Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

On Expressibility of Non-Monotone Operators in SPARQL

Roman Kontchakov
Birkbeck, University of London

Egor V. Kostylev
University of Oxford

Abstract

SPARQL, a query language for RDF graphs, is one of the key technologies for the Semantic Web. The expressivity and complexity of various fragments of SPARQL have been studied extensively. It is usually assumed that the optional matching operator OPTIONAL has only two graph patterns as arguments. The specification of SPARQL, however, defines it as a ternary operator, with an additional filter condition. We address the problem of expressibility of the full ternary OPTIONAL via the simplified binary version and show that it is possible, but only with an exponential blowup in the size of the query (under common complexity-theoretic assumptions). We also study expressibility of other non-monotone SPARQL operators via optional matching and each other.

Introduction

The Resource Description Framework (RDF) (Cyganiak, Wood, and Lanthaler 2014; Hayes and Patel-Schneider 2014) is the W3C standard for representing data and knowledge on the Web. The RDF data model is based on labelled graphs, which are sets of triples of internationalised resource identifiers (IRIs) and literals (strings, numbers, etc.).

SPARQL is the standard query language for RDF graphs. Since the first W3C recommendation (Prud'hommeaux and Seaborne 2008), it has been recognised as a key technology for the Semantic Web. The current version, SPARQL 1.1 (Harris and Seaborne 2013), is supported by a number of academic and commercial query engines, such as Apache Jena (jena.apache.org), Sesame (rdf4j.org), and OpenLink Virtuoso (virtuoso.openlinksw.com).

The seminal work of Pérez, Arenas, and Gutierrez (2009) laid the theoretical foundations of SPARQL. Now the complexity of query evaluation is quite well understood (Schmidt, Meier, and Lausen 2010; Losemann and Martens 2013; Arenas, Conca, and Pérez 2012; Kaminski and Kostylev 2016; Kostylev et al. 2015); algorithms and optimisation techniques have been developed (Letelier et al. 2013; Pichler and Skritek 2014; Zhang and Van den Bussche 2014b). SPARQL entailment regimes, a way of extending queries with OWL reasoning capabilities, have been studied by (Kollia and Glimm 2013; Kostylev and Cuenca Grau 2014; Ahmetaj et al. 2015; Arenas, Gottlob, and Pieris 2014; Kontchakov et al. 2014; Bischof et al. 2014). Other recently considered issues include federation (Buil Aranda, Polleres, and Umbrich 2014; Buil-Aranda, Arenas, and Corcho 2011) and provenance (Geerts et al. 2013; Halpin and Cheney 2014). These studies have had a great impact on the community and influenced the SPARQL specification.

Expressive power of SPARQL and its fragments is another fundamental problem that has been studied extensively: it is known, for example, that SPARQL as a whole has the same expressive power as first-order logic and relational algebra (Angles and Gutierrez 2008; Polleres and Wallner 2013; Kostylev, Reutter, and Ugarte 2015) and most of the SPARQL operators are primitive, that is, not expressible via each other (Zhang and Van den Bussche 2014a). The present paper continues this line of research and concentrates on the expressive power of non-monotone operators in SPARQL.

The core of the SPARQL algebra, which originates in SPARQL 1.0, consists of operators $\textsc{Join}$, $\textsc{Union}$, $\textsc{Filter}_F$, $\textsc{Proj}_V$ and $\textsc{Opt}_F$. The first four roughly correspond to the positive relational algebra with inequalities, a well-studied formalism in relational databases. The last, optional matching, is a distinctive feature of SPARQL in comparison to SQL. This operator was introduced to *"not reject the solutions because some part of the query pattern does not match"* (Prud'hommeaux and Seaborne 2008), and accounts naturally for the open-world assumption and fundamental incompleteness of the information on the Web. To illustrate optional matching, consider the SPARQL algebra expression (called a *pattern*)

$$\{(?p,\ \texttt{a},\ \texttt{Prof})\}\ \textsc{Opt}_{?d \neq \texttt{CS}}\ \{(?p,\ \texttt{dept},\ ?d)\}, \qquad (1)$$

which retrieves professors and all their departments; departments, however, are optional—if the graph does not contain information about any departments for a professor, she/he is still retrieved but variable $?d$ is left *unbound* in the answer; moreover, the CS department is out of interest of this query, and so, professors affiliated with CS are either paired with their other departments, or, if there are none, then $?d$ is unbound as well. For instance, on the graph with triples

$(\texttt{Ivanov, a, Prof})$,
$(\texttt{Petrov, a, Prof})$, $(\texttt{Petrov, dept, CS})$,
$(\texttt{Sidorov, a, Prof})$, $(\texttt{Sidorov, dept, Maths})$,

query (1) has three answers (called *solution mappings*):

$$\{\,?p \mapsto \texttt{Ivanov}\,\}, \quad \{\,?p \mapsto \texttt{Petrov}\,\},$$
$$\{\,?p \mapsto \texttt{Sidorov}, \ ?d \mapsto \texttt{Maths}\,\}.$$

The optional matching operator OPT_F is non-monotone, and it adds considerable expressive power to the language. In fact, it allows for translation of the first-order logic negation and makes SPARQL expressively equivalent to the full relational algebra. This power, however, comes at a price: evaluation of SPARQL with OPT_F is PSPACE-complete, in contrast to NP-completeness of the positive fragment (Pérez, Arenas, and Gutierrez 2009).

In addition to OPT_F, the SPARQL algebra includes non-monotone operators DIFF_F and MINUS, and the subquery construction not exists in filters. Although DIFF_F has no counterpart in the syntax, it is used in the definition of OPT_F, while MINUS and not exists were introduced in SPARQL 1.1 to extend the querying toolkit. Also, the binary operators $\text{OPT}_\top$ and $\text{DIFF}_\top$, which have no filtering conditions such as $?d \neq \texttt{CS}$ in (1), have been used in theoretical studies of SPARQL instead of their full ternary counterparts. This has been justified by the common belief that the ternary operators are linearly expressible via the binary ones (Angles and Gutierrez 2008).

In this paper, we compare the expressive power of the non-monotone operators. We define two notions of expressibility of a SPARQL operator O via a set of operators $\mathcal{B}$—expressibility in principle and expressibility by means of a polynomially large pattern. In our setting, $\mathcal{B}$ is the set of the positive operators (either with or without projection) extended by one of the other non-monotone operators. We focus on OPT_F, DIFF_F, their binary counterparts and MINUS; for some results on not exists, we refer the interested reader to (Kaminski, Kostylev, and Cuenca Grau 2016). We adopt the set-based semantics, while the bag (multiset) semantics is left for future work. It can be seen, however, that our inexpressibility results are also applicable to the bag semantics.

Our results can be summarised as follows. (*i*) We show that DIFF_F and OPT_F are polynomially equi-expressible via each other in the presence of projection, but DIFF_F is strictly stronger without it. (*ii*) We challenge the common belief on the expressivity of binary $\text{DIFF}_\top$ and $\text{OPT}_\top$ and prove that although the full versions are expressible via the simplified binary ones, the resulting pattern may be *exponential* in the size of the original. Moreover, such a blowup is unavoidable under the common complexity-theoretic assumptions (if $\Delta_2^p \neq \Sigma_2^p$). If, however, NP $=$ CONP then they are polynomially expressible with projection. (*iii*) Finally, we prove that MINUS is generally weaker than $\text{DIFF}_\top$ and $\text{OPT}_\top$.

Preliminaries

RDF Graphs Let T be a set of *RDF terms*, which consists of IRIs and literals. An (*RDF*) *graph* is a (possibly empty) finite set of triples $(s, p, o) \in \mathsf{T} \times \mathsf{T} \times \mathsf{T}$, where s is called the *subject*, p the *predicate* and o the *object* of the triple. Although graphs in the W3C specification cannot have literals as subjects and predicates but can contain *blank nodes*, we adopt a simplified setting to avoid the notational clutter. These assumptions do not affect any of results in the paper.

SPARQL Syntax We concentrate on the core of SPARQL and build upon the formalisation by Pérez, Arenas, and Gutierrez (2009). Let V be a set of *variable names*. A *triple pattern* is an element of $(\mathsf{T}\cup\mathsf{V}) \times (\mathsf{T}\cup\mathsf{V}) \times (\mathsf{T}\cup\mathsf{V})$. A *basic graph pattern* (*BGP*) is a (possibly empty) finite set of triple patterns; the empty BGP is denoted by $\{\}$. A *graph pattern*, P, is an expression defined by the following grammar:

$$\begin{aligned} P \ ::= \ & B \ \mid \ \text{FILTER}_F\, P \ \mid \ P\ \text{UNION}\ P \ \mid \\ & P\ \text{JOIN}\ P \ \mid \ P\ \text{DIFF}_F\ P \ \mid \ P\ \text{OPT}_F\ P \ \mid \\ & P\ \text{MINUS}\ P \ \mid \ \text{PROJ}_V\, P, \end{aligned}$$

where B is a basic graph pattern, V is a set of variables and F, a *filter*, is a formula constructed from atoms of the form $\mathsf{bnd}(?v)$, $(?v = c)$, $(?v\ = ?u)$, for $?v, ?u \in \mathsf{V}$ and $c \in \mathsf{T}$, using logical connectives $\wedge$ and $\neg$. We employ standard abbreviations: $?v \neq x = \neg(?v = x)$, where $x \in \{c, ?u\}$, $\top = \mathsf{bnd}(?v) \vee \neg\mathsf{bnd}(?v)$, for some $?v \in \mathsf{V}$, $F_1 \vee F_2 = \neg(\neg F_1 \wedge \neg F_2)$, $F_1 \to F_2 = \neg F_1 \vee F_2$ and $F_1 \leftrightarrow F_2 = (F_1 \to F_2) \wedge (F_2 \to F_1)$. The set of variables in P is denoted by $\mathsf{var}(P)$, and the size of P, that is, the number of symbols in its writing, by $|P|$.

