Theory Comput Syst (2018) 62:772–809
DOI 10.1007/s00224-017-9802-9

Complexity and Expressive Power of Weakly Well-Designed SPARQL

Mark Kaminski[1] · **Egor V. Kostylev**[1]

Published online: 14 August 2017

Abstract SPARQL is the standard query language for RDF data. The distinctive feature of SPARQL is the OPTIONAL operator, which allows for partial answers when complete answers are not available due to lack of information. However, optional matching is computationally expensive—query answering is PSPACE-complete. The well-designed fragment of SPARQL achieves much better computational properties by restricting the use of optional matching—query answering becomes coNP-complete. On the downside, well-designed SPARQL captures far from all real-life queries—in fact, only about half of the queries over DBpedia that use OPTIONAL are well-designed. In the present paper, we study queries outside of well-designed SPARQL. We introduce the class of weakly well-designed queries that subsumes well-designed queries and includes most common meaningful non-well-designed queries: our analysis shows that the new fragment captures over 99% of DBpedia queries with OPTIONAL. At the same time, query answering for weakly well-designed SPARQL remains coNP-complete, and our fragment is in a certain sense maximal for this complexity. We show that the fragment's expressive power is strictly in-between well-designed and full SPARQL. Finally, we provide an intuitive normal form for weakly well-designed queries and study the complexity of containment and equivalence.

Keywords RDF query languages · SPARQL · Optional matching

This article is part of the Topical Collection on *Special Issue on Database Theory*

✉ Mark Kaminski
mark.kaminski@cs.ox.ac.uk

✉ Egor V. Kostylev
egor.kostylev@cs.ox.ac.uk

[1] Department of Computer Science, University of Oxford, Oxford, UK

1 Introduction

The Resource Description Framework (RDF) [14, 18, 31] is the W3C standard for representing linked data on the Web. RDF models information in terms of labelled graphs consisting of triples of resource identifiers (IRIs). The first and last IRIs in such a triple, called *subject* and *object*, represent entity resources, while the middle IRI, called *predicate*, represents a relation between the two entities.

SPARQL [17, 37] is the default query language for RDF graphs. First standardised in 2008 [37], SPARQL is now recognised as a key technology for the Semantic Web. This is witnessed by a recent adoption of a new version of the standard, SPARQL 1.1 [17], as well as by active development of SPARQL query engines in academia and industry, for instance, as part of the systems AllegroGraph (http://franz.com/agraph/allegrograph/), Apache Jena (http://jena.apache.org), RDF4J (http://rdf4j.org), or OpenLink Virtuoso (http://virtuoso.openlinksw.com).

In recent years, SPARQL has been subject to a substantial amount of theoretical research, based on the foundational work by Pérez et al. [32, 33]. In particular, we now know much about evaluation [1, 3, 4, 6, 8, 20, 22, 23, 25, 28, 34, 38], optimisation [8, 9, 12, 13, 24, 27, 35], federation [10, 11], expressive power [2, 20, 21, 25, 36, 39], and provenance tracking [15, 16] for queries from various fragments and extensions of SPARQL. These studies have had a great impact in the community, in fact influencing the evolution of SPARQL as a standard.

A distinctive feature of SPARQL as compared to SQL is the OPTIONAL operator (abbreviated as OPT in this paper). This operator was introduced to "*not reject (solutions) because some part of the query pattern does not match*" [37]. For instance, consider the SPARQL query

$$\begin{array}{l}\textsf{SELECT}\ ?i, ?n\\ \qquad \textsf{WHERE}\ (?i, \texttt{rdf:type}, \texttt{foaf:person})\ \textsf{OPT}\ (?i, \texttt{foaf:name}, ?n),\end{array} \tag{1}$$

which retrieves all person IDs from the graph together with their names; names, however, are optional—if the graph does not contain information about the name of a person, the person ID is still retrieved but the variable $?n$ is left undefined in the answer. For instance, query (1) has two answers over the graph G in Fig. 1a, where the second answer is partial (see Fig. 1b). However, if we extend G with a triple supplying a name for `P2`, the second answer will include this name.

