SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Expressivity Issues in SPARQL: Monotonicity and Two-versus Three-valued Semantics

Xiaowang ZHANG[1,2*], Chenhong MENG[1,2] & Lei ZOU[3]

[1]*School of Computer Science and Technology, Tianjin University, Tianjin* 300350*, China;*
[2]*Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin* 300350*, China;*
[3]*Peking University, Beijing* 100871*, China*

Appendix A Preliminaries

In this section, we briefly introduce SPARQL and relational algebra. For more comprehensive background knowledge, we refer the reader to [1, 5].

RDF graph Let I, B, and L be infinite sets of *IRIs*, *blank nodes* and *literals*, respectively. These three sets are pairwise disjoint. We denote the union $I \cup B \cup L$ by U, and elements of $I \cup L$ will be referred to as *constants*.

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

The syntax of SPARQL The official syntax of SPARQL [6] considers operators OPTIONAL, UNION, and FILTER, and concatenation via a point symbol (.), to construct (graph) pattern. The syntax also considers {} to group patterns, and some implicit rules of precedence and association. For example, the point symbol (.) has precedence over OPTIONAL, and OPTIONAL is left associative. In order to avoid ambiguities in the parsing, we present the syntax of SPARQL patterns in a more traditional algebraic formalism, using binary operators AND, UNION, OPT, and FILTER, respectively. We fully parenthesize expressions making explicit the precedence and association of operators.

A (SPARQL) pattern is defined in an inductive way:

- A tuple from $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$ is a pattern (a *triple pattern*).
- If P_1 and P_2 are patterns, then $(P_1$ AND $P_2)$, $(P_1$ OPT $P_2)$, and $(P_1$ UNION $P_2)$ are patterns.
- If P is a pattern and C is a SPARQL built-in condition, then (P FILTER C) is a pattern.

A SPARQL *constraint* (or *built-in condition*) is constructed using elements of the set $I \cup L \cup V$ and constants, logical connectives ($\neg$, $\wedge$, $\vee$), inequality symbols ($<, \leqslant, \geqslant, >$), the equality symbol ($=$), unary predicates like bound, isBlank, and isIRI, plus other features (see [6] for a complete list). In this paper, we restrict to the fragment where constraints are a boolean combination of terms constructed by using $=$ and bound, that is:

- If $?x, ?y \in V$ and $c \in I \cup L$, then bound($?x$), $?x = c$ and $?x = ?y$ are built-in conditions;
- If C_1 and C_2 are built-in conditions, then $(\neg C_1)$, $(C_1 \vee C_2)$ and $(C_1 \wedge C_2)$ are built-in conditions.

Let P be a pattern. We use $var(P)$ to denote the set of variables occurring in P. In particular, if t is a triple pattern, then $var(t)$ denotes the set of variables occurring in the components of t. Similarly, for a built-in condition C, we use $var(C)$ to denote the set of variables occurring in C.

Let P be a pattern and let $S \subset V$ a finite set of variables. A SELECT *query* is of the form SELECT$_S(P)$. All SELECT queries are on the top of patterns. Note that SELECT can be nested as subquery in SPARQL 1.1 [9]. In this paper, we don't discuss SELECT as subquery following SPARQL 1.0.

Three-valued and two-valued semantics of SPARQL There are two formalizations of semantics of SPARQL, namely, *set-based semantics* and *bag-based semantics*. The semantics we use is set-based [5], whereas the semantics of real SPARQL is bag-based [6].

Set-based semantics

Semantically based on sets, a mapping μ from an infinite set of variables V to a set of RDF term U is a partial function. For each triple $t = (s, p, o)$, we use $\mu(t)$ to denote the triple obtained by replacing the variables in t according to μ. That is, $dom(u)$ is a subset of V. We say two mapping μ_1, μ_2 are *compatible*, written by $\mu_1 \sim \mu_2$, if they agree on all shared

* Corresponding author (email: xiaowangzhang@tju.edu.cn)

Table A1 Truth value table

$\mu(C_1)$	$\mu(C_2)$	$\mu(C_1) \wedge \mu(C_2)$	$\mu(C_1) \vee \mu(C_2)$
true	*true*	*true*	*true*
true	*false*	*false*	*true*
true	*error*	*error*	*true*
false	*true*	*false*	*true*
false	*false*	*false*	*false*
false	*error*	*false*	*error*
error	*true*	*error*	*true*
error	*false*	*false*	*error*
error	*error*	*error*	*error*

variables, i.e., if $\mu_1(?x) = \mu_2(?x)$ for all $?x \in dom(\mu_1) \cap dom(\mu_2)$. That is to say, $\mu_1 \cup \mu_2$ is also a mapping. Intuitively, μ_1 and μ_2 are compatibles if μ_1 can be extended with μ_2 to obtain a new mapping, and vice versa.