Solution Mappings and Filter Evaluation The values are assigned to variables by means of (*solution*) *mappings*, which are partial functions $\mu\colon \mathsf{V} \rightharpoonup \mathsf{T}$ with (possibly empty) domain $\mathit{dom}(\mu)$. Solution mappings μ_1 and μ_2 are said to be *compatible* ($\mu_1 \sim \mu_2$, in symbols) if $\mu_1(?v) = \mu_2(?v)$, for any $?v \in \mathit{dom}(\mu_1) \cap \mathit{dom}(\mu_2)$, in which case $\mu_1 \oplus \mu_2$ denotes the following mapping with domain $\mathit{dom}(\mu_1) \cup \mathit{dom}(\mu_2)$:

$$(\mu_1 \oplus \mu_2)(?v) = \begin{cases} \mu_1(?v), & \text{if } ?v \in \mathit{dom}(\mu_1), \\ \mu_2(?v), & \text{if } ?v \in \mathit{dom}(\mu_2). \end{cases}$$

The *truth-value* $F^\mu \in \{\mathsf{true}, \mathsf{false}\}$ of a filter F *on* a mapping μ is defined inductively as follows:

- $(\mathsf{bnd}(?v))^\mu$ is true iff $?v \in \mathit{dom}(\mu)$;
- $(?v = c)^\mu$ is true iff $?v \in \mathit{dom}(\mu)$ and $\mu(?v) = c$;
- $(?v = ?u)^\mu$ is true iff $?v, ?u \in \mathit{dom}(\mu)$ and $\mu(?v) = \mu(?u)$;
- $(\neg F)^\mu = \neg F^\mu$ and $(F_1 \wedge F_2)^\mu = F_1^\mu \wedge F_2^\mu$.

According to the SPARQL specification, the filters can evaluate not only to the Boolean values of true and false, but also to a special value error. Any filter, however, can always be replaced by a linearly large one so that the error truth value never materialises, but the semantics of the pattern does not change; see (Zhang and Van den Bussche 2014a) for details. So, we adopt the simplified semantics for brevity.

SPARQL Semantics Following (Pérez, Arenas, and Gutierrez 2009), we adopt the set-based semantics, that is, strictly speaking, study SELECT DISTINCT queries.

Given an RDF graph G, the *answer to a graph pattern P on G* is a set $[\![P]\!]_G$ of solution mappings defined by induction on the structure of P. For the basis of induction, if P is a BGP, let

$$[\![P]\!]_G = \{\mu\colon \mathsf{var}(P) \to \mathsf{T} \mid \mu(P) \subseteq G\},$$

where $\mu(P)$ is the set of triples obtained by replacing each variable $?v$ in P by $\mu(?v)$. In particular, for the empty

$\mathcal{S} = \{\text{FILTER}, \text{UNION}, \text{JOIN}\}$

$O' \backslash O$	DIFF_F	OPT_F	$\text{DIFF}_\top$	$\text{OPT}_\top$	MINUS
DIFF_F		$+$ [def.]	$+$ [def.]	$+$ [def.]	$+$ [Th. 18]
OPT_F	$-$ [Th. 2]		$-$ [Th. 2]	$+$ [def.]	$+$ [Th. 18]
$\text{DIFF}_\top$	$\pm$ [Th. 6, 9]	$\pm$ [Th. 6, 9]		$+$ [def.]	$+_?$ [Th. 6, 18]
$\text{OPT}_\top$	$-$ [Th. 2]	$\pm$ [Th. 6, 9]	$-$ [Th. 2]		$+_?$ [Th. 6, 18]
MINUS	$-$ [Th. 19]	$-$ [Th. 19]	$-$ [Th. 19]	$-$ [Th. 19]	

$\mathcal{S}_\pi = \mathcal{S} \cup \{\text{PROJ}\}$

$O' \backslash O$	DIFF_F	OPT_F	$\text{DIFF}_\top$	$\text{OPT}_\top$	MINUS
DIFF_F		$+$ [def.]	$+$ [def.]	$+$ [def.]	$+$ [Th. 18]
OPT_F	$+$ [Th. 4]		$+$ [Corr. 5]	$+$ [def.]	$+$ [Th. 18]
$\text{DIFF}_\top$	$\pm^\dagger$ [Th. 6, 9]	$\pm^\dagger$ [Th. 6, 9]		$+$ [def.]	$+_?^\dagger$ [Th. 6, 18]
$\text{OPT}_\top$	$\pm^\dagger$ [Th. 6, 9]	$\pm^\dagger$ [Th. 6, 9]	$+$ [Corr. 5]		$+_?^\dagger$ [Th. 6, 18]
MINUS	$-$ [Th. 19]	$-$ [Th. 19]	$-$ [Th. 19]	$-$ [Th. 19]	

Table 1: Summary of results on $\mathcal{S} \cup \{O'\}$- and $\mathcal{S}_\pi \cup \{O'\}$-expressibility of O: '$-$' stands for 'not expressible', '$+$' for 'polynomially expressible', '$\pm$' for 'expressible, but not polynomially if $\Delta_2^p \neq \Sigma_2^p$', and '$+_?$' for 'expressible, but not known if polynomially' (by Theorem 17, the results with † become $+$ if NP $=$ CONP).

BGP $\{\}$, we have $[\![\{\}]\!]_\emptyset = \{\mu_\emptyset\}$ even for the empty graph $\emptyset$, where $\mu_\emptyset$ is the mapping with the empty domain (we will use $\mu_\emptyset$ throughout the paper). For the inductive step, we define the following, for a graph G, graph patterns P, P_1 and P_2, filter F, and sets of variables $V \subseteq \mathsf{V}$:

$$[\![\text{FILTER}_F\ P]\!]_G = \{\mu \mid \mu \in [\![P]\!]_G \text{ and } F^\mu = \mathsf{true}\},$$

$$[\![P_1 \text{ UNION } P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G,$$

$$[\![P_1 \text{ JOIN } P_2]\!]_G = \{\mu_1 \oplus \mu_2 \mid \mu_1 \in [\![P_1]\!]_G, \mu_2 \in [\![P_2]\!]_G, \mu_1 \sim \mu_2\},$$

$$[\![P_1 \text{ DIFF}_F\ P_2]\!]_G = \{\mu_1 \in [\![P_1]\!]_G \mid \text{ for all } \mu_2 \in [\![P_2]\!]_G, \text{ either } \mu_1 \not\sim \mu_2 \text{ or } F^{\mu_1 \oplus \mu_2} \neq \mathsf{true}\},$$

$$[\![P_1 \text{ OPT}_F\ P_2]\!]_G = [\![\text{FILTER}_F(P_1 \text{ JOIN } P_2)]\!]_G \cup [\![P_1 \text{ DIFF}_F\ P_2]\!]_G,$$

$$[\![P_1 \text{ MINUS } P_2]\!]_G = \{\mu_1 \in [\![P_1]\!]_G \mid \text{ for all } \mu_2 \in [\![P_2]\!]_G, \text{ either } \mu_1 \not\sim \mu_2 \text{ or } \mathit{dom}(\mu_1) \cap \mathit{dom}(\mu_2) = \emptyset\},$$

$$[\![\text{PROJ}_V\ P]\!]_G = \{\mu|_V \mid \mu \in [\![P]\!]_G\},$$

where $\mu|_V$ is the restriction of μ to V. Note that each BGP is equivalent to the join of its constituent triple patterns (seen as singleton BGPs). The semantics of $\text{DIFF}_\top$ and $\text{OPT}_\top$ is obtained from the general cases: e.g., $[\![P_1 \text{ DIFF}_\top\ P_2]\!]_G$ consists of all the mappings in $[\![P_1]\!]_G$ that are not compatible with *any* mapping in $[\![P_2]\!]_G$. Note the subtle difference of the semantics of $\text{DIFF}_\top$ from MINUS—the latter gives all $\mu_1 \in [\![P_1]\!]_G$ that are not compatible with any $\mu_2 \in [\![P_2]\!]_G$ whose domain *overlaps* the domain of μ_1.[1]

Since our formalisation does not allow for value invention (e.g., by means of BIND operator), all the values in an answer for our graph patterns come from the queried graph. Therefore, for any P, we have

$$\text{either} \quad [\![P]\!]_\emptyset = \emptyset \quad \text{or} \quad [\![P]\!]_\emptyset = \{\mu_\emptyset\}. \tag{2}$$

Observe also that, for a graph G, a pattern $P_\emptyset$ such that $[\![P_\emptyset]\!]_G = \emptyset$ and any filter F, we have

$$[\![\{\} \text{ OPT}_F\ \{\}]\!]_G = \{\mu_\emptyset\}, \qquad [\![\{\} \text{ OPT}_F\ P_\emptyset]\!]_G = \{\mu_\emptyset\},$$
$$[\![\{\} \text{ DIFF}_\top\ \{\}]\!]_G = \emptyset, \qquad [\![\{\} \text{ DIFF}_\top\ P_\emptyset]\!]_G = \{\mu_\emptyset\},$$
$$[\![\{\} \text{ MINUS } \{\}]\!]_G = \{\mu_\emptyset\}, \qquad [\![\{\} \text{ MINUS } P_\emptyset]\!]_G = \{\mu_\emptyset\}.$$

Note that, unlike for OPT_F, the equalities for $\text{DIFF}_\top$ hold for $\top$ but not necessarily for other filters.

[1] In many research papers $\text{DIFF}_\top$ is called MINUS. As far as we are aware, the MINUS of SPARQL 1.1 has not been studied yet.

The SPARQL specification allows for projection (by means of the SELECT clause) only at the outermost level of patterns, rather than at an arbitrary level, as we define. However, using an appropriate variable renaming, PROJ can be pushed outside any other operator and any two projections can be merged. So, our definition is equivalent to the W3C specification, but it is more convenient for our exposition.

Expressibility Problems

In this section we formalise the problem of expressibility of one SPARQL operator via others and outline the results of this paper. We say that patterns P_1 and P_2 are *equivalent* and write $P_1 \equiv P_2$ if $[\![P_1]\!]_G = [\![P_2]\!]_G$, for any graph G.