The OPT operator accounts in a natural way for the open world assumption and the fundamental incompleteness of the Web. However, evaluating queries that use OPT is computationally expensive—Pérez et al. [33] showed PSPACE-completeness of SPARQL query evaluation, and Schmidt et al. [38] refined this result by proving PSPACE-hardness even for queries using no operators besides OPT. This is not surprising given that SPARQL queries are equivalent in expressive power to first-order logic queries, and translations in both directions can be done in polynomial time [2, 25, 36].

This spurred a search for restrictions on the use of OPT that would ensure lower complexity of query evaluation. It was also recognised that queries that are difficult to evaluate are often unintuitive. For instance, they may produce less specified answers

$(\texttt{P1}, \texttt{rdf:type}, \texttt{foaf:person})$
$(\texttt{P2}, \texttt{rdf:type}, \texttt{foaf:person})$
$(\texttt{P1}, \texttt{foaf:name}, Ana)$

(a)

$?i$	$?n$
P1	Ana
P2	

(b)

$(\texttt{P1}, \texttt{rdf:type}, \texttt{foaf:person})$
$(\texttt{P2}, \texttt{rdf:type}, \texttt{foaf:person})$
$(\texttt{P1}, \texttt{v\_card:name}, Anastasia)$

(c)

$?i$	$?n$
P1	$Anastasia$
P2	

(d)

Fig. 1 (**a**) Graph G; (**b**) answers to query (1) over G; (**c**) graph G'; and (**d**) answers to (2) over G'

(i.e., answers with fewer bound variables) as the graph over which they are evaluated grows larger.

Pérez et al. [33] introduced the *well-designed* fragment of SPARQL queries by imposing a syntactic restriction on the use of variables in OPT-expressions. Roughly speaking, each variable in the optional (i.e., right) argument of an OPT-expression should either appear in the mandatory (i.e., left) argument or be globally fresh for the query, i.e., appear nowhere outside of the argument. Well-designed queries have lower complexity of query evaluation—the problem is coNP-complete (provided all the variables in the query are selected). Moreover, such queries have a more intuitive behaviour than arbitrary SPARQL queries; in particular, they enjoy the monotonicity property that we observed for query (1): each partial answer over a graph can potentially be extended to undefined variables if the graph is completed with the missing information, and the more information we have the more specified are the answers. Well-designed queries can be efficiently transformed to an intuitive normal form allowing for a transparent graphical representation of queries as trees [27, 35]. Hence, many recent studies concentrate partially [23, 25, 27, 40, 41] or entirely [1, 8, 35] on well-designed queries.

Such a success of well-designed queries may lead to the impression that non-well-designed SPARQL queries are just a useless side effect of the early specification. But is this impression justified by the use of SPARQL in practice? To answer this question, a comprehensive analysis of real-life queries is required. We are aware of two works that analyse the distribution of operators in SPARQL queries asked over DBpedia [7, 34]. Both studies show that OPT is used in a non-negligible amount of practical queries. However, only Picalausa and Vansummeren [34] go further and analyse how many of these queries are well-designed; and the result is quite interesting—well-designed queries make up only about half of all queries with OPT. In other words, well-designed queries are common, but by far not exclusive.

The main goal of this paper is to investigate SPARQL queries beyond the well-designed fragment. We wanted to see if the well-designedness condition could be extended so as to include most practical queries while preserving good computational properties. The main result of our study is very positive—we identified a new fragment of SPARQL queries, called *weakly well-designed* queries, that covers over 99% of queries over DBpedia and has the same complexity of query evaluation as the well-designed fragment. We also show that our fragment is in a sense maximal for this complexity.