Let Ω_1, Ω_2 be two sets of mappings. We define the join of, the union of, and the difference between Ω_1 and Ω_2 as follows:

- $\Omega_1 \cup \Omega_2 = \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\}$;
- $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 : \mu_1 \sim \mu_2\}$;
- $\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \forall \mu_2 \in \Omega_2 : \mu_1 \not\sim \mu_2\}$;
- $\Omega_1 \mathbin{⟕} \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2)$;
- $\pi_S(\Omega) = \{\mu_1 \mid \exists \mu_2 : \mu_1 \cup \mu_2 \in \Omega \text{ and } dom(\mu_1) \subseteq S \text{ and } dom(\mu_2) \cap S = \emptyset\}$.

Let G be an RDF graph, t a triple pattern, P, P_1, P_2 patterns and C a filter condition. We define

$$\begin{aligned} [\![t]\!]_G &= \{\mu \mid dom(\mu) = var(t) \text{ and } \mu(t) \in G\}; \\ [\![P_1 \text{ AND } P_2]\!]_G &= [\![P_1]\!]_G \bowtie [\![P_2]\!]_G; \\ [\![P_1 \text{ UNION } P_2]\!]_G &= [\![P_1]\!]_G \cup [\![P_2]\!]_G; \\ [\![P_1 \text{ OPT } P_2]\!]_G &= [\![P_1]\!]_G \mathbin{⟕} [\![P_2]\!]_G; \\ [\![P \text{ FILTER } C]\!]_G &= \{\mu \in [\![P]\!]_G \mid \mu \models C\}; \\ [\![\text{SELECT}_S(P)]\!]_G &= \pi_S([\![P]\!]_G). \end{aligned} \tag{A1}$$

Three-valued set-based semantics

The three-valued semantics officially recommended by W3C of filter expressions goes as follows. Given a mapping μ and a constraint C, we say that the evaluation of C against μ, denoted by $\mu(C)$, is defined in a three-valued logic with truth values $\{true, false, error\}$ as follows (see [1]):

- If C is an atomic constraint, then

(1) If $var(C) \subseteq dom(\mu)$ then $\mu(C) = true$ when

  (a) C is $?x = c$ and $\mu(?x) = c$; or

  (b) C is $?x = ?y$ and $\mu(?x) = \mu(?y)$; or

  (c) C is bound($?x$);

  and $\mu(C) = false$ otherwise.

(2) If $var(C) \not\subseteq dom(\mu)$ then if C is bound($?x$) then $\mu(C) = false$ else $\mu(C) = error$.

- If C is complex constraint, then $\mu(C)$ is defined as follows: (1) $\neg\mu(C) = true$ (*false* or *error*) if $\mu(C) = false$ (*true* or *error*) respectively; and (2) the boolean constraints are defined under the three-valued semantics in Table A1.

A mapping μ *satisfies* a constraint C, denoted by $\mu \models C$, if and only if $\mu(C) = true$.

Two-valued set-based semantics

The two-valued semantics [5] (classically, *true* and *false* are truth values) is different from the three-valued semantics only in characterizing constraints. The two-valued semantics of them goes as follows. Given a mapping μ and a constraint C, the evaluation of C against μ, denoted by $\mu(C)$, is defined in a two-valued logic with truth values $\{true, false\}$ as follows:

- If C is an atomic constraint, then

(1) If $var(C) \subseteq dom(\mu)$ then $\mu(C) = true$ when

  (a) C is $?x = c$ and $\mu(?x) = c$; or

  (b) C is $?x = ?y$ and $\mu(?x) = \mu(?y)$; or

  (c) C is bound($?x$);

  and $\mu(C) = false$ otherwise.

Table A2 Truth value table under the two-valued semantics

$\mu(C_1)$	$\mu(C_2)$	$\mu(C_1) \wedge \mu(C_2)$	$\mu(C_1) \vee \mu(C_2)$
true	*true*	*true*	*true*
true	*false*	*false*	*true*
false	*true*	*false*	*true*
false	*false*	*false*	*false*

(2) If $var(C) \not\subseteq dom(\mu)$ then $\mu(C) = false$.

• If C is complex constraint, then $\mu(C)$ is defined as follows:

(1) $\neg\mu(C) = true$ (or *false*) if $\mu(C) = false$ (or *true*) respectively; and

(2) the boolean constraints are defined under the two-valued semantics in Table A2.

A mapping μ *satisfies* a constraint C, denoted by $\mu \models_2 C$, if and only if $\mu(C) = true$.