Definition 1 *Let $\mathcal{B}$ be a set of SPARQL operators. We say that a SPARQL operator O is*

- $\mathcal{B}$-expressible *if, for any pattern over $\mathcal{B} \cup \{O\}$, there is an equivalent pattern over $\mathcal{B}$;*
- polynomially $\mathcal{B}$-expressible *if there exists a polynomial p such that, for any $P = O(P_1, \dots, P_n)$ with the P_i over $\mathcal{B}$, there is an equivalent P' over $\mathcal{B}$ with $|P'| = p(|P|)$.*

Clearly, if O is polynomially $\mathcal{B}$-expressible, then it is $\mathcal{B}$-expressible. Note that expressibility and polynomial expressibility are *composable*: if O is (polynomially) $\mathcal{B} \cup \{O'\}$-expressible and O' is (polynomially) $\mathcal{B}$-expressible, then O is also (polynomially) $\mathcal{B}$-expressible. On the other hand, it is generally possible that O is $\mathcal{B}$-expressible, but not $\mathcal{B}'$-expressible for some $\mathcal{B}' \supset \mathcal{B}$ (and the same for polynomial expressibility). In this paper, however, we will not see such situations, because all the presented proofs for smaller $\mathcal{B}$ go through for any larger $\mathcal{B}'$ considered here. Hence, we will not mention this any more explicitly.

Zhang and Van den Bussche (2014a) studied expressibility of SPARQL operators and showed that none of FILTER, UNION, JOIN, $\text{OPT}_\top$ and PROJ is expressible via the others except for JOIN, which is polynomially $\{\text{FILTER}, \text{OPT}_\top\}$-expressible.

In this paper we focus on expressibility and polynomial expressibility of non-monotone operators $\mathcal{N} = \{\text{DIFF}_F, \text{OPT}_F, \text{DIFF}_\top, \text{OPT}_\top, \text{MINUS}\}$ via each other in the presence of other SPARQL operators. In particular, we consider two sets of monotone basic algebra operators $\mathcal{S} = \{\text{FILTER}, \text{UNION}, \text{JOIN}\}$ and $\mathcal{S}_\pi = \mathcal{S} \cup \{\text{PROJ}\}$ and study the problem of $\mathcal{S} \cup \{O'\}$- and $\mathcal{S}_\pi \cup \{O'\}$-expressibility of an operator $O \in \mathcal{N}$ for another $O' \in \mathcal{N}$. Our results are summarised in Table 1. In the rest of the paper we prove these results and discuss them in detail.

OPT via DIFF and Back

We begin with the expressibility of OPT via DIFF and back, both for general ternary and restricted binary versions. By definition, $\textsc{Opt}_F$ is polynomially $\mathcal{S} \cup \{\textsc{Diff}_F\}$-expressible:

$$P_1 \ \textsc{Opt}_F \ P_2 \ \equiv \ \textsc{Filter}_F(P_1 \ \textsc{Join} \ P_2) \ \textsc{Union} \ (P_1 \ \textsc{Diff}_F \ P_2), \quad (3)$$

for any graph patterns P_1, P_2. As the filter, F, is the same on both sides, $\textsc{Opt}_\top$ is polynomially $\mathcal{S} \cup \{\textsc{Diff}_\top\}$-expressible.

The question of expressibility of DIFF via OPT is, however, less trivial. It was shown (Angles and Gutierrez 2008) that $\textsc{Diff}_F$ is polynomially $\mathcal{S} \cup \{\textsc{Opt}_F\}$-expressible. Alas, the justifying equivalence

$$P_1 \ \textsc{Diff}_F \ P_2 \ \equiv \ \textsc{Filter}_{\neg \mathsf{bnd}(?u)} (P_1 \ \textsc{Opt}_F \ (P_2 \ \textsc{Join} \ \{(?u, ?v, ?w)\})), \quad (4)$$

where $?u, ?v, ?w \notin \mathsf{var}(P_1) \cup \mathsf{var}(P_2)$, holds only under a simplifying assumption that BGPs cannot be empty (or, alternatively, that the graph cannot be empty): it has been observed that if $P_1 = P_2 = \{\}$ and $G = \emptyset$, then the answer to the left-hand side is $\emptyset$, but to the right-hand side is $\{\mu_\emptyset\}$.

Our first result is that this problem is insurmountable, at least without projection. In the proof of the following theorem, as well as in the rest of the paper, we use the *universal triple patterns* of the form $\{(?u, ?v, ?w)\}$ (possibly with sub- or super-scripts) assuming that their variables $?u$, $?v$, $?w$ do not occur elsewhere unless explicitly mentioned.

Theorem 2 $\textsc{Diff}_\top$ *is not* $\mathcal{S} \cup \{\textsc{Opt}_F\}$*-expressible.*

Proof We first claim that patterns P over $\mathcal{S} \cup \{\textsc{Opt}_F\}$ enjoy the following property: for any non-empty graph G that does not contain terms (i.e., IRIs and literals) occurring in P,

$$\text{if } \mu_\emptyset \in [\![P]\!]_G \quad \text{then} \quad \mu_\emptyset \in [\![P]\!]_\emptyset. \quad (5)$$

We prove this claim by induction on the structure of the pattern P (for a fixed G). The basis of induction, for a BGP B in P, follows from the observation that, since G contains no terms of P (and, so, of B), we can only have $\mu_\emptyset \in [\![B]\!]_G$ if $B = \{\}$, whence, $\mu_\emptyset \in [\![\{\}]\!]_\emptyset$. For the inductive step, consider all the operators.

- If $\mu_\emptyset \in [\![\textsc{Filter}_F \ P_1]\!]_G$ then $\mu_\emptyset \in [\![P_1]\!]_G$ and $F^{\mu_\emptyset} = \mathsf{true}$, whence, by the induction hypothesis, $\mu_\emptyset \in [\![P_1]\!]_\emptyset$ and so, $\mu_\emptyset \in [\![\textsc{Filter}_F \ P_1]\!]_\emptyset$.
- If $\mu_\emptyset \in [\![P_1 \ \textsc{Union} \ P_2]\!]_G$, then $\mu_\emptyset \in [\![P_i]\!]_G$ for either $i = 1$ or $i = 2$, whence, by the induction hypothesis, $\mu_\emptyset \in [\![P_i]\!]_\emptyset$, and so, $\mu_\emptyset \in [\![P_1 \ \textsc{Union} \ P_2]\!]_\emptyset$.
- If $\mu_\emptyset \in [\![P_1 \textsc{Join} P_2]\!]_G$, then $\mu_\emptyset \in [\![P_1]\!]_G, [\![P_2]\!]_G$, whence, by the induction hypothesis, $\mu_\emptyset \in [\![P_1]\!]_\emptyset$ and $\mu_\emptyset \in [\![P_2]\!]_\emptyset$, and thus, $\mu_\emptyset \in [\![P_1 \ \textsc{Join} \ P_2]\!]_\emptyset$.
- If $\mu_\emptyset \in [\![P_1 \textsc{Opt}_F P_2]\!]_G$, then $\mu_\emptyset \in [\![P_1]\!]_G$, whence, by the induction hypothesis, $\mu_\emptyset \in [\![P_1]\!]_\emptyset$. By (2), there are two possible cases. If $\mu_\emptyset \in [\![P_2]\!]_\emptyset$ and $F^{\mu_\emptyset \oplus \mu_\emptyset} = \mathsf{true}$ then $\mu_\emptyset = \mu_\emptyset \oplus \mu_\emptyset \in [\![P_1 \ \textsc{Opt}_F \ P_2]\!]_\emptyset$, as required. Otherwise, $\mu_\emptyset \in [\![P_1 \ \textsc{Opt}_F \ P_2]\!]_\emptyset$ by construction.

This completes the proof of the claim.

To show that $\textsc{Diff}_\top$ is not $\mathcal{S} \cup \{\textsc{Opt}_F\}$-expressible, let

$$P = \{\} \ \textsc{Diff}_\top \ \textsc{Filter}_{\neg \mathsf{bnd}(?u)}(\{\} \ \textsc{Opt}_\top \ \{(?u, ?v, ?w)\}).$$

Observe that $[\![P]\!]_\emptyset = \emptyset$ and $[\![P]\!]_G = \{\mu_\emptyset\}$, for any non-empty G. If P were equivalent to a pattern P' over $\mathcal{S} \cup \{\textsc{Opt}_F\}$ then, since $\mu_\emptyset \in [\![P']\!]_G$ for any $G \neq \emptyset$ (in particular, for a non-empty graph not containing any term occurring in P'), we would have, by (5), $\mu_\emptyset \in [\![P']\!]_\emptyset$, contrary to P' being equivalent to P. ❑

In the light of this negative result, the immediate question is whether projection can help in expressibility of DIFF via OPT. We answer affirmatively for the full ternary versions of the operators. To show this, we make the following observation. For a pattern P, let

$$\mathsf{ON\_EMPTY}_P = \textsc{Filter}_{\neg \mathsf{bnd}(?u)}(P_0 \ \textsc{Opt}_\top \ \{(?u, ?v, ?w)\}),$$

where P_0 is $\{\}$ if $[\![P]\!]_\emptyset = \{\mu_\emptyset\}$ and $\{(?u', ?v', ?w')\}$ otherwise, that is, if $[\![P]\!]_\emptyset = \emptyset$. It is immediate to verify that this pattern simulates P on the empty graph and gives the empty answer on all other graphs.

Proposition 3 *For any graph pattern P and graph G,*

$$[\![\mathsf{ON\_EMPTY}_P]\!]_G = \begin{cases} [\![P]\!]_\emptyset, & \textit{if } G = \emptyset, \\ \emptyset, & \textit{otherwise}. \end{cases}$$

Note that, given a pattern P, $\mathsf{ON\_EMPTY}_P$ can be computed in deterministic polynomial time in $|P|$. With Proposition 3 at hand, we are ready to prove our first positive result.