We next describe our results and techniques in more detail. Our first step was to identify typical real-life queries that are not well-designed. We analysed DBpedia query logs in recent USEWOD research datasets [29, 30] and found two interesting types of non-well-designed queries. The first type is exemplified by the following query:

$$\begin{aligned}&\mathsf{SELECT}\ ?i, ?n\\ &\quad \mathsf{WHERE}\ ((?i, \texttt{rdf:type}, \texttt{foaf:person})\ \mathsf{OPT}\ (?i, \texttt{foaf:name}, ?n))\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{OPT}\ (?i, \texttt{v\_card:name}, ?n).\end{aligned} \tag{2}$$

This query is clearly not well-designed because variable $?n$, binding the name of a person, appears in two different unrelated optional parts. Let us analyse answers to this query over different graphs. On graph G in Fig. 1a the result is exactly the same as for query (1), shown in Fig. 1b, simply because the IRI `v_card:name` is not present in G, and so cannot be matched against the second optional part of the query. Similarly, on graph G' in Fig. 1c, where the source of the name and the name itself are different, the result is as in Fig. 1d. In this case, the first optional part in the query does not match anything in the graph so the variable $?n$ is left unbound at this point; then the second optional is matched, and the variable is assigned with the name from `v_card`. More interestingly, query (2) evaluated over the graph $G \cup G'$ once again yields the result in Fig. 1b. Indeed, in this case, the first optional part has a match again and $?n$ is assigned the value Ana; then, this variable is already bound and there is no match for the second optional part that agrees with this value, meaning that the alternative `v_card` name is disregarded by the query. To summarise, query (2) is once again looking for person IDs and, optionally, their names. Now, however, names are collected from two different sources, `foaf` and `v_card`, where the first source is given preference over the second (maybe because it is considered more reliable or more informative, or for some other reason). In other words, if we know the `foaf` name of a person, it is returned as part of the answer regardless of their `v_card` name; however, if there is no `foaf` name, then the `v_card` name is also acceptable and should be returned; variable $?n$ is left unbound only if the name cannot be extracted from either source.

Of course, preference patterns encountered in real-life queries are often more complex. Still, we will see that in most cases they do not increase the complexity of query evaluation.

Our second example query is as follows:

$$\begin{aligned}&\mathsf{SELECT}\ ?i, ?n\\ &\quad \mathsf{WHERE}\ ((?i, \texttt{rdf:type}, \texttt{foaf:person})\ \mathsf{OPT}\ (?i, \texttt{foaf:name}, ?n))\\ &\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{FILTER}\ (\neg bound(?n) \vee \neg(?n = Ana)).\end{aligned} \tag{3}$$

The query uses FILTER, a standard SPARQL operator that admits only answers conforming to a specified constraint. Again, this query is not well-designed because the FILTER constraint mentions the variable $?n$, which occurs in the optional part of the query but not in the mandatory part. However, the intention of the query is quite clear: it searches for people whose names are not known to be *Ana*, including people whose names are unknown.

This use of FILTER is in fact very common in real-life queries. Moreover, it is intuitive as long as FILTER is essentially the outermost operator in the query, as it is in our example. We will see that in all such cases FILTER cannot lead to an increase in complexity.

Having isolated these typical uses of non-well-de-signed-ness, we identify a new fragment of SPARQL that (a) includes all queries of the above two types, (b) subsumes well-designed queries, and (c) has the same complexity of query evaluation as well-designed queries. We call such queries *weakly well-designed.* They are the maximal fragment without structural restrictions on conjunctive blocks and filter conditions that has the above properties. Our analysis shows that more than 99% of DBpedia queries with OPT are weakly well-designed.

Besides low complexity of query evaluation, we establish a few more useful properties of weakly well-designed queries, which are summarised in the following outline of the paper. After introducing the syntax and semantics of SPARQL in Section 2, we formally define our new fragment in Section 3. In Section 4, we show that, similarly to the well-designed case, weakly well-designed queries can be transformed to an intuitive normal form, which allows for a natural graphical representation as *constraint pattern trees*. Using this representation, in Section 5, we formally show that the step from well-designed to weakly well-designed queries does not increase complexity of query evaluation; minimal relaxations of weak well-designedness, however, already lead to a complexity jump. In Section 6, we compare the expressive power of our fragment (and its extensions with additional operators) with well-designed queries and unrestricted SPARQL queries; in all cases, we show that the expressivity of weakly well-designed queries lies strictly in-between well-designed and unrestricted queries. In Section 7, we study static analysis problems for weakly well-designed queries and establish Π_2^p-completeness of equivalence and containment. Finally, in Section 8, we detail our analysis of DBpedia logs.