Let P be a pattern and G an RDF graph. We use $[\![P]\!]^2_G$ to denote the collection of all mappings satisfying P under the two-valued semantics.

In SPARQL, two semantics are slightly different in constraints $\neg(?x = c)$ and $\neg(?x = ?y)$.

• Under the three-valued semantics,

(1) $\mu \models \neg(?x = c)$ if $?x \in dom(\mu)$ and $\mu(x) \neq c$;

(2) $\mu \models \neg(?x = ?y)$ if $\{?x, ?y\} \subseteq dom(\mu)$ and $\mu(x) \neq \mu(?y)$.

• Under the two-valued semantics,

(1) $\mu \models \neg(?x = c)$ if either $?x \in dom(\mu)$ and $\mu(x) \neq c$ or $?x \notin dom(\mu)$;

(2) $\mu \models \neg(?x = ?y)$ if $\{?x, ?y\} \subseteq dom(\mu)$ and $\mu(x) \neq \mu(?y)$ or $\{?x, ?y\} \not\subseteq dom(\mu)$.

For simplification, we directly use $?x \neq c$ to denote $\neg(?x = c)$ and $?x \neq ?y$ to denote $\neg(?x = ?y)$ in this paper.

Finally, given two patterns P_1, P_2, we say P_1 is *subsumed* in P_2 under the three-valued semantics, written by $P_1 \subseteq P_2$, if for every RDF graph G, $[\![P_1]\!]_G \subseteq [\![P_2]\!]_G$. Analogously, we say P_1 is *subsumed* in P_2 under the two-valued semantics, written by $P_1 \subseteq_2 P_2$, if for every RDF graph G, $[\![P_1]\!]^2_G \subseteq [\![P_2]\!]^2_G$.

Well-designed patterns *Well-designed patterns* are introduced to characterize weak monotonicity [5].

A UNION-free pattern P is *well-designed* if the followings hold:

• P is safe, that is, for every sub-pattern Q of form (Q FILTER C) of P, $var(C) \subseteq var(Q)$;

• for every sub-pattern Q of form (Q_1 OPT Q_2) of P and for every variable ?x occurring in P, the following condition hold: If $?x$ occurs both inside Q_2 and outside Q, then it also occurs in Q_1.

A well-designed pattern P is of the following form (called *UNION norm form*):

$$Q_1 \text{ UNION} \ldots \text{UNION } Q_m, \tag{A2}$$

where each Q_i $(i = 1, 2, \ldots, m)$ is UNION-free well-designed pattern.

Let P be a pattern. We use $\Delta(P)$ to denote the least reduction of P (defined in [5]) as follows:

• $\Delta(t) := t$;
• $\Delta(P_1 \text{ UNION } P_2) := \Delta(P_1) \text{ UNION } \Delta(P_2)$;
• $\Delta(P_1 \text{ AND } P_2) := \Delta(P_1) \text{ AND } \Delta(P_2)$;
• $\Delta(P_1 \text{ OPT } P_2) := \Delta(P_1)$;
• $\Delta(P_1 \text{ FILTER } C) := \Delta(P_1) \text{ FILTER } C$.

Appendix B Proofs

Proposition 1. The two-valued pattern semantics is equivalent to the three-valued pattern semantics in SPARQL$^+$.

Proof. We only need to prove that for any pattern P in SPARQL$^+$, for any RDF graph G, we have $[\![P]\!]_G = [\![P]\!]^2_G$.

By induction on the structure of P.

If P is a triple pattern that this claim directly follows definition since triple patterns do not contain any constraint.

If P is of the form P_1 AND P_2, P_1 OPT P_2, and P_1 UNION P_2, then this claim holds by induction.

If P is of the form P_1 FILTER C then this claim follows from the following claim.

Claim 1. Let C be a positive constraint. For all mappings μ, $\mu \models C$ if and only if $\mu \models_2 C$.

We will prove this claim by induction on the structure of C.

• (basic step) Let's consider three kinds of atomic constraints.

(1) $\mu \models ?x = c \Leftrightarrow \mu(?x = c) = true \Leftrightarrow \mu(?x) = c \Leftrightarrow \mu \models_2 ?x = c$;

(2) $\mu \models ?x = ?y \Leftrightarrow \mu(?x = ?y) = true \Leftrightarrow \mu(?x) = \mu(?y) \Leftrightarrow \mu \models_2 ?x = ?y$;

(3) $\mu \models \text{bound}(?x) \Leftrightarrow \mu(\text{bound}(?x)) = \mathit{true} \Leftrightarrow ?x \in \mathit{dom}(\mu) \Leftrightarrow \mu \models_2 \text{bound}(?x)$.