Theorem 4 $\textsc{Diff}_F$ *is polynomially* $\{\textsc{Filter}, \textsc{Union}, \textsc{Proj}, \textsc{Opt}_F\}$*-expressible.*

Proof Recall that JOIN is polynomially $\{\textsc{Filter}, \textsc{Opt}_\top\}$-expressible (Zhang and Van den Bussche 2014a), so we can use it in the the proof (although the result was shown for the language without the empty BGP, it holds in our setting as well). We now show the following equivalence:

$$P_1 \ \textsc{Diff}_F \ P_2 \ \equiv \ \mathsf{ON\_EMPTY}_{P_1 \textsc{Diff}_F P_2} \ \textsc{Union} \ \textsc{Proj}_{\mathsf{var}(P_1)} \ \textsc{Filter}_{\neg \mathsf{bnd}(?u_2)} ((P_1 \ \textsc{Join} \ \mathsf{utp}_1) \ \textsc{Opt}_F \ (P_2 \ \textsc{Join} \ \mathsf{utp}_2)), \quad (6)$$

where $\mathsf{utp}_i = \{(?u_i, ?v_i, ?w_i)\}$ are universal patterns with fresh variables. Indeed, if $G = \emptyset$ then $[\![P_i \ \textsc{Join} \ \mathsf{utp}_i]\!]_G = \emptyset$, for $i = 1, 2$, and so, the second component of the union is empty. Then, by Proposition 3, the answers of both sides of the equivalence coincide. If $G \neq \emptyset$, then, by Proposition 3, the first component of the union is empty and the equivalence follows from the observation that $?u_2$ is not bound precisely in those solution mappings that come from the $\textsc{Diff}_F$ component of $\textsc{Opt}_F$. Equivalence (6) implies polynomial (in fact, linear) $\{\textsc{Filter}, \textsc{Union}, \textsc{Proj}, \textsc{Opt}_F\}$-expressibility of $\textsc{Diff}_F$. ❑

Again, since filter F is the same on both sides of (6), the result holds for the binary versions of the operators as well.

Corollary 5 $\textsc{Diff}_\top$ *is polynomially* $\{\textsc{Filter}, \textsc{Union}, \textsc{Proj}, \textsc{Opt}_\top\}$*-expressible.*

Ternary OPT and DIFF via Binary Versions

In this section we study expressibility of the general ternary OPT_F via the simplified binary $\text{OPT}_\top$ as well as the ternary DIFF_F via the binary $\text{DIFF}_\top$. In particular, we show that $\mathcal{S}$-expressibility holds in both cases; however, polynomial $\mathcal{S}_\pi$-expressibility does not hold at least under the common complexity-theoretic assumptions.

We begin with a discussion of known suggestions for expressing OPT_F via $\text{OPT}_\top$. Angles and Gutierrez (2008) claimed that OPT_F is polynomially $\{\text{FILTER}, \text{JOIN}, \text{OPT}_\top\}$-expressible. Alas, the justifying equivalence

$$P_1 \;\text{OPT}_F\; P_2 \;\equiv\; P_1 \;\text{OPT}_\top\; \text{FILTER}_F(P_1 \;\text{JOIN}\; P_2) \qquad (7)$$

is incorrect. Indeed, consider $F = \mathsf{bnd}(?v)$,

$$P_1 = \{(?u, a, ?u), (?v, b, ?v)\} \;\text{UNION}\; \{(?u, a, ?u)\}, \text{ and}$$
$$P_2 = \{(?u, a, ?u), (?w, c, ?w)\}.$$

Then, on the graph $G = \{(a, a, a),\ (b, b, b),\ (c, c, c)\}$, we obtain $[\![P_1]\!]_G = \{\mu_1, \mu_2\}$ and $[\![P_2]\!]_G = \{\mu_3\}$, where

$$\mu_1 = \{?u \mapsto a, ?v \mapsto b\}, \qquad \mu_2 = \{?u \mapsto a\},$$
$$\mu_3 = \{?u \mapsto a, ?w \mapsto c\}.$$

Hence, the answer to the left-hand side of (7) consists of $\mu_1 \oplus \mu_3$ and μ_2 (because μ_2 has no compatible solution mapping in $[\![P_2]\!]_G$ that would satisfy F), while the answer to the right-hand side contains only $\mu_1 \oplus \mu_3$ (because the answer to the right argument of $\text{OPT}_\top$ has only $\mu_1 \oplus \mu_3$).

Theorem 6 *The following holds*:

- OPT_F *is* $\{\text{FILTER}, \text{OPT}_\top\}$*-expressible,*
- DIFF_F *is* $\{\text{FILTER}, \text{JOIN}, \text{DIFF}_\top\}$*-expressible,*
- OPT_F *is* $\mathcal{S} \cup \{\text{DIFF}_\top\}$*-expressible, and*
- DIFF_F *is* $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$*-expressible.*

Proof We claim the following modification of (7) holds (recall that JOIN is $\{\text{FILTER}, \text{OPT}_\top\}$-expressible):

$$P_1 \;\text{OPT}_F\; P_2 \;\equiv\; \text{UNION}_{V \subseteq U}((\text{FILTER}_{F_V}\, P_1)\text{OPT}_\top \;\text{FILTER}_F((\text{FILTER}_{F_V}\, P_1) \;\text{JOIN}\; P_2)), \qquad (8)$$

where $U = \mathsf{var}(P_1) \cap \mathsf{var}(P_2)$, $\text{UNION}_{V \subseteq U}\, P_V$ is the $2^{|U|}$-ary UNION of patterns P_V, for $V \subseteq U$, and

$$F_V \;=\; \bigwedge\nolimits_{?v \in V} \mathsf{bnd}(?v) \;\wedge\; \bigwedge\nolimits_{?v \in U \setminus V} \neg\mathsf{bnd}(?v).$$

Although the idea of this equivalence is similar to (7), we, however, construct the $\text{OPT}_\top$ for patterns whose answers are defined on the same subset V of the variables shared by P_1 and P_2, which prevents counterexamples as above. A similar equivalence holds for DIFF (for the same U and F_V):

$$P_1 \;\text{DIFF}_F\; P_2 \;\equiv\; \text{UNION}_{V \subseteq U}((\text{FILTER}_{F_V}\, P_1)\text{DIFF}_\top \;\text{FILTER}_F((\text{FILTER}_{F_V}\, P_1) \;\text{JOIN}\; P_2)). \qquad (9)$$

Finally, $\mathcal{S} \cup \{\text{DIFF}_\top\}$-expressibility of OPT_F follows from (3) and (9), while $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$-expressibility of DIFF_F follows from (8) by Theorem 4. ❑

The drawback of the expressions in the proof of Theorem 6 is that they iterate over all subsets V of U, that is, they are not polynomial. Next, we address the problem of polynomial expressibility, and show that it is not possible even in the presence of projection for both DIFF and OPT. We establish the result using a complexity-theoretic argument, so our results are relative to commonly believed assumptions.

One way of proving such a result would be to show that the *evaluation problems*, that is, the problems of checking whether a mapping is in the answer to a given pattern on a given graph, are complete for different complexity classes for patterns over $\mathcal{S} \cup \{\text{OPT}_\top\}$ and patterns of the form $P_1\; \text{OPT}_F\; P_2$ with P_1, P_2 over $\mathcal{S} \cup \{\text{OPT}_\top\}$. Then, a polynomial $\mathcal{S} \cup \{\text{OPT}_\top\}$-expressibility of OPT_F would imply that the complexity classes coincide. Alas, evaluation for the former is PSPACE-complete (Pérez, Arenas, and Gutierrez 2009), and it is not difficult to show that the latter is also PSPACE-complete; same applies to $\mathcal{S}_\pi$ instead of $\mathcal{S}$. This is why we have to consider a more involved problem instead of evaluation.

In particular, we look at the problem of emptiness of the answer to a pattern on *singular graphs* G_a, that is, graphs of the form $\{(a, a, a)\}$, and show that its complexity is different for patterns over $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$ and patterns of the form $P_1\; \text{OPT}_F\; P_2$ with P_1, P_2 over $\mathcal{S} \cup \{\text{OPT}_\top\}$.

Lemma 7 *The problem of whether* $[\![P]\!]_{G_a} \neq \emptyset$*, for patterns* P *over* $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$ *and singular graphs* G_a*, is in* Δ_2^p*.*

Proof We first observe that the emptiness problem for a pattern over $\mathcal{S}_\pi$ on G_a is in NP. Indeed, we can first guess a solution mapping and an argument of each UNION, and then check that the solution mapping is indeed in the answer to the resulting UNION-free pattern.

Next, we show that, for any pattern P over $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$, the problem of whether $\mu \in [\![P]\!]_{G_a}$ is decidable by a polynomial deterministic algorithm with $|P| + 1$ calls to an NP-oracle. The key observation is that, for any $P_1\; \text{OPT}_\top\; P_2$,

$$[\![P_1 \;\text{OPT}_\top\; P_2]\!]_{G_a} = \begin{cases} [\![P_1 \;\text{JOIN}\; P_2]\!]_{G_a}, & \text{if } [\![P_2]\!]_{G_a} \neq \emptyset, \\ [\![P_1]\!]_{G_a}, & \text{if } [\![P_2]\!]_{G_a} = \emptyset. \end{cases}$$

The algorithm then proceeds by repeatedly taking, while possible, the innermost occurrence $P_1\; \text{OPT}_\top\; P_2$ of $\text{OPT}_\top$ in P, calling the NP-oracle to decide whether P_2, which is a pattern over $\mathcal{S}_\pi$, returns at least one solution mapping on G_a, and depending on the outcome, replacing $P_1\; \text{OPT}_\top\; P_2$ in P with either $P_1\; \text{JOIN}\; P_2$ or P_1. After at most $|P|$ interations, we obtain a pattern P' over $\mathcal{S}_\pi$ that has the same answer on G_a as P. It remains to call the NP-oracle one more time to check whether there is a μ with $\mu \in [\![P']\!]_{G_a} = [\![P]\!]_{G_a}$. ❑

For the separation result, we define the *o-rank* of a pattern P over $\mathcal{S} \cup \{\text{OPT}_F\}$ as the depth of nesting of OPT_F in P (Schmidt, Meier, and Lausen 2010): e.g., the o-rank of $(P_1\; \text{OPT}_F\; P_2)\; \text{OPT}_{F'}\; P_3$ for OPT_F-free patterns P_i is 2.