This article significantly extends the conference paper [19]. Besides providing full proofs of our technical claims, we have extended the analysis section and updated the evaluation to use more recent datasets. Furthermore, we have removed the erroneous claim that queries over unions of weakly well-designed patterns have the same expressive power as unrestricted SPARQL queries; on the contrary, we show that the former are strictly less expressive than the latter.

2 SPARQL Query Language

We begin by formally introducing the syntax and semantics of SPARQL that we adopt in this paper. Our formal setup mostly follows [33], which has some differences from the W3C specification [17, 37]; in particular, we use two-placed OPT and two-valued FILTER (conditional OPT and errors in FILTER evaluation as in the standard are expressible in our formalisation [2, 21]), do not consider blank nodes (their presence in RDF graphs would not change any of our results), and adopt set semantics, leaving multiset answers for future work.

RDF Graphs An RDF graph is a labelled graph where nodes can also serve as edge labels. Formally, let $\mathbf{I}$ be a set of *IRIs*. Then an *RDF triple* is a tuple (s, p, o) from $\mathbf{I} \times \mathbf{I} \times \mathbf{I}$, where s is called *subject*, p *predicate*, and o *object*. An *RDF graph* is a finite set of RDF triples.

SPARQL Syntax Let $\mathbf{X}$ be an infinite set $\{?x, ?y, \ldots\}$ of *variables*, disjoint from $\mathbf{I}$. *Filter constraints* are conditions of the form

- $\top$, $?x = u$, $?x = ?y$, or $bound(?x)$ for $?x, ?y$ in $\mathbf{X}$ and $u \in \mathbf{I}$ (these constraints are called *atomic*),
- $\neg R_1$, $R_1 \wedge R_2$, or $R_1 \vee R_2$ for filter constraints R_1 and R_2.

A *basic pattern* is a possibly empty set of triples from $(\mathbf{I} \cup \mathbf{X}) \times (\mathbf{I} \cup \mathbf{X}) \times (\mathbf{I} \cup \mathbf{X})$ (to avoid notational clutter, in examples we will often omit braces when writing singleton basic patterns, e.g., we will write $(?x, u, ?y)$ instead of $\{(?x, u, ?y)\}$). Then, SPARQL *(graph) patterns* P are defined by the grammar

$$P \ ::= \ B \mid (P \ \mathsf{AND} \ P) \mid (P \ \mathsf{OPT} \ P) \mid (P \ \mathsf{UNION} \ P) \mid (P \ \mathsf{FILTER} \ R),$$

where B ranges over basic patterns and R over filter constraints. Additionally, we require all filter constraints to be *safe*, that is, $\mathsf{vars}(R) \subseteq \mathsf{vars}(P)$ for every pattern $(P \ \mathsf{FILTER} \ R)$, where $\mathsf{vars}(S)$ is the set of all variables in S (which can be a pattern, constraint, etc.) When needed, we distinguish between patterns by their top-level operator; e.g., we write OPT-pattern or FILTER-pattern.

We write $\mathcal{U}$ for the set of all patterns. We also distinguish the fragment $\mathcal{P}$ of $\mathcal{U}$ that consists of all UNION-*free* patterns, that is, patterns that do not use the UNION operator.

Projection is realised in SPARQL by means of *queries with select result form*, or *queries* for short, which are expressions of the form

$$\mathsf{SELECT} \ X \ \mathsf{WHERE} \ P, \tag{4}$$

where X is a set of variables and P is a graph pattern. We write $\mathcal{S}$ for the set of all queries. The set of all triples in basic patterns of a query Q is denoted $\mathsf{triples}(Q)$.

Note that every pattern P can be seen as a query of the form (4) where $X = \mathsf{vars}(P)$. Hence, all definitions that refer to "queries" implicitly extend to patterns in the obvious way.