- (Inductive step) Assume that $\mu \models C_i \Leftrightarrow \mu \models_2 C_i$ $(i = 1, 2)$.

(1) $\mu \models C_1 \wedge C_2 \Leftrightarrow \mu \models C_1$ and $\mu \models C_2 \Leftrightarrow \mu \models_2 C_1$ and $\mu \models_2 C_2 \Leftrightarrow \mu \models_2 C_1 \wedge C_2$;

(2) $\mu \models C_1 \vee C_2 \Leftrightarrow \mu \models C_1$ or $\mu \models C_2 \Leftrightarrow \mu \models_2 C_1$ or $\mu \models_2 C_2 \Leftrightarrow \mu \models_2 C_1 \vee C_2$.

Here $\Leftrightarrow$ means "if and only if". □

Proposition 2. The two-valued pattern semantics is not equivalent to the three-valued pattern semantics in SPARQL$^{\text{b}}$.

Proof. We only need to prove that there exists some pattern P in SPARQL$^{\text{b}}$ and some RDF graph G such that $[\![P]\!]_G \neq [\![P]\!]^2_G$.

Consider a pattern P of the form $(?x, p, ?y)$ FILTER $?z \neq c$ and an RDF graph $G = \{(a, p, b)\}$. Thus $[\![P]\!]_G = \emptyset$ while $[\![P]\!]^2_G = \{(?x \to a, ?y \to b)\}$. □

Proposition 3. SPARQL is expressible in SPARQL$^{\text{b}}$ under the two semantics.

Proof. To prove Proposition 3, we define $\gamma(P)$ as a new pattern obtained from P by applying the following rule:

$$\text{substituting } \neg \text{ bound } (?x) \text{ by } ?x \neq ?x.$$

We can conclude the following claim:

Claim 2. For any pattern P in SPARQL, for any RDF graph G, we have $[\![P]\!]^2_G = [\![\gamma(P)]\!]^2_G$.

And the following claim that the expressivity of SPARQL under the two-valued semantics is the same as the expressivity of SPARQL under the three-valued semantics [8, Proposition 17].

Claim 3. [8] For all SPARQL pattern P, there exists some SPARQL pattern Q such that for any graph G, $[\![P]\!]_G = [\![Q]\!]^2_G$.

We only need to show that for any pattern P of the form Q FILTER $\neg$bound$(?x)$, for any RDF graph G, $[\![P]\!]^2_G = [\![\gamma(P)]\!]^2_G$, that is, $[\![Q \text{ FILTER } \neg \text{ bound } (?x)]\!]^2_G = [\![Q \text{ FILTER } ?x \neq ?x]\!]^2_G$. Let $\mu \in [\![Q \text{ FILTER } \neg \text{ bound } (?x)]\!]^2_G$. Thus $\mu \in [\![Q]\!]^2_G$. Since $?x \notin \mathit{dom}(\mu)$, $\mu \models ?x \neq ?x$ under the two-valued semantics. Then $[\![Q \text{ FILTER } \neg \text{ bound } (?x)]\!]^2_G \subseteq [\![Q \text{ FILTER } ?x \neq ?x]\!]^2_G$. On the other hand, $\mu \in [\![Q \text{ FILTER } ?x \neq ?x]\!]^2_G$. Thus $\mu \in [\![Q]\!]^2_G$. Since $\mu \models ?x \neq ?x$ under the two-valued semantics, $?x \notin \mathit{dom}(\mu)$. Then $[\![Q \text{ FILTER } ?x \neq ?x]\!]^2_G \subseteq [\![Q \text{ FILTER } \neg \text{bound } (?x)]\!]^2_G$. Therefore, $[\![Q \text{ FILTER } \neg \text{bound } (?x)]\!]^2_G = [\![Q \text{ FILTER } ?x \neq ?x]\!]^2_G$. □

By Claim 2, for any pattern P in SPARQL, for any RDF graph G, we have $[\![P]\!]^2_G = [\![\gamma(P)]\!]^2_G$. We conclude that for any pattern P in SPARQL, there exists some pattern Q' (here Q' is $\gamma(P)$) in SPARQL$^{\text{b}}$ such that $[\![P]\!]^2_G = [\![Q']\!]^2_G$ for any RDF graph G since $\gamma(P)$ is a pattern in SPARQL$^{\text{b}}$. Then SPARQL is expressible in SPARQL$^{\text{b}}$ under the two-valued semantics. We also have that SPARQL is expressible in SPARQL$^{\text{b}}$ under the three-valued semantics by Claim 3.