Lemma 8 *The problem of whether* $[\![P]\!]_{G_a} \neq \emptyset$*, for patterns* P *over* $\mathcal{S} \cup \{\text{OPT}_F\}$ *of rank at most* n*,* $n \geq 0$*, and singular graphs* G_a*, is* Σ_{n+1}^p*-hard.*

Proof The proof is by reduction of the validity problem for quantified Boolean formulas (QBFs) with n quantifier alternations. Let $\phi = \exists\bar{x}_1 \forall\bar{x}_2 \exists\bar{x}_3 \ldots \mathsf{Q}\bar{x}_{n+1}\, \psi$ be a closed QBF, where the $\bar{x}_i$ are tuples of variables and ψ is a quantifier-free propositional formula over all the $\bar{x}_i$. For each tuple $\bar{x}_i = x_{i1}, \ldots, x_{im_i}$ and $k \leq n+1$, consider the pattern

$$X_i^k \;=\; U_{i1}^k \text{ JOIN} \ldots \text{JOIN } U_{im_i}^k,$$

where $U_{ij}^k = \{(?u_{ij}^k, ?u_{ij}^k, ?u_{ij}^k)\}$ UNION $\{\}$, for $j \leq m_i$. It is easy to see that $[\![X_i^k]\!]_{G_a}$ represents all possible assignments to variables $\bar{x}_i$. More precisely, we say that x_{ij} is assigned true in $\mu \in [\![X_i^k]\!]_{G_a}$ if $(\mathsf{bnd}(?u_{ij}^k))^\mu = \mathsf{true}$ and is assigned false otherwise.

Next, for $k \leq n+1$, consider the following pattern:

$$B_k \;=\; \{(?v_k, ?v_k, ?v_k)\} \text{ JOIN } X_1^k \text{ JOIN} \ldots \text{JOIN } X_k^k.$$

Clearly, each $\mu \in [\![B_k]\!]_{G_a}$ defines a possible assignment to $\bar{x}_1, \ldots, \bar{x}_k$ and the other way round; also, μ has $?v_k$ bound.

We define the required pattern using induction on k along a sequence of QBFs. If n is even and so $\mathsf{Q} = \exists$, then we take

$$\phi_{n+1} = \psi \quad \text{and} \quad \phi_k = \forall \bar{x}_{k+1}\, \neg\phi_{k+1}, \text{ for all } k \leq n.$$

We clearly have $\phi = \exists \bar{x}_1\, \phi_1$. Consider first

$$P_{n+1} \;=\; \text{FILTER}_{F_\psi}\, B_{n+1},$$

where F_ψ is obtained from ψ by replacing each occurrence of x_{ij} by $\mathsf{bnd}(?u_{ij}^{n+1})$. Then $[\![P_{n+1}]\!]_{G_a}$ consists of all satisfying assignments for ϕ_{n+1} labelled by $?v_{n+1}$. For the inductive step, given P_{k+1}, for $k \leq n$, we define P_k by taking

$$P_k \;=\; \text{FILTER}_{\neg\mathsf{bnd}(?v_{k+1})}(B_k \text{ OPT}_{F_k}\, P_{k+1}),$$

where F_k is a conjunction of the following formulas:

$$\mathsf{bnd}(?u_{ij}^k) \leftrightarrow \mathsf{bnd}(?u_{ij}^{k+1}), \qquad \text{for all } i \leq k \text{ and } j \leq m_i.$$

Since each $\mu \in [\![P_{k+1}]\!]_{G_a}$ has $?v_{k+1}$ bound by it, the filter effectively leaves only $[\![B_k \text{ DIFF}_{F_k}\, P_{k+1}]\!]_{G_a}$ from $[\![P_k]\!]_{G_a}$. Observe also that all solution mappings in $[\![B_k]\!]_{G_a}$ and $[\![P_{k+1}]\!]_{G_a}$ are compatible (their domains are in fact disjoint). So, $[\![B_k \text{ DIFF}_{F_k}\, P_{k+1}]\!]_{G_a}$ consists of mappings $\mu \in [\![B_k]\!]_{G_a}$ such that there is no $\mu_1 \in [\![P_{k+1}]\!]_{G_a}$ with $F_k^{\mu\oplus\mu_1} = \mathsf{true}$. By the definition of F_k, the assignment of $\bar{x}_1, \ldots, \bar{x}_k$ in μ cannot be extended to an assignment of $\bar{x}_1, \ldots, \bar{x}_k, \bar{x}_{k+1}$ in a solution mapping in $[\![P_{k+1}]\!]_{G_a}$, which, by induction hypothesis, means that $[\![P_k]\!]_{G_a}$ consists of all satisfying assignments for ϕ_k labelled by $?v_k$.

If n is odd and $\mathsf{Q} = \forall$, we build F_ψ from $\neg\psi$ instead of ψ.

Finally, observe that ϕ is true just in case $[\![P_1]\!]_{G_a} \neq \emptyset$ and that P_1 for ϕ with $n+1$ quantifier groups has o-rank n. ❑

These two lemmas give us the following result.

Theorem 9 *Provided that $\Delta_2^p \neq \Sigma_2^p$, operators* OPT_F *and* DIFF_F *are not polynomially $\mathcal{S}_\pi \cup \{O'\}$-expressible for any $O' \in \{\text{OPT}_\top, \text{DIFF}_\top\}$.*

Proof The claim for OPT_F and $O' = \text{OPT}_\top$ is immediate from Lemmas 7 and 8. Indeed, if OPT_F were polynomially $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$-expressible, then we would be able to polynomially rewrite any pattern of the form $P_1 \text{ OPT}_F\, P_2$ with P_1 and P_2 over $\mathcal{S}$ (i.e., with o-rank 1) to an equivalent pattern over $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$ and then solve the emptiness problem on G_a in Δ_2^p, contrary to Σ_2^p-hardness of the problem.

The proofs for other three cases are similar; we give one for DIFF_F and $O' = \text{DIFF}_\top$. If DIFF_F were polynomially $\mathcal{S}_\pi \cup \{\text{DIFF}_\top\}$-expressible, then we would be able to polynomially rewrite any $P_1 \text{ OPT}_F\, P_2$, as above, to a pattern over $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$ in three steps: first, we would apply (3) to replace each OPT_F by DIFF_F, then apply the expressibility hypothesis to go to $\text{DIFF}_\top$, and finally apply (4) to arrive at $\text{OPT}_\top$. Even though the resulting pattern may not be equivalent to $P_1 \text{ OPT}_F\, P_2$ (due to possibly different semantics on the empty graph), it would still have the correct answer on G_a, and we would be able to decide its emptiness in Δ_2^p. ❑

Polynomial Expressibility on Non-Singular Graphs

The results in Theorem 9 are negative—the ternary versions of DIFF and OPT are not polynomially expressible via the binary ones under the commonly believed complexity-theoretic assumptions (i.e., unless $\Sigma_2^p = \Delta_2^p$ and the polynomial hierarchy collapses). The proof, however, relies on a very specific case of singular graphs G_a, which use only one term (e.g., IRI). In this section we show that, in the presence of projection, this is the only difficult case, and, if we have an operator O that gives the same answers as DIFF_F (or OPT_F) on singular graphs, then both OPT_F and DIFF_F are polynomially $\mathcal{S}_\pi \cup \{\text{DIFF}_\top, O\}$-expressible.

By (3) and Theorem 4, DIFF_F and OPT_F are polynomially expressible via each other in the presence of projection; the same applies to $\text{DIFF}_\top$ and $\text{OPT}_\top$. Since in this section we focus on $\mathcal{S}_\pi$, which includes projection, we use DIFF_F and OPT_F (as well as $\text{DIFF}_\top$ and $\text{OPT}_\top$) interchangeably. In fact, it will be more convenient to use yet another operator that is polynomially equi-expressible with $\text{DIFF}_F/\text{OPT}_F$.

Definition 10 *For patterns P_1, P_2, let P_1 SETMINUS P_2 be a pattern with the following semantics for a graph G (where '$\setminus$' is the usual set difference):*

$$[\![P_1 \text{ SETMINUS } P_2]\!]_G = [\![P_1]\!]_G \setminus [\![P_2]\!]_G.$$

To prove that SETMINUS is polynomially $\mathcal{S}_\pi \cup \{\text{DIFF}_F\}$-expressible (and vice versa), we need some extra notation. Given a pattern P, an injective mapping θ is called a *variable renaming for P* if its domain, $dom(\theta)$, coincides with $\mathsf{var}(P)$ and its image is disjoint from $\mathsf{var}(P)$. In other words, θ renames all variables of P into fresh ones. Given such a θ, we denote by $P\theta$ the result of replacing each occurrence of a variable $?v$ in P by its image, $?v\theta$. Similar notation, $F\theta$, will also be used with filters F. A special filter EQ^θ for θ is defined as a conjunction of formulas, for all $?v \in dom(\theta)$,

$$\big(\mathsf{bnd}(?v) \leftrightarrow \mathsf{bnd}(?v\theta)\big) \wedge \big(\mathsf{bnd}(?v) \rightarrow ?v = ?v\theta\big).$$

Intuitively, EQ^θ filters out solution mappings that disagree on $\mathsf{var}(P)$ and $\theta(\mathsf{var}(P))$.

We are now ready to prove equi-expressibility of DIFF_F and SETMINUS, and begin with the forward direction.

Lemma 11 DIFF_F *(and, hence,* OPT_F*) is polynomially $\mathcal{S}_\pi \cup$ {SETMINUS}-expressible.*

Proof For any P_1 and P_2 over $\mathcal{S}_\pi \cup \{\text{SETMINUS}\}$ and variable renaming θ for P_1 DIFF_F P_2, we claim that

$$P_1 \ \text{DIFF}_F \ P_2 \ \equiv \ P_1 \ \text{SETMINUS} \ \big(\text{PROJ}_{\mathsf{var}(P_1)} \ \text{FILTER}_{F\theta}(\text{FILTER}_{\mathsf{EQ}^\theta}(P_1 \ \text{JOIN} \ P_1\theta) \ \text{JOIN} \ P_2\theta)\big).$$

To see this equivalence, observe that, for any graph G, the answer to the pattern on the right consists of all mappings $\mu_1 \in [\![P_1]\!]_G$, whose copies $\mu_1\theta$ are not compatible with any $\mu_2\theta \in [\![P_2\theta]\!]_G$ such that $F\theta^{\mu_1\theta\oplus\mu_2\theta} = \mathsf{true}$; equivalently, not compatible with any $\mu_2 \in [\![P_2]\!]_G$ with $F^{\mu_1\oplus\mu_2} = \mathsf{true}$. The latter is just the definition of $[\![P_1 \ \text{DIFF}_F \ P_2]\!]_G$. ❑

The following lemma establishes the converse direction.