SPARQL Semantics The semantics of graph patterns is defined in terms of *mappings*, that is, partial functions from variables to IRIs. The *domain* $\mathsf{dom}(\mu)$ of a mapping μ is the set of variables on which μ is defined. Two mappings μ_1 and μ_2 are *compatible* (written $\mu_1 \sim \mu_2$) if $\mu_1(?x) = \mu_2(?x)$ for all variables $?x \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$. If $\mu_1 \sim \mu_2$, then $\mu_1 \cup \mu_2$ constitutes a mapping with domain $\mathsf{dom}(\mu_1) \cup \mathsf{dom}(\mu_2)$ that coincides with μ_1 on $\mathsf{dom}(\mu_1)$ and with μ_2 on $\mathsf{dom}(\mu_2)$. Given two sets of mappings Ω_1 and Ω_2, we define their *join*, *union* and *difference* as follows:

$$\begin{aligned}
\Omega_1 \bowtie \Omega_2 &= \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \text{ and } \mu_1 \sim \mu_2\}, \\
\Omega_1 \cup \Omega_2 &= \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\}, \\
\Omega_1 \setminus \Omega_2 &= \{\mu_1 \mid \mu_1 \in \Omega_1, \mu_1 \not\sim \mu_2 \text{ for all } \mu_2 \in \Omega_2\}.
\end{aligned}$$

Based on these, the *left outer join* operation is defined as

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

Given a graph G, the *evaluation* $[\![P]\!]_G$ of a graph pattern P over G is defined as follows:

1. if B is a basic pattern, then $[\![B]\!]_G = \{\mu : \mathsf{vars}(B) \to \mathbf{I} \mid \mu(B) \subseteq G\}$;
2. $[\![(P_1 \text{ AND } P_2)]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$;
3. $[\![(P_1 \text{ OPT } P_2)]\!]_G = [\![P_1]\!]_G ⟕ [\![P_2]\!]_G$;
4. $[\![(P_1 \text{ UNION } P_2)]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$;
5. $[\![(P' \text{ FILTER } R)]\!]_G = \{\mu \mid \mu \in [\![P']\!]_G \text{ and } \mu \models R\}$,
   where μ *satisfies* a filter constraint R, denoted by $\mu \models R$, if one of the following holds:

   – R is $\top$;
   – R is $?x = u$, $?x \in \mathsf{dom}(\mu)$, and $\mu(?x) = u$;
   – R is $?x = ?y$, $\{?x, ?y\} \subseteq \mathsf{dom}(\mu)$, and $\mu(?x) = \mu(?y)$;
   – R is $\mathit{bound}(?x)$ and $?x \in \mathsf{dom}(\mu)$;
   – R is a Boolean combination of filter constraints evaluating to *true* under the usual interpretation of $\neg$, $\wedge$, and $\vee$.

Let $\mu|_X$ be the *projection* of a mapping μ to variables X, that is, $\mu|_X(?x) = \mu(?x)$ if $?x \in X$ and $\mu|_X(?x)$ is undefined if $?x \notin X$. The *evaluation* $[\![Q]\!]_G$ of a query Q of the form (4) is the set of all mappings $\mu|_X$ such that $\mu \in [\![P]\!]_G$.

Finally, a *solution* to a query (or pattern) Q over G is a mapping μ such that $\mu \in [\![Q]\!]_G$.

3 Weakly Well-Designed Patterns

We begin by recalling the notion of well-designed patterns and then formulate our generalisation. For now, we focus on the fragment $\mathcal{P}$ of UNION-free patterns (also known as the AND-OPT-FILTER fragment of SPARQL), leaving the operators UNION and SELECT for later sections.

Note that a given pattern can occur more than once within a larger pattern. In what follows we will sometimes need to distinguish between a (sub-)pattern P as a possibly repeated building block of another pattern P' and its *occurrences* in P', that is, unique subtrees in the parse tree. Then, the *left (right) argument* of an occurrence i is the subtree rooted in the left (right) child of the root of i in the parse tree, and an occurrence i is *inside* an occurrence j if the root of i is a descendant of the root of j.