Therefore, SPARQL is expressible in SPARQL$^{\text{b}}$ under two semantics. □

Proposition 4. $\mathcal{AFU}$ is monotonic under the two-valued semantics.

Proof. Firstly, we introduce a claim as follows:

Claim 4. Let Ω_i and Ω'_i be a set of mappings $(i = 1, 2)$. If $\Omega_i \subseteq \Omega'_i$ then the followings hold.

- $\Omega_1 \cup \Omega_2 \subseteq \Omega'_1 \cup \Omega'_2$;
- $\Omega_1 \bowtie \Omega_2 \subseteq \Omega'_1 \bowtie \Omega'_2$.
- $$\begin{aligned} \Omega_1 \cup \Omega_2 &= \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\}; \\ &\subseteq \{\mu \mid \mu \in \Omega'_1 \text{ or } \mu \in \Omega'_2\}; \\ &= \Omega'_1 \cup \Omega'_2. \end{aligned}$$
- $$\begin{aligned} \Omega_1 \bowtie \Omega_2 &= \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ and } \mu_1 \sim \mu_2\}; \\ &= \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega'_1 \text{ and } \mu_2 \in \Omega'_2 \text{ and } \mu_1 \sim \mu_2\}; \\ &= \Omega'_1 \bowtie \Omega'_2. \end{aligned}$$

By Claim 4, we will prove Proposition 4.

Next, we only need to show that for any pattern P in $\mathcal{AFU}$, for any two RDF graphs G_1 and G_2, $[\![P]\!]_{G_1} \subseteq [\![P]\!]_{G_2}$ if $G_1 \subseteq G_2$.

By induction on the structure of P.

- If P is a triple pattern then this claim directly follows by definition. That is to say, $[\![P]\!]_{G_1} \subseteq [\![P]\!]_{G_2}$ if $G_1 \subseteq G_2$.
- If P is of the form P_1 UNION P_2 then $[\![P_i]\!]_{G_1} \subseteq [\![P_i]\!]_{G_2}$ $(i = 1, 2)$ if $G_1 \subseteq G_2$. By Claim 4, we can conclude that $[\![P_1 \text{ UNION } P_2]\!]_{G_1} \subseteq [\![P_1 \text{ UNION } P_2]\!]_{G_2}$.
- If P is of the form P_1 AND P_2 then $[\![P_i]\!]_{G_1} \subseteq [\![P_i]\!]_{G_2}$ $(i = 1, 2)$ if $G_1 \subseteq G_2$. By Claim 4, we can conclude that $[\![P_1 \text{ AND } P_2]\!]_{G_1} \subseteq [\![P_1 \text{ AND } P_2]\!]_{G_2}$.
- Finally, if P is of the form P_1 FILTER C then $[\![P_1]\!]_{G_1} \subseteq [\![P_1]\!]_{G_2}$ if $G_1 \subseteq G_2$. Let's discuss six cases of C.

(1) If C is of the form bound($?x$) then

$$\begin{aligned}
[\![P]\!]_{G_1} &= [\![P_1 \text{ FILTER bound}(?x)]\!]_{G_1};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \mu \models \text{bound}(?x)\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } ?x \in dom(\mu)\};\\
&\subseteq \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } ?x \in dom(\mu)\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } \mu \models \text{bound}(?x)\};\\
&= [\![P_1 \text{ FILTER bound}(?x)]\!]_{G_2};\\
&= [\![P]\!]_{G_2}.
\end{aligned}$$

(2) If C is of the form $?x = c$ then

$$\begin{aligned}
[\![P]\!]_{G_1} &= [\![P_1 \text{ FILTER } ?x = c]\!]_{G_2};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \mu \models ?x = c\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } ?x \in dom(\mu) \text{ and } \mu(?x) = c\};\\
&\subseteq \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } ?x \in dom(\mu) \text{ and } \mu(?x) = c\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } ?x \in dom(\mu) \text{ and } \mu(?x) = c\};\\
&= [\![P_1 \text{ FILTER } ?x = c]\!]_{G_2};\\
&= [\![P]\!]_{G_2}.
\end{aligned}$$

(3) If C is of the form $?x = ?y$ then

$$\begin{aligned}
[\![P]\!]_{G_1} &= [\![P_1 \text{ FILTER } ?x = ?y]\!]_{G_1};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \mu \models ?x = ?y\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \{?x, ?y\} \subseteq dom(\mu) \text{ and } \mu(?x) = \mu(?y)\};\\
&\subseteq \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \{?x, ?y\} \subseteq dom(\mu) \text{ and } \mu(?x) = \mu(?y)\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } \mu \models ?x = ?y\};\\
&= [\![P_1 \text{ FILTER } ?x = ?y]\!]_{G_2};\\
&= [\![P]\!]_{G_2}.
\end{aligned}$$