Lemma 12 SETMINUS *is polynomially* $\mathcal{S}_\pi \cup \{\text{DIFF}_F\}$-*expressible.*

Proof For any P_1 and P_2 over $\mathcal{S}_\pi \cup \{\text{DIFF}_F\}$ and variable renaming θ for P_1 SETMINUS P_2, we claim that

$$P_1 \ \text{SETMINUS} \ P_2 \ \equiv \ P_1 \ \text{DIFF}_{\mathsf{EQ}^\theta} \ P_2\theta.$$

To see this, observe that, for any G, all solution mappings in $[\![P_1]\!]_G$ are compatible with all mappings in $[\![P_2\theta]\!]_G$. Thus, $[\![P_1 \ \text{DIFF}_{\mathsf{EQ}^\theta} \ P_2\theta]\!]_G$ consists of mappings $\mu_1 \in [\![P_1]\!]_G$ that have no mapping in $\mu_2 \in [\![P_2\theta]\!]_G$ such that $\mu_1 \oplus \mu_2$ satisfies EQ^θ; in other words, of all mappings in $[\![P_1]\!]_G \setminus [\![P_2]\!]_G$. ❑

Having established the fact that, in the presence of projection, SETMINUS is polynomially equi-expressible with DIFF_F and OPT_F, we are ready to prove the main result of this section (note that instead of MONOMINUS below we may equivalently require existence of MONODIFF_F or MONOOPT_F with similar semantics).

Theorem 13 *Let* MONOMINUS *be an operator that gives the same answers as* SETMINUS *on singular graphs. Then* SETMINUS *is polynomially* $\mathcal{S}_\pi \cup \{\text{MONOMINUS}, \text{OPT}_\top\}$-*expressible.*

Proof We begin by defining auxiliary patterns. First, for any variable $?v$, consider a pattern whose answers are all possible bindings of $?v$ to terms in the graph:

$$\mathsf{ADOM}_{?v} \ = \ \text{PROJ}_{\{?v\}}\big(\{?v, ?v', ?v''\} \ \text{UNION} \ \{?v', ?v, ?v''\} \ \text{UNION} \ \{?v', ?v'', ?v\}\big).$$

Then, for each pair of variables, $?u$ and $?v$, consider a pattern that gives all pairs of distinct terms in the graph:

$$\mathsf{NEQ}_{?u,?v} \ = \ \text{FILTER}_{?u\neq?v}\big(\mathsf{ADOM}_{?u} \ \text{JOIN} \ \mathsf{ADOM}_{?v}\big).$$

We can now separate graphs by the number of terms in them:

$$\mathsf{two} \ = \ \text{PROJ}_\emptyset \ \mathsf{NEQ}_{?u,?v},$$
$$\mathsf{one} \ = \ \text{PROJ}_\emptyset \ \text{FILTER}_{\neg\mathsf{bnd}(?u)} \big(\{?w, ?w', ?w''\} \ \text{OPT}_\top \ \mathsf{NEQ}_{?u,?v}\big).$$

It is easy to verify that $[\![\mathsf{two}]\!]_G \neq \emptyset$ if and only if G has at least two terms and $[\![\mathsf{one}]\!]_G \neq \emptyset$ if and only if G is singular.

Consider now a pattern P_1 SETMINUS P_2 with variables $V = \{?v_1, \dots, ?v_n\}$. Let θ and θ' be variable renamings for P_1 SETMINUS P_2 with disjoint ranges. For $i = 1, 2$, let

$$P_i^* \ = \ (\dots(\text{FILTER}_{\mathsf{EQ}^{\theta^{-1}\theta'}}(P_i\theta \ \text{JOIN} \ P_i\theta')) \ \text{OPT}_\top \ \mathsf{NEQ}_{?v_1\theta,?v_1\theta'} \dots) \ \text{OPT}_\top \ \mathsf{NEQ}_{?v_n\theta,?v_n\theta'};$$

note that $\theta^{-1}\theta'$ is a variable renaming with the range of θ as its domain such that it maps new variables of θ to corresponding new variables of θ'. It is readily seen that, for any graph G with at least two terms, $[\![P_i^*]\!]_G$ consists of solution mappings with domain $V\theta \cup V\theta'$ and of the form

$$\mu\theta \oplus \mu\theta' \oplus \delta, \tag{10}$$

where $\mu \in [\![P_i]\!]_G$ and δ is such that $\delta(?v\theta) \neq \delta(?v\theta')$, for all $?v \in V \setminus \mathit{dom}(\mu)$. We say that μ is the *origin* of (10) in $[\![P_i]\!]_G$; note that the origin is uniquely defined. In other words, we transform each solution mapping μ in $[\![P_i]\!]_G$ in the following way: if $?v_j$ is bound in μ then both $?v_j\theta$ and $?v_j\theta'$ are bound by the same value; otherwise, both $?v_j\theta$ and $?v_j\theta'$ are bound by all possible combinations of *different* values.

Let $P^* = P_1^* \ \text{DIFF}_\top \ P_2^*$. Observe that, on graphs with at least two terms, P^* gives the transformed versions of the mappings in the answer to P_1 SETMINUS P_2. Indeed, the domains of all solution mappings in $[\![P_1^*]\!]_G$ and $[\![P_2^*]\!]_G$ coincide with $V\theta \cup V\theta'$. So, each $\mu_1 \in [\![P_1^*]\!]_G$ is compatible with at most one $\mu_2 \in [\![P_2^*]\!]_G$; moreover, μ_1 can only be compatible with μ_2 if the domain of the origin of μ_1 in $[\![P_1]\!]_G$ coincides with the domain of the origin of μ_2 in $[\![P_2]\!]_G$.

We are now in a position to express SETMINUS. We consider three cases for G, which either (a) is empty, or (b) contains exactly one term, or (c) contains at least two terms:

$$P_1 \ \text{SETMINUS} \ P_2 \ \equiv \ \mathsf{ON\_EMPTY}_{P_1 \text{SETMINUS} P_2} \ \text{UNION} \ ((P_1 \ \text{MONOMINUS} \ P_2) \ \text{JOIN} \ \mathsf{one}) \ \text{UNION} \ (\text{PROJ}_{\mathsf{var}(P_1)}\big(\text{FILTER}_F(P_1 \ \text{JOIN} \ P^*)\big) \ \text{JOIN} \ \mathsf{two}),$$

where F is the following conjunction, for all $?v \in \mathit{dom}(\theta)$:

$$\big(\mathsf{bnd}(?v) \leftrightarrow (?v\theta = ?v\theta')\big) \wedge \big(\mathsf{bnd}(?v) \rightarrow (?v = ?v\theta)\big).$$

Note that $?v\theta$ and $?v\theta'$ are bound in all solution mappings in $[\![P_1 \ \text{JOIN} \ P^*]\!]_G$ for G with at least two terms. It should be now clear that the patterns are as required. ❑

We leave open the question of whether DIFF_F and OPT_F are polynomially expressible via their binary counterparts and patterns that give the same answers on singular graphs without projection.

Polynomial Expressibility of MonoMinus

In the previous section we have shown that the question of polynomial $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$-expressibility of OPT_F boils down to the existence of an operator MONOMINUS that gives the same answers as SETMINUS on singular graphs. By Theorem 9, there is no such polynomially $\mathcal{S}_\pi \cup \{\text{OPT}_\top\}$-expressible MONOMINUS under usual complexity-theoretic assumptions. In this section, however, we show that if NP = coNP then MONOMINUS does exist.

Let $\Sigma = \{0, 1\}$. A (*partial*) *multivalued function* is a relation on Σ^*. For a multivalued function f, we write $\mathsf{set}\text{-}f(x)$ for the set $\{y \mid (x, y) \in f\}$. A *transducer* is a nondeterministic Turing machine over Σ with read-only input tape, write-only output tape, read-write work tapes, and accepting states as usual. A transducer computes a value y on input x if there is an accepting computation that starts with x on the input tape and ends with y on the output tape. Hence, in general, transducers compute partial multivalued functions.

Let NPMV be the class of multivalued functions computed by nondeterministic polynomial-time bounded transducers (Book, Long, and Selman 1985) and NPMV$^{\text{NP}}$ the class of multivalued functions computed by polynomial-time bounded transducers that have access to an NP oracle.

A multivalued function f is *strongly metric many-one polynomial reducible (psm-reducible)* to a function g if there exist polynomially computable functions t_1 and t_2 such that $\mathsf{set}\text{-}f(x) = \{t_2(x, y) \mid (t_1(x), y) \in g\}$ for any x (Krentel 1988; Fenner et al. 1999). Intuitively, given a value of g on $t_1(x)$, we can compute in polynomial time a value of f on x and, by varying over all values of g on $t_1(x)$, obtain all values of f on x. In the sequel, when talking about hardness of functional problems, we silently assume psm-reductions.

The class NPMV is closed under psm-reducibility. The multivalued function FSAT, defined so that $\bar{s}$ is a value of $\text{FSAT}(\psi)$ if and only if $\bar{s}$ is a satisfying assignment for propositional Boolean formula $\psi(\bar{x})$, is complete for NPMV (Fenner et al. 1999).

Consider now the following multivalued function, parametrised by a sublanguage $\mathcal{B}$ of SPARQL (e.g., a set of operators). The function $\text{FMONOEVAL}_{\mathcal{B}}$ is defined on (binary representations of) all patterns in $\mathcal{B}$ that do not contain terms: for each such pattern P, the set $\mathsf{set}\text{-}\text{FMONOEVAL}_{\mathcal{B}}(P)$ consists of all binary strings representing $[\![P]\!]_{G_a}$ in the sense that 1 in position i of the string means that the ith variable is bounded by the solution mapping and 0 that it is not (we assume that an order on $\mathsf{var}(P)$ is fixed). This function is well-defined because P does not contain terms, and so, answers on different singular graphs are the same modulo renaming of the term in the graph.

We can establish NPMV-hardness of computing answers to patterns over $\mathcal{S}_\pi$.