Definition 1 (Pérez et al. [33]) A pattern P from $\mathcal{P}$ is *well-designed* (or *wd-pattern*, for short) if for every occurrence i of an OPT-pattern P_1 OPT P_2 in P the variables from $\mathsf{vars}(P_2) \setminus \mathsf{vars}(P_1)$ occur in P only inside (the labels of) i.

We write $\mathcal{P}_{\text{wd}}$ for the fragment of wd-patterns. Such patterns comply with the basic intuition for optional matching in SPARQL: “*do not reject (solutions) because some part of the query pattern does not match*” [37]; indeed, our canonical use case (1) is clearly well-designed. Evaluation of wd-patterns, that is, checking if $\mu \in [\![P]\!]_G$ for a mapping μ, graph G and pattern $P \in \mathcal{P}_{\text{wd}}$, is coNP-complete (in combined complexity), as opposed to PSPACE-completeness for $\mathcal{P}$ [33, 38]. The high complexity of unrestricted patterns is partially due to the fact that unrestricted combinations of OPT and FILTER allow to express nesting of the difference operator DIFF with semantics $[\![P_1 \text{ DIFF } P_2]\!]_G = [\![P_1]\!]_G \setminus [\![P_2]\!]_G$ (unless P_1 or P_2 are empty basic patterns, see [21] for details):

$$P_1 \text{ DIFF } P_2 \equiv (P_1 \text{ OPT } (P_2 \text{ AND } (?x, ?y, ?z))) \text{ FILTER } \neg \mathit{bound}(?x), \qquad (5)$$

where $?x$, $?y$ and $?z$ do not occur in $\mathsf{vars}(P_1) \cup \mathsf{vars}(P_2)$. This property is well-known [2, 21, 33], and has been usually believed to be an important source of non-well-designed patterns in practice. We challenge this belief by answering differently the question on the prevalent structure of real-life queries beyond the well-designed fragment. This question is not just of theoretical interest: as previous studies [34] show (and our analysis confirms), about half of queries with OPT asked over DBpedia are not well-designed.

Next we discuss two sources of non-well-designedness in patterns as revealed by the example queries (2) and (3) in the introduction—one based on OPT and another one on FILTER.

Source 1. There are two substantially different ways of nesting the OPT operator in patterns:

$$P_1 \text{ OPT } (P_2 \text{ OPT } P_3), \qquad (\texttt{Opt-R})$$

$$(P_1 \text{ OPT } P_2) \text{ OPT } P_3. \qquad (\texttt{Opt-L})$$

Non-well-designed nesting of type (Opt-R) is responsible for the PSPACE-hardness of query evaluation [33, 38]. Moreover, such nesting is not very intuitive unless well-designed. On the contrary, as we saw in the introduction, non-well-designed nesting of type (Opt-L) can be used for prioritising some parts of patterns to others, and is indeed used in real life. As we will see later, nesting of type (Opt-L) cannot lead to high complexity of evaluation.

Source 2. Well-designedness can be violated by using "dangerous" variables from the right argument of OPT in filter constraints. In particular, patterns of the form $(P_1 \text{ OPT } P_2)$ FILTER R with R using a variable from $\mathsf{vars}(P_2) \setminus \mathsf{vars}(P_1)$ are not well-designed, but rather frequent in practice. However, such patterns almost never occur inside the right argument of other OPT-patterns. We will see that if we restrict the usage of such filters to the "top level", we preserve the good computational properties of wd-patterns.

Motivated by these observations, we considerably generalise the notion of wd-patterns to allow for useful queries like (2) and (3) while retaining important properties of such patterns. We start with two auxiliary notions.

Definition 2 Given a pattern P, an occurrence i_1 in P *dominates* an occurrence i_2 if there exists an occurrence j of an OPT-pattern such that i_1 is inside the left argument of j and i_2 is inside the right argument.

Definition 3 An occurrence i of a FILTER-pattern P' FILTER R in P is *top-level* if there is no occurrence j of an OPT-pattern such that i is inside the right argument of j.

We are ready to give the main definition of this paper.