(4) If C is of the form $C_1 \vee C_2$ then

$$\begin{aligned}
[\![P]\!]_{G_1} &= [\![P_1 \text{ FILTER } C_1 \vee C_2]\!]_{G_1};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \mu \models C_1 \vee C_2\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } (\mu \models C_1 \text{ or } \mu \models C_2\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \mu \models C_1\} \cup \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \mu \models C_2\};\\
&\subseteq \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } \mu \models C_1\} \cup \{\mu \mid \mu \in [\![P]\!]_{G_2} \text{ and } \mu \models C_2\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } (\mu \models C_1 \text{ or } \mu \models C_2\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } \mu \models C_1 \vee C_2\};\\
&= [\![P \text{ FILTER } C_1 \vee C_2]\!]_{G_2};\\
&= [\![P]\!]_{G_2}.
\end{aligned}$$

(5) If C is of the form $C_1 \wedge C_2$ then

$$\begin{aligned}
[\![P]\!]_{G_1} &= [\![P_1 \text{ FILTER } C_1 \wedge C_2]\!]_{G_1}\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \mu \models C_1 \wedge C_2\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } (\mu \models C_1 \text{ and } \mu \models C_2\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \text{ and } \mu \models C_1\} \cap \{\mu \mid \mu \in [\![P_1]\!]_{G_1} \,\&\, \mu \models C_2\};\\
&\subseteq \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } \mu \models C_1\} \cap \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \,\&\, \mu \models C_2\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } (\mu \models C_1 \text{ and } \mu \models C_2\};\\
&= \{\mu \mid \mu \in [\![P_1]\!]_{G_2} \text{ and } \mu \models C_1 \wedge C_2\};\\
&= [\![P_1 \text{ FILTER } C_1 \wedge C_2]\!]_{G_2};\\
&= [\![P]\!]_{G_2}.
\end{aligned}$$

(6) If C is of the form $\neg C_1$ then:

$$\begin{aligned}
[\![P]\!]_{G_1} &= [\![P \text{ FILTER } \neg C_1]\!]_{G_1};\\
&= \{\mu \mid \mu \in [\![P]\!]_{G_1} \text{ and } \mu \models \neg C_1\};\\
&= \{\mu \mid \mu \in [\![P]\!]_{G_1} \text{ and } \mu \not\models C_1\};\\
&\subseteq \{\mu \mid \mu \in [\![P]\!]_{G_2} \text{ and } \mu \not\models C_1\};\\
&= \{\mu \mid \mu \in [\![P]\!]_{G_2} \text{ and } \mu \models \neg C_1\};\\
&= [\![P \text{ FILTER } \neg C_1]\!]_{G_2};\\
&= [\![P]\!]_{G_2}. \quad \square
\end{aligned}$$

Proposition 5. Any fragment consisting of OPT is non-monotonic under the two semantics.

Proof. Consider a pattern $P = (?x, p, ?y)$ OPT $(?y, q, ?z)$ and two RDF graphs $G_1 = \{(a, p, b)\}$ and $G_2 = \{(a, p, b), (b, q, c)\}$. We have the followings.

- $[\![P]\!]_{G_1} = \{\mu_1\}$ where $\mu_1(?x) = a$ and $\mu_1(?y) = b$;
- $[\![P]\!]_{G_2} = \{\mu_2\}$ where $\mu_2(?x) = a$, $\mu_2(?y) = b$, and $\mu_2(?z) = c$.

We have both $[\![P]\!]_{G_1} \not\subseteq [\![P]\!]_{G_2}$ and $[\![P]\!]^2_{G_1} \not\subseteq [\![P]\!]^2_{G_2}$ while $G_1 \subseteq G_2$ while since $\mu_1 \neq \mu_2$. □

Corollary 1. Both OPT-free SPARQL$^+$ and OPT-free SPARQL$^{\mathrm{b}}$ are monotonic under the two semantics.

Proof. By Proposition 4 and the fact that $\mathcal{AFU}$ is monotonic under the three-valued semantics [8], we know that OPT-free SPARQL$^+$ and OPT-free SPARQL$^{\mathrm{b}}$ is monotonic since both OPT-free SPARQL$^+$ and OPT-free SPARQL$^{\mathrm{b}}$ are fragments of $\mathcal{AFU}$. □

Appendix B.1 Weak monotonicity and well-designed patterns

Proposition 6. For any pattern P, if P is weak monotonic then P is non-optionally monotonic under the two semantics, but not vice versa.