Lemma 14 $\text{FMONOEVAL}_{\mathcal{S}_\pi}$ *is* NPMV*-hard.*

Proof We prove the claim by reduction of FSAT. Given a propositional Boolean formula ψ over variables $x_1, \dots, x_n$, consider a pattern

$$t_1(\psi) = \text{FILTER}_{F_\psi}(U_1 \text{ JOIN} \dots \text{JOIN } U_n),$$

where $U_i = \{(?u_i, ?u_i, ?u_i)\} \text{ UNION } \{\}$, for $i \leq n$, and F_ψ is obtained from ψ by replacing each occurrence of x_i by $\mathsf{bnd}(?u_i)$; see Lemma 8. Note that $t_1(\psi)$ is a pattern over $\mathcal{S}_\pi$ without terms. Consider a function t_2 sending each mapping μ over $?u_1, \dots, ?u_n$ to assignments of $x_1, \dots, x_n$ in such a way that x_i is true if and only if $?u_i$ is bound by μ. The polynomial functions t_1 and t_2 define a psm-reduction. ❑

On the other hand, we have the following for the class $\mathcal{S}_\pi^{SM}$ of patterns of the form P_1 SETMINUS P_2, for $P_1, P_2 \in \mathcal{S}_\pi$.

Lemma 15 $\text{FMONOEVAL}_{\mathcal{S}_\pi^{SM}}$ *is in* NPMV$^{\text{NP}}$.

Proof Given P_1 SETMINUS P_2, the algorithm works as follows: it builds all mappings over $\mathsf{var}(P_1) \cup \mathsf{var}(P_2)$ and then, for each of them, checks, by means of two oracle calls, whether the mapping is in the answer to P_1 but not to P_2. ❑

We are ready to prove the key lemma in this section.

Lemma 16 *If* NP = coNP *then, for any pattern over* $\mathcal{S}_\pi^{SM}$*, there is a polynomial-size pattern over* $\mathcal{S}_\pi$ *that gives the same answers on singular graphs.*

Proof Consider any P over $\mathcal{S}_\pi^{SM}$. Let $a_1, \dots, a_m$ be all the terms that occur in P. It should then be clear that, on singular graphs, P gives the same answers as the following pattern:

$$\text{PROJ}_{\mathsf{var}(P)}\big[\text{FILTER}_{\bigwedge_{1\leq k\leq m} ?v \neq a_k}\big(P_0 \text{ JOIN } B\big) \text{ UNION} \\ \text{UNION}_{1 \leq k \leq m} \text{FILTER}_{?v = a_k}\big(P_k \text{ JOIN } B\big)\big], \quad (11)$$

where $B = \{(?v, ?v, ?v)\}$, for a fresh variable $?v$, and every P_k, $0 \leq k \leq m$, is obtained from P by replacing

- each BGP that contains a term *different from* a_i with $\text{FILTER}_\perp\{\}$ (for $k = 0$, all BGPs with terms are replaced);
- each $?u = a_k$ in a filter by $\top$ and each $?u = a_\ell$, for $\ell \neq k$, by $\perp$ (for $k = 0$, only the second option is applicable).

We now show that, if NP = coNP then, for each P_k, $0 \leq k \leq m$, we can construct a polynomial-size pattern P'_k over $\mathcal{S}_\pi$ such that P_k and P'_k have the same answers on the singular graph $\{(a_k, a_k, a_k)\}$ (for $k = 0$, we take $a_k = a$, for some fresh a). We then will replace patterns P_k in (11) by P'_k to obtain the required polynomial-size pattern over $\mathcal{S}_\pi$ that has the same answers as P on arbitrary singular graphs.

Fix some index k in $0, \dots, m$. Observe that P_k is a pattern over $\mathcal{S}_\pi^{SM}$ that contains no terms. If NP = coNP then NPMV = NPMV$^{\text{NP}}$ (Fenner et al. 1999). Hence, there is a psm-reduction of $\text{FMONOEVAL}_{\mathcal{S}_\pi^{SM}}$ to $\text{FMONOEVAL}_{\mathcal{S}_\pi}$, that is, there are polynomial functions t_1 and t_2 specifying the reduction: $t_1(P_k)$ is a pattern over $\mathcal{S}_\pi$ and t_2 maps P_k and each mapping in the answer of $t_1(P_k)$ on $\{(a_k, a_k, a_k)\}$ to a mapping in the answer of P_k on $\{(a_k, a_k, a_k)\}$. Note that each mapping in the answer to P_k and $t_1(P_k)$ is completely characterised by its domain. Now, t_2 is implementable by a deterministic polynomial-time Turing machine (transducer) $\mathfrak{M}$ over alphabet $\Sigma = \{0, 1\}$. It is routine (see e.g., the proof of the Cook-Levin theorem in (Kozen 2006)) to construct a CNF $\chi_{\mathfrak{M}}$ such that $\chi_{\mathfrak{M}} \wedge \chi_{\bar{y}}^{in} \wedge \chi_{\bar{z}}^{out}$ is satisfiable if and only if $\mathfrak{M}$, having started with $\bar{y}$ written on the input tape, terminates with $\bar{z}$ written on the output tape (here, $\chi_{\bar{y}}^{in}$ and $\chi_{\bar{z}}^{out}$ are conjunctions of propositional literals that encode the contents of the input tape in the initial state and the contents of the output tape in the final state, respectively). Let $x_1, \dots, x_n$ be the variables in $\chi_{\mathfrak{M}}$ and let $F_{\mathfrak{M}}$ be the result of replacing each occurrence of x_i in $\chi_{\mathfrak{M}}$ with $\mathsf{bnd}(?u_i)$. The variables encoding the contents of the input tape consists of two parts: variables $x'_1, \dots, x'_\ell$ representing pattern P_k as the first argument of t_2 and the variables corresponding to the free (not projected out) variables of the pattern $t_1(P_k)$ as the second argument. Without loss of generality, we assume that the variables encoding the contents of the output tape correspond to the free variables of P_k.

We now define P'_k by taking

$$\text{PROJ}_{\mathsf{var}(P_k)} \text{FILTER}_F\big(\big(U_1 \text{ JOIN} \dots \text{JOIN } U_n\big) \text{ JOIN } t_1(P_k)\big),$$

where all the U_i are as in the proof of Lemma 14, and F is a conjunction of $F_{\mathfrak{M}}$ and filters for each $1 \leq j \leq \ell$, which are $\mathsf{bnd}(?u'_j)$ if jth bit in the representation of P_k is 1, and

$\neg\mathsf{bnd}(?u'_j)$, otherwise. It should be clear that the size of P'_k is polynomial in the size of P_k, and they give the same answers on $\{(a_k, a_k, a_k)\}$. ❑

Combining this lemma with Theorem 13 we obtain the last result of this section.

Theorem 17 *Provided that* NP $=$ CONP*, operators* OPT_F *and* DIFF_F *are polynomially* $\mathcal{S}_\pi \cup \{O'\}$*-expressible for any* $O' \in \{\text{OPT}_\top, \text{DIFF}_\top\}$.

Expressivity of MINUS

Operators DIFF and OPT, studied in the previous sections, comprise non-monotonic tools in SPARQL 1.0. Version 1.1, however, also includes operator P_1 MINUS P_2, which checks whether a mapping from P_1 is extendable to a mapping from P_2 with an overlapping domain. In this section we compare the expressive power of MINUS and the 1.0 operators. In particular, we show that MINUS is polynomially expressible via DIFF_F and OPT_F, but does not give the full power of $\text{DIFF}_\top$ and $\text{OPT}_\top$, even in the presence of projection.

Theorem 18 MINUS *is polynomially* $\{\text{DIFF}_F\}$*- and* $\{\text{OPT}_F, \text{FILTER}\}$*-expressible.*

Proof We can express MINUS using the following equivalences (recall that JOIN is $\{\text{FILTER}, \text{OPT}_\top\}$-expressible):

$$P_1 \text{ MINUS } P_2 \equiv P_1 \text{ DIFF}_F\ P_2\theta,$$
$$P_1 \text{ MINUS } P_2 \equiv \text{FILTER}_{\neg\mathsf{bnd}(?u)} (P_1 \text{ OPT}_F\ (P_2\theta \text{ JOIN } \{(?u, ?v, ?w)\})),$$

where θ is a variable renaming for P_1 MINUS P_2 and

$$F = \bigwedge_{?v\in dom(\theta)} (\mathsf{bnd}(?v) \wedge \mathsf{bnd}(?v\theta) \to ?v = ?v\theta) \wedge \bigvee_{?v\in dom(\theta)} (\mathsf{bnd}(?v) \wedge \mathsf{bnd}(?v\theta))$$

(in particular, $F = \bot$ if $\mathsf{var}(P_1) \cap \mathsf{var}(P_2) = \emptyset$). For the first equivalence, observe that P_1 and $P_2\theta$ do not share variables and so, for any graph G, every solution mapping in $[\![P_1]\!]_G$ is compatible with any solution mapping in $[\![P_2\theta]\!]_G$. Thus, $[\![P_1 \text{DIFF}_F P_2\theta]\!]_G$ contains only those solution mappings μ in $[\![P_1]\!]_G$ that have no compatible solution mapping in $[\![P_2]\!]_G$, whose domain overlaps $dom(\mu)$, as defined by F.

For the second equivalence, note again that P_1 and $P_2\theta$ JOIN $\{(?u, ?v, ?w)\}$ contain no shared variables and that $\text{FILTER}_{\neg\mathsf{bnd}(?v)}$ leaves only the solution mappings in $[\![P_1 \text{DIFF}_F (P_2\theta \text{JOIN} \{(?u, ?v, ?w)\})]\!]_G$, for any G. The rest of the argument is similar to the case of $\text{DIFF}_\top$. ❑

Theorem 18 shows polynomial $\mathcal{S} \cup \{\text{DIFF}_F\}$- and $\mathcal{S} \cup \{\text{OPT}_F\}$-expressibility of MINUS. The former follows, of course, from (3) and the latter, but the theorem states that it is enough to have only DIFF_F. Then, by Theorem 6, MINUS is also $\mathcal{S}\cup\{\text{DIFF}_\top\}$- and $\mathcal{S}\cup\{\text{OPT}_\top\}$-expressible. We leave open the question of whether it can be done polynomially (possibly, with projection), but show non-expressibility of DIFF and OPT via MINUS.

Theorem 19 $\text{DIFF}_\top$ *and* $\text{OPT}_\top$ *are not* $\mathcal{S}_\pi \cup \{\text{MINUS}\}$*-expressible.*

Proof We claim that patterns P over $\{\text{MINUS}\} \cup \mathcal{S}$ enjoy the following property:

$$\text{if} \quad \mu_\emptyset \in [\![P]\!]_\emptyset \quad \text{then} \quad \mu_\emptyset \in [\![P]\!]_G, \text{ for any } G. \qquad (12)$$

The proof of the claim is by induction on the structure of the pattern P. The basis of induction, for a BGP B in P, follows from the observation that $\mu_\emptyset \in [\![B]\!]_\emptyset$ is only possible if $B = \{\}$, whence, $\mu_\emptyset \in [\![\{\}]\!]_G$, for any G. For the inductive step, consider all the operators.

