ P \equiv_{\text{ASK}} (\text{DIFF } ((?!x', ?y', ?z') \text{ DIFF } P_1) \ldots \text{ DIFF } P_m) \]

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 $Q, P_1, \ldots, P_m$ be patterns. Then

\[ Q \text{ DIFF } (P_1 \text{ UNION } \ldots \text{ UNION } P_{m-1} \text{ UNION } P_m) \]

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

Since each pattern in \( AD_U \) 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.
- \( D_F \cdot U \preceq D_F \).
- \( AD_F \cdot U \preceq AD_F \).

It is interesting to see whether $A$ is expressible in $D_F$. The answer is negative.

Proposition 8. $A \not\preceq D_F$.

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 $W_1 = \{ D_F, D_F \cdot F \cdot U, D_F \cdot U \}$ are equivalent to each other.
2. All fragments in $W_2 = \{ AD_F, AD_F \cdot F \cdot U, AD_F \cdot FO \cdot U, AD_F \cdot F \cdot O \cdot U, AD_F \cdot U, D_F \cdot FO \cdot U, D_F \cdot F \cdot O \cdot U, D_F \cdot O \cdot F, D_F \cdot F \cdot O \cdot U \}$ are equivalent to each other.
3. If $W_1 \in W_2$ and $W_2 \in W_2$, then $W_1 \not\preceq W_2$.

Figure 4: Expressivity of ASK queries with DIFF$_F$. $\overline{AD_F, AD_F \cdot F \cdot U, AD_F \cdot FO \cdot U, AD_F \cdot F \cdot O \cdot U, AD_F \cdot O \cdot U, D_F \cdot FO \cdot U, D_F \cdot F \cdot O \cdot U, D_F \cdot O \cdot F, D_F \cdot F \cdot O \cdot U}$

Expressivity of ASK queries with either DIFF or MINUS
In this subsection, we consider the 32 fragments of AFOU with either DIFF or MINUS.

We show that DIFF$_F$ is expressible in MO$_F$ and DO$_F$.

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

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

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

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

\[ (P \text{ OPT } ((?x_1, ?y_1, ?z_1) \text{ OPT } ((?x_2, ?y_2, ?z_2)) \text{ DIFF } Q, \]

\[ Q = ((?!x_1, ?y, ?z) \text{ OPT } (?x_2, ?u_1, ?v_1) \text{ OPT } (?x_1, ?x_2, ?u_2, ?v_2) \text{ OPT } (?x_1, ?u_3, ?v_1, ?x_2) \text{ OPT } (?x_1, ?u_3, ?v_3, ?x_2)) \]

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:

\[ [P \text{ FILTER } F]_G = [\text{Filter}(P, F)]_G. \]
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, $\left[P \text{ DIFF } P \right]_{G} \neq \emptyset$ since $[P]_{G} = \emptyset$ no matter what $F$ is.

So, we consider all patterns in $ADF_{P}U$ but graphs containing only one constant.

For the moment we consider the fragment $ADF_{0}$ where all triple patterns are constant-free and all filters are atomic.

Given a pattern $P$ in $ADF_{0}$, its mapping schema, denoted by $\Gamma(P)$, is defined as a set of variables by
- $\Gamma((?u,?u,?w)) = \{(?u,?v,?w)\};$
- $\Gamma(P_{1} \text{ AND } P_{2}) = \Gamma(P_{1}) \cap \Gamma(P_{2});$
- $\Gamma(P_{1} \text{ DIFF } P_{2}) = \Gamma(P_{1});$
- $\Gamma(P_{1} \text{ FILTER } F) = \Gamma(P_{1}).$

A pattern $P$ in $ADF_{0}$ is said to be safe if $\text{var}(F) \subseteq \Gamma(P)$ 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 $ADF_{0}^{safe}$ where all patterns are safe and all filters are atomic.