Definition 4 A pattern $P \in \mathcal{P}$ is *weakly well-designed* (or *wwd-pattern*, for short) if, for each occurrence i of an OPT-subpattern P_1 OPT P_2, the variables in $\mathsf{vars}(P_2) \setminus \mathsf{vars}(P_1)$ appear outside i only in

– subpatterns whose occurrences are dominated by i, and
– constraints of top-level occurrences of FILTER-patterns.

We write $\mathcal{P}_{\mathrm{wwd}}$ for the fragment of wwd-patterns. They extend wd-patterns by allowing variables from the right argument of an OPT-subpattern that are not "guarded" by the left argument to appear in certain positions outside of the subpattern. Note that the patterns of queries (2) and (3) are wwd-patterns. Also, patterns which allow only for OPT nesting of type (`Opt-L`) are always weakly well-designed, same as the pattern on the right hand side of (5), which expresses DIFF. However, patterns that have subpatterns of the atter form in the right argument of OPT are not weakly well-designed. Next we give a few more examples.

Example 1 Consider the following patterns and their parse trees in Fig. 2 (we write $?x \neq ?y$ for $\neg(?x = ?y)$):

$$((?x, a, a)\ \mathsf{OPT}\ ((?x, b, ?y)\ \mathsf{OPT}\ (?y, c, ?z)))\ \mathsf{OPT}\ (?x, d, ?z), \tag{6}$$

$$((?x, a, a)\ \mathsf{OPT}\ (?x, d, ?z))\ \mathsf{OPT}\ ((?x, b, ?y)\ \mathsf{OPT}\ (?y, c, ?z)), \tag{7}$$

$$(((?u, f, ?v)\ \mathsf{OPT}\ (?u, g, ?w))\ \mathsf{FILTER}\ ?v \neq ?w)\ \mathsf{OPT}\ (?u, h, ?s), \tag{8}$$

$$(?u, h, ?s)\ \mathsf{OPT}\ (((?u, f, ?v)\ \mathsf{OPT}\ (?u, g, ?w))\ \mathsf{FILTER}\ ?v \neq ?w). \tag{9}$$

Pattern (6) is not well-designed because of variable $?z$, but is weakly well-designed since the occurrence of $(?y, c, ?z)$ dominates $(?x, d, ?z)$. However, the similar pattern (7) is not weakly well-designed because the occurrence of the inner OPT-pattern with the second occurrence of $?z$ does not dominate the first. Pattern (8) is weakly well-designed since the FILTER-pattern (which is not dominated by the inner OPT-pattern) is top-level, but pattern (9) is not, because of variable $?w$ in a non-top-level FILTER.

Proposition 1 *Checking whether a* UNION*-free pattern P belongs to the fragment $\mathcal{P}_{\mathrm{wwd}}$ can be done in time $O(|P|^2)$, where $|P|$ is the length of the string representation of P.*

Proof First note that a UNION-free pattern P is weakly well-designed if and only if so is the pattern $\mathsf{rm\_toplevel\_filters}(P)$, which is obtained from P by removing all top-level occurrences of filters. The operation $\mathsf{rm\_toplevel\_filters}$ can be implemented in linear time by the recursive procedure in Fig. 3a.

Next consider the recursive procedure $\mathsf{is\_wwd}$ in Fig. 3b, where $\mathsf{sort}(S)$ denotes a sorted, repetition-free list representation of a set S.