- If $\mu_\emptyset \in [\![\text{FILTER}_F\ P_1]\!]_\emptyset$ then $\mu_\emptyset \in [\![P_1]\!]_\emptyset$ and $F^{\mu_\emptyset}$ is true, whence, by the induction hypothesis, $\mu_\emptyset \in [\![\text{FILTER}_F\ P_1]\!]_G$, for any G.
- If $\mu_\emptyset \in [\![P_1 \text{UNION} P_2]\!]_\emptyset$, then $\mu_\emptyset \in [\![P_i]\!]_\emptyset$ for either $i = 1$ or 2, whence, by the induction hypothesis, $\mu_\emptyset \in [\![P_i]\!]_G$, for any G, and so, $\mu_\emptyset \in [\![P_1 \text{ UNION } P_2]\!]_G$, for any G.
- If $\mu_\emptyset \in [\![P_1 \text{ JOIN } P_2]\!]_\emptyset$, then $\mu_\emptyset \in [\![P_1]\!]_\emptyset$ and $\mu_\emptyset \in [\![P_2]\!]_\emptyset$, whence, by the induction hypothesis, $\mu_\emptyset \in [\![P_1]\!]_G$ and $\mu_\emptyset \in [\![P_2]\!]_G$, for any G, and so, $\mu_\emptyset \in [\![P_1 \text{ JOIN } P_2]\!]_G$.
- If $\mu_\emptyset \in [\![P_1 \text{ MINUS } P_2]\!]_\emptyset$, then $\mu_\emptyset \in [\![P_1]\!]_\emptyset$, whence, by the induction hypothesis, $\mu_\emptyset \in [\![P_1]\!]_G$, for any G, and so, as $dom(\mu_\emptyset)$ does not overlap the domain of any other solution mapping, $\mu_\emptyset \in [\![P_1 \text{ MINUS } P_2]\!]_G$, for any G.
- If $\mu_\emptyset \in [\![\text{PROJ}_V\ P_1]\!]_\emptyset$, then $\mu_\emptyset \oplus \mu \in [\![P_1]\!]_\emptyset$, for μ with $dom(\mu) \cap V = \emptyset$. By (2), $\mu = \mu_\emptyset$. By the induction hypothesis, $\mu_\emptyset \in [\![P_1]\!]_G$ and $\mu_\emptyset \in [\![\text{PROJ}_V\ P_1]\!]_G$, for any G.

This completes the proof of the claim.

To show that $\text{DIFF}_\top$ is not $\mathcal{S}_\pi \cup \{\text{MINUS}\}$-expressible, consider pattern $P = \{\}\ \text{DIFF}_\top\ \{(?u, ?v, ?w)\}$. We have $[\![P]\!]_\emptyset = \{\mu_\emptyset\}$ and $[\![P]\!]_G = \emptyset$, for any $G \neq \emptyset$. If P were equivalent to a pattern P' over $\mathcal{S}_\pi \cup \{\text{MINUS}\}$, then since $\mu_\emptyset \in [\![P']\!]_\emptyset$, by (12), we would have $\mu_\emptyset \in [\![P']\!]_G$, for any G, contrary to P' being equivalent to P.

The proof for $\text{OPT}_\top$ is similar: it is enough to consider pattern $P = \{\}\ \text{OPT}_\top\ \{(?u, ?v, ?w)\}$. ❑

Conclusions and Future Work

We studied the problem of expressing non-monotone SPARQL operators. Our results show that projection has a dramatic effect, and polynomial expressibility of OPT_F and DIFF_F is connected to open problems in complexity theory: in fact, contrary to popular belief, the ternary OPT_F is not polynomially expressible via the binary $\text{OPT}_\top$ if $\Delta_2^p \neq \Sigma_2^p$.

Besides several open questions mentioned in the text, a different and interesting problem is whether *any* pattern over some $\mathcal{B}$ is expressible as a pattern of polynomial size over another $\mathcal{B}'$. Although our strong negative results carry over, the notion of polynomial expressibility in this paper assumes that operators in $\mathcal{B} \setminus \mathcal{B}'$ are applied only at the outermost level, without any nesting. In this stricter sense, however, even polynomial expressibility of $P_1\ \text{OPT}_\top\ P_2$ via $\text{DIFF}_\top$ is not clear because both P_i occur twice in expression (3).

References

Ahmetaj, S.; Fischl, W.; Pichler, R.; Simkus, M.; and Skritek, S. 2015. Towards reconciling SPARQL and certain answers. In *Proc. of the 24th Int. Conf. on World Wide Web (WWW 2015)*, 23–33. ACM.

Angles, R., and Gutierrez, C. 2008. The expressive power of SPARQL. In *Proc. of the 7th Int. Semantic Web Conf. (ISWC 2008)*, volume 5318 of *LNCS*, 114–129. Springer.

Arenas, M.; Conca, S.; and Pérez, J. 2012. Counting beyond a yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In *Proc. of the 21 Int. Conf. on World Wide Web (WWW 2012)*, 629–638. ACM.

Arenas, M.; Gottlob, G.; and Pieris, A. 2014. Expressive languages for querying the semantic web. In *Proc. of the 33rd ACM Symposium on Principles of Database Systems (PODS'14)*, 14–26. ACM.

Bischof, S.; Krötzsch, M.; Polleres, A.; and Rudolph, S. 2014. Schema-agnostic query rewriting in SPARQL 1.1. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2014), Part I*, volume 8796 of *LNCS*, 584–600. Springer.

Book, R. V.; Long, T. J.; and Selman, A. L. 1985. Qualitative relativizations of complexity classes. *Journal of Computer and System Sciences* 30(3):395–413.

Buil-Aranda, C.; Arenas, M.; and Corcho, Ó. 2011. Semantics and optimization of the SPARQL 1.1 federation extension. In *Proc. of the 8th Extended Semantic Web Conf. (ESWC 2011), Part II*, volume 6644 of *LNCS*, 1–15. Springer.

Buil Aranda, C.; Polleres, A.; and Umbrich, J. 2014. Strategies for executing federated queries in SPARQL1.1. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2014), Part II*, volume 8797 of *LNCS*, 390–405. Springer.

Cyganiak, R.; Wood, D.; and Lanthaler, M. 2014. RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C. http://www.w3.org/TR/rdf11-concepts.

Fenner, S. A.; Green, F.; Homer, S.; Selman, A. L.; Thierauf, T.; and Vollmer, H. 1999. Complements of multivalued functions. *Chicago Journal of Theoretical Computer Science* 1999(3).

Geerts, F.; Karvounarakis, G.; Christophides, V.; and Fundulaki, I. 2013. Algebraic structures for capturing the provenance of SPARQL queries. In *Proc. of Joint 2013 EDBT/ICDT Conferences (ICDT'13)*, 153–164. ACM.

Halpin, H., and Cheney, J. 2014. Dynamic provenance for SPARQL updates. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2014), Part I*, volume 8796 of *LNCS*, 425–440. Springer.

Harris, S., and Seaborne, A. 2013. SPARQL 1.1 query language. W3C recommendation, W3C. http://www.w3.org/TR/sparql11-query.

Hayes, P. J., and Patel-Schneider, P. F. 2014. RDF 1.1 semantics. W3C recommendation, W3C. http://www.w3.org/TR/rdf11-mt.

Kaminski, M., and Kostylev, E. V. 2016. Beyond well-designed SPARQL. In *Proc. of the 19th Int. Conf. on Database Theory (ICDT 2016)*. ACM.

Kaminski, M.; Kostylev, E. V.; and Cuenca Grau, B. 2016. Semantics and expressive power of subqueries and aggregates in SPARQL 1.1. In *Proc. of the 25th Int. Conf. on World Wide Web (WWW 2016)*. ACM.

Kollia, I., and Glimm, B. 2013. Optimizing SPARQL query answering over OWL ontologies. *Journal of Artificial Intelligence Research (JAIR)* 48:253–303.

Kontchakov, R.; Rezk, M.; Rodriguez-Muro, M.; Xiao, G.; and Zakharyaschev, M. 2014. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2014), Part I*, volume 8796 of *LNCS*, 552–567. Springer.

Kostylev, E. V., and Cuenca Grau, B. 2014. On the semantics of SPARQL queries with optional matching under entailment regimes. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2014), Part II*, volume 8797 of *LNCS*, 374–389. Springer.

Kostylev, E. V.; Reutter, J. L.; Romero, M.; and Vrgoc, D. 2015. SPARQL with property paths. In *Proc. of the 14th Int. Semantic Web Conf. (ISWC 2015), Part I*, volume 9366 of *LNCS*, 3–18. Springer.

Kostylev, E. V.; Reutter, J. L.; and Ugarte, M. 2015. CONSTRUCT queries in SPARQL. In *Proc. of the 18th Int. Conf. on Database Theory (ICDT 2015)*. ACM.

Kozen, D. 2006. *Theory of Computation*. Springer.

Krentel, M. W. 1988. The complexity of optimization problems. *Journal of Computer and System Sciences* 36(3):490–509.

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).

Losemann, K., and Martens, W. 2013. The complexity of regular expressions and property paths in SPARQL. *ACM Transactions on Database Systems* 38(4):24.

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

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

Polleres, A., and Wallner, J. P. 2013. On the relation between SPARQL 1.1 and answer set programming. *Journal of Applied Non-Classical Logics* 23(1-2):159–212.

Prud'hommeaux, E., and Seaborne, A. 2008. SPARQL query language for RDF. W3C recommendation, W3C. http://www.w3.org/TR/rdf-sparql-query.

Schmidt, M.; Meier, M.; and Lausen, G. 2010. Foundations of SPARQL query optimization. In *Proc. of the 13th Int. Conf. on Database Theory (ICDT 2010)*, 4–33. ACM.

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

Zhang, X., and Van den Bussche, J. 2014b. On the satisfiability problem for SPARQL patterns. arXiv:1406.1404.