Let $?X_{0}$ be a fixed variable and $P$ be a pattern in $ADF_{0}^{safe}$. We define $\delta(P)$ as an $ADF$-pattern in an inductive way.
- $\delta((?u,?u,?w)) = (??u,?w,?z);$
- $\delta(P_{1} \text{ AND } P_{2}) = \delta(P_{1}) \text{ AND } \delta(P_{2});$
- $\delta(P_{1} \text{ DIFF } P_{2}) = \delta(P_{1}) \text{ DIFF } \delta(P_{2});$
- $\delta(P \text{ FILTER } \text{bound}(?y)) = \delta(P);$
- $\delta(P \text{ FILTER } ?y =?z) = \delta(P);$
- $\delta(P \text{ FILTER } ?y \neq ?z) = \delta(P) \text{ DIFF } (?x_{0},?x_{0},?x_{0});$
- $\delta(P \text{ FILTER } ?y = c) = \delta(P) \text{ AND } (c,c,c);$
- $\delta(P \text{ FILTER } ?y \neq c) = \delta(P) \text{ DIFF } (c,c,c).$

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

Lemma 12. Let $P$ be a pattern in $ADF_{P}U$. 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 $ADF_{P}U$ are expressible in $DO_{F}$.

We first prove a proposition.

Given a pattern $P$ in $ADF_{P}U$, we define
- $P^{*} = P^{F} \text{ UNION } \delta(P) \text{ AND } P_{1};$ (5)

where $P^{F}$ is a FILTER-free pattern by substituting Filter$(P', F)$ for all sub-patterns of the form $P' \text{ FILTER } F$ in $P$. $P_{1}$ is defined by
- $P_{1} = ((?x_{3},?y_{3},?z_{3}) \text{ OPT}\_?z_{3}=?z_{2} ((?x_{1},?y_{1},?z_{1}) \text{ OPT}\_?x_{2}=?x_{2},?y_{2}=?y_{2}) \text{ DIFF } Q).$ (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 $ADF_{P}U$ and the pattern $P^{*}$ is defined as above. Then $P \equiv_{ASK} P^{*}$.

Now, we show that $ADF_{P}U$ is expressible in $DO_{F}$.

Theorem 14. $ADF_{P}U \subseteq DO_{F}$.

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

Corollary 15. The following hold.
- $ADF_{MO}U \subseteq MO_{F}$.
- $ADF_{P}U \subseteq DO_{F}$.

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 $ADM_{U}$, we have the following result.

Proposition 16. $ADM_{U} \subseteq AMU$.

In the proof of Proposition 16, in a pattern $P$ in $ADM_{U}$, for any subpattern $Q$ of the form $P_{1} \text{ DIFF } P_{2}$, we can rewrite it into $P_{1} \text{ MINUS } P_{2}$.

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

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 $C = \text{con}(P)$. Let $G$ and $G'$ be two graphs such that there is a surjective strict $C$-homomorphism $h$ from $G$ onto $G'$. Then
- $\mu \in [P]_{G'}$ if and only if $\mu \circ h \in [P]_{G}$.
- $\mu \in [P]_{G'}$ if and only if $\mu = \mu' \circ h$ 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. $F \not\subseteq \text{AMU}$.

Proof. Consider the pattern $P = (?(x,p,?y) \text{ FILTER } ?x \neq ?y)$ 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 $h(a) = h(b) = a$ and $h(p) = p$.

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

We note that $D_{M}U$ 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. \( A \not\subseteq D M U \).

By Proposition 20, we have that \( A \not\subseteq M U \) and \( A \not\subseteq D U \). Next, we would like to know whether ASK queries in a fragment of \( A F D U \) are expressible in the corresponding fragment with \( \text{UNION} \). The answer is positive.

To do so, we need an important lemma.