Proof. Let P be a pattern. Given two RDF graphs G_1 and G_2, $G_1 \subseteq G_2$ implies $[\![P]\!]_{G_1} \sqsubseteq [\![P]\!]_{G_2}$ since P is weak monotonic. That is, for any mapping $\mu_1 \in [\![P]\!]_{G_1}$, there exists some mapping $\mu_2 \in [\![P]\!]_{G_2}$ such that $\mu_1 \sqsubseteq \mu_2$. Let μ be a mapping in $[\![\Delta(P)]\!]_{G_1}$ and $\mu \sqsubseteq \mu_1$. Since $\mu_1 \sqsubseteq \mu_2$ and $\mu \sqsubseteq \mu_1$, $\mu \sqsubseteq \mu_2$. Therefore, P is non-optionally monotonic.

Consider a pattern $P = ((?x, p, ?y)$ OPT $(?y, q, ?z))$ OPT $(?z, r, ?w)$ and two RDF graphs $G_1 = \{(a, p, b), (d, r, e)\}$, $G_2 = \{(a, p, b), (d, r, e), (b, q, c)\})$. We have that P is non-optionally monotonic while P is not weak monotonic since $\mu_1 \neq \mu_2$. □

Proposition 7. For any pattern P in $\mathcal{AFU}$, P is monotonic, weak monotonic, and non-optionally monotonic under two semantics.

Proof. Since P is OPT-free, $\Delta(P) = P$. Thus P is weak monotonic iff P is non-optionally monotonic.

Next, we only need to show that P is weak monotonic. By Proposition 4, P is monotonic. That is, for any two RDF graphs G_1 and G_2 with $G_1 \subseteq G_2$, for any mapping $\mu_1 \in [\![P]\!]_{G_1}$, $\mu_1 \in [\![P]\!]_{G_2}$ and then, $\mu_1 \in [\![P]\!]_{G_2}$ and $\mu_1 \sqsubseteq \mu_1$. Therefore, P is weak monotonic. □

Proposition 8. Each well-designed pattern is weak monotonic and non-optionally monotonic under two semantics.

Proof. Since the three-valued semantics is equivalent to the two valued-semantics by Proposition 2, we can conclude that P is weak monotonic under the two-valued semantics. Next, we need to show that P is non-optionally monotonic under two semantics. For any two RDF graphs G_1 and G_2 with $G_1 \subseteq G_2$, for any mapping $\mu_1 \in [\![P]\!]_{G_1}$, there exists some mapping $\mu_2 \in [\![P]\!]_{G_2}$ such that $\mu_1 \sqsubseteq \mu_2$ and then, for any mapping $\mu \in [\![\Delta(P)]\!]_{G_1}$, $\mu \sqsubseteq \mu_1$, we have $\mu \sqsubseteq \mu_2$ since $\mu_1 \sqsubseteq \mu_2$. Therefore, P is non-optionally monotonic. □

Proposition 9. $\mathcal{OU}$ is non-optionally monotonic under the two semantics.

Proof. Firstly, we introduce the following claim:

Claim 5. For every pattern P in $\mathcal{OU}$, the followings hold.

- $\Delta(P)$ has of the following form: $\Delta(P) = t_1$ UNION $\cdots$ UNION t_n; where t_i $(i = 1, 2, \ldots, n)$ is a triple pattern.
- For every RDF graph G, for every mapping $\mu \in [\![P]\!]_G$, there exists some mapping $\mu' \in [\![t_i]\!]_G$ for some $i \in \{1, \ldots, n\}$ such that $\mu' \sqsubseteq \mu$.
- For every RDF graph G, for every mapping $\mu \in [\![t_i]\!]_G$ with $i \in \{1, \ldots, n\}$, there exists some mapping $\mu' \in [\![P]\!]_G$ such that $\mu \sqsubseteq \mu'$.

Those claims directly follow the definition of $\Delta(P)$.

Let P be a pattern in $\mathcal{OU}$ and let $\Delta(P)$ be of the following form: t_1 UNION ... UNION t_n by the first item of Claim 5.

Given two RDF graphs G_1 and G_2 with $G_1 \subseteq G_2$, let μ_1 be a mapping $[\![P]\!]_{G_1}$, there exists some mapping $\mu'_1 \in [\![t_i]\!]_{G_1}$ for some $i \in \{1, \ldots, n\}$ such that $\mu' \sqsubseteq \mu$ by the second item of Claim 5. Thus $\mu'_1 \in [\![t_i]\!]_{G_2}$ since each triple pattern is monotonic. Then there exists some mapping $\mu'_1 \in [\![P]\!]_{G_2}$ by the third item of Claim 5. Therefore, P is non-optionally monotonic. □

Proposition 10. $\mathcal{AO}$ and $\mathcal{FO}$ are not non-optionally monotonic under the two semantics.