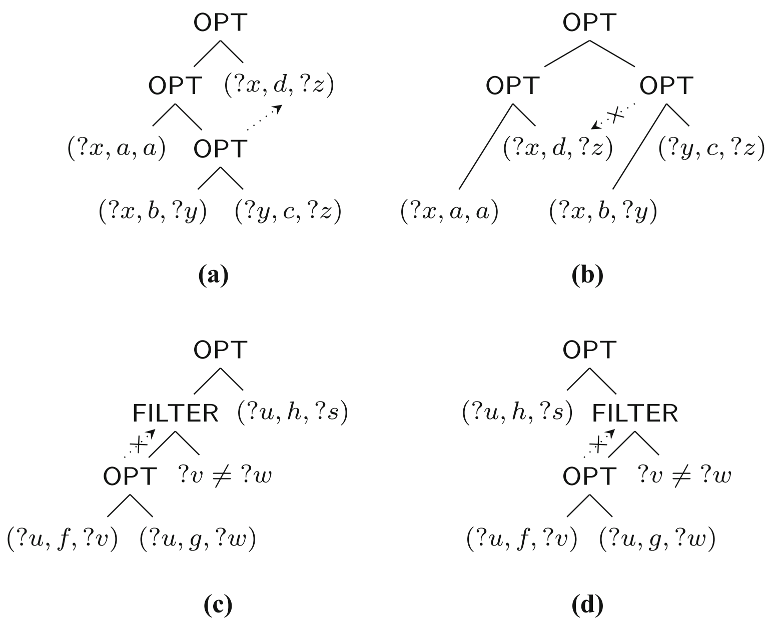

Fig. 2 (**a**)–(**d**) Parse trees for patterns (6)–(9) in Example 1, respectively; (*crossed*) *dotted arrows* represent the relevant (non-)dominance relations between subpatterns

Given a UNION-free pattern P without top-level filters, it is easily seen that is_wwd(P) returns a tuple of the form $(true, vs, ws)$ if and only if P is weakly well-designed, where ws is the sorted list of "unguarded" variables in P, that is, variables

$$\begin{aligned}
\mathsf{rm\_toplevel\_filters}(B) &= B, \quad // \text{ if } B \text{ is basic;}\\
\mathsf{rm\_toplevel\_filters}(P_1 \ \mathsf{AND}\ P_2) &= \mathsf{rm\_toplevel\_filters}(P_1)\ \mathsf{AND}\ \mathsf{rm\_toplevel\_filters}(P_2);\\
\mathsf{rm\_toplevel\_filters}(P_1 \ \mathsf{OPT}\ P_2) &= \mathsf{rm\_toplevel\_filters}(P_1)\ \mathsf{OPT}\ P_2;\\
\mathsf{rm\_toplevel\_filters}(P \ \mathsf{FILTER}\ R) &= \mathsf{rm\_toplevel\_filters}(P).
\end{aligned}$$

(a)

$$\begin{aligned}
\mathsf{is\_wwd}(B) &= (true, \mathsf{sort}(\mathsf{vars}(B)), \emptyset), \quad // \text{ if } B \text{ is basic;}\\
\mathsf{is\_wwd}(P_1\ \mathsf{AND}\ P_2) &= \mathsf{let}\ (wd_1, vs_1, ws_1) = \mathsf{is\_wwd}(P_1)\ \mathsf{in}\\
&\ \mathsf{let}\ (wd_2, vs_2, ws_2) = \mathsf{is\_wwd}(P_2)\ \mathsf{in}\\
&\ \mathsf{if}\ (ws_1 \cap (vs_2 \cup ws_2)) \cup (ws_2 \cap (vs_1 \cup ws_1)) = \emptyset\\
&\ \mathsf{then}\ (wd_1 \wedge wd_2, vs_1 \cup vs_2, ws_1 \cup ws_2)\ \mathsf{else}\ (\mathit{false}, \emptyset, \emptyset);\\
\mathsf{is\_wwd}(P_1\ \mathsf{OPT}\ P_2) &= \mathsf{let}\ (wd_1, vs_1, ws_1) = \mathsf{is\_wwd}(P_1)\ \mathsf{in}\\
&\ \mathsf{let}\ (wd_2, vs_2, ws_2) = \mathsf{is\_wwd}(P_2)\ \mathsf{in}\\
&\ \mathsf{if}\ (ws_2 \cap (vs_1 \cup ws_1)) = \emptyset\\
&\ \mathsf{then}\ (wd_1 \wedge wd_2, vs_1, (vs_2 \setminus vs_1) \cup ws_1 \cup ws_2)\ \mathsf{else}\ (\mathit{false}, \emptyset, \emptyset);\\
\mathsf{is\_wwd}