Lemma 21. Let \( P \) be a pattern in \( \text{UNF} \) as Equation (1) and \( P' \) be the \( \text{UNION-free} \) pattern
\[
(?x, ?y, ?z) \ \text{DIFF} \ \left( ((?x', ?y', ?z') \ \text{DIFF} \ (P_1)) \ldots \ \text{DIFF} \ (P_m) \right),
\]
where \(?x, ?y, ?z, ?x', ?y', ?z'\) are fresh variables.

Since each pattern in \( AF D U \) is equivalent to an \( AF D \) pattern in \( \text{UNF} \), by Lemma 21, we conclude the following.

Proposition 22. The following hold.

- \( ADU \leq AD \);
- \( FDU \leq FD \);
- \( DU \leq D \).

Finally, we show that ASK queries in a fragment of \( AF, MU \) are expressible in the corresponding fragment without \( \text{UNION} \).

Proposition 23. \( AMU \leq AM \).

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 \( \text{UNION} \) is expressible by \( \text{FILTER} \) and \( \text{MINUS} \).

Proposition 24. \( FMU \leq FM \).

Surprisingly, different from \( \text{DIFF} \), \( \text{MINUS} \) cannot express \( \text{UNION} \).

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 \( G \), \( Q^G = \text{true} \) if \( \text{ASK}(P)^G = \text{true} \).

In order to prove this claim, consider three graphs:

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

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

Therefore, we can conclude that \( U \not\subseteq M \). \( \square \)

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 \( AFO_F U \) (i.e., SPARQL 1.0). We study the expressivity of each operator from \( \text{FILTER} \), \( \text{UNION} \), \( \text{OPT}_F \) and \( \text{AND} \). So, there are four cases.

The case of \( \text{FILTER} \) is easy. We show that an ASK query containing \( \text{FILTER} \) is not expressible in \( AOF_U \).

From Proposition 28, we see that \( \text{FILTER} \) can be used to express ASK queries with \( \text{UNION} \). Moreover, the next proposition shows that \( \text{FILTER} \) 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 \( c \notin \text{con}(P) \).
This lemma can be shown by a simple induction on the structure of patterns.

**Proposition 30.** \( U \not\subseteq \mathcal{AO}_F \).

When \( \text{OPT}_F \) is involved, the conclusion of Proposition 28 still holds.

**Proposition 31.** \( \mathcal{AFO}_F U \not\subseteq \mathcal{FO}_F \).

### 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 \( \mathcal{OFU} \) of ASK queries are non-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 \( [P]_{G_1} = \{(?x \rightarrow a, ?y \rightarrow b, ?z \rightarrow c)\} \) while \( [P]_{G_2} = \emptyset \). Thus, \( P \) is not monotone. This implies \( \mathcal{AO}_F \) is non-monotone.

**Proposition 32.** The following hold.

- \( \mathcal{AO}_F \not\subseteq \mathcal{AFU} \);
- \( \mathcal{AO}_F \not\subseteq \mathcal{OFU} \).

By Proposition 32, it is clear that \( \mathcal{AO}_F U \not\subseteq \mathcal{AU} \).

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 \( ps(P) \) of \( P \) is an OPT-free subpattern of \( P \) defined as follows (Zhang et al. 2018):

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

**Lemma 33.** If \( P \) is a pattern in \( \mathcal{OFU} \), then \( \text{ASK}(P) \equiv_{\text{ASK}} \text{ASK}(ps(P)) \).

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

**Proposition 34.** The following hold.

- \( \mathcal{OFU} \not\subseteq \mathcal{U} \);
- \( \mathcal{OF} \not\subseteq \mathcal{O} \).

### 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 \( \text{OPT}_F \) and \( \text{FILTER} \) 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.** \( A \not\subseteq \mathcal{FU} \).

By Propositions 34 and 36, \( \mathcal{U} \) cannot be expressed by \( \mathcal{OF} \) alone.

**Corollary 37.** \( \mathcal{OFU} \not\subseteq \mathcal{OF} \).

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 \( \mathcal{AFU}_F U \) 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 the 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):1-1–39.

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 Conference 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 of 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.