Proof. Consider a pattern $P_1 = ((?x, p, ?y)$ OPT $(?y, q, ?z))$ AND $(?x, r, ?z)$ and two RDF graphs $G_1 = \{(a, p, b), (a, r, d)\}$ and $G_2 = G_1 \cup \{(b, q, c)\}$. We have $[\![P_1]\!]_{G_1} = \{\mu_1\}$ where $\mu_1(?x) = a$, $\mu_1(?y) = b$, and $\mu_1(?z) = d$ while $[\![P_1]\!]_{G_2} = \emptyset$. Then P_1 is not non-optionally monotonic since $[\![\Delta(P_1)]\!]_{G_1} = \{\mu_1\}$.

Consider a pattern $P_2 = ((?x, p, ?y)$ OPT $(?y, q, ?z))$ FILTER $\neg$ bound $(?z)$ and two RDF graphs $G_3 = \{(a, p, b))\}$ and $G_4 = G_2 \cup \{(b, q, c)\}$. We have $[\![P_2]\!]_{G_3} = \{\mu_2\}$ where $\mu_2(?x) = a$ and $\mu_2(?y) = b$ while $[\![P_2]\!]_{G_2} = \emptyset$. Then P_2 is not non-optionally monotonic under two semantics since $[\![\Delta(P_2)]\!]_{G_1} = \{\mu_1\}$. □

Proposition 11. Let $\mathcal{F}$ be a fragment of SPARQL. If DIFF is expressible in $\mathcal{F}$ then $\mathcal{F}$ is not non-optionally monotonic under the two semantics.

Proof. Consider a pattern $P = (?x, p, ?y)$ DIFF $(?x, q, ?y)$, there exists some pattern Q in $\mathcal{F}$ such that $[\![P]\!]_G = [\![Q]\!]_G$ for any RDF graph G. Given two RDF graphs $G_1 = \{(a, p, b)\}$ and $G_2 = G_1 \cup \{(a, q, b)\}$, we have $[\![P]\!]_{G_1} = \{\mu\}$ where $\mu(?x) = a$ and $\mu(?y) = b$ while $[\![P]\!]_{G_2} = \emptyset$. That is, $[\![Q]\!]_{G_1} = \{\mu\}$ where $\mu(?x) = a$ and $\mu(?y) = b$ while $[\![Q]\!]_{G_2} = \emptyset$ since $[\![P]\!]_G = [\![Q]\!]_G$ for any RDF graph G. Since $[\![Q]\!]_{G_2} = \emptyset$, P is not non-optionally monotonic under the two semantics. □

Proposition 12. $\mathcal{FO}^+$ and $\mathcal{FO}^{\mathrm{b}}$ are not non-optionally monotonic under the two semantics.

Proof. We know that DIFF is expressible in both $\mathcal{FO}^+$ and $\mathcal{FO}^{\mathrm{b}}$ since the constraint of the form $?x = c$ is allowed [7]. By Proposition 11, we can conclude that $\mathcal{FO}^+$ and $\mathcal{FO}^{\mathrm{b}}$ are not non-optionally monotonic under the two semantics. □

References

1 Angles, R. & Gutierrez, C. The expressive power of SPARQL. In: *Proc. of Int. Semantic Web Conf. (ISWC-08)*, 2008, pp.114 –129.

2 Arenas, M., Gutierrez, C., P. Miranker, D., Pérez, J. & Sequeda, J. Querying semantic data on the web? *SIGMOD Record*, 2012, 41(4): 6 –17.

3 Kontchakov, R. and V. Kostylev, E. On expressibility of non-monotone operators in SPARQL. In: *Proc. of Int. Conf. Principles of Knowledge Representation and Reasoning (KR-16)*, 2016, pp.369–379.

4 Kaminski, M. & V. Kostylev, E. (2016) Beyond well-designed SPARQL. In: *Proc. of Int. Conf. Database Theory (ICDT-16)*, 2016, pp.5:1-5:18.

5 Pérez, J., Arenas, M. & Gutierrez, C. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 2009, 34(3):1–45

6 Prudhommeaux, E. & Seaborne, A. SPARQL query language for RDF. *W3C recommendation.* 2008.

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

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

9 Zhang, X., Van den Bussche, J., Wang, K., & Wang, Z. On the satisfiability problem of patterns in SPARQL 1.1. In: *Proc. of AAAI. on Artificial Intelligence (AAAI-18)*, 2018, to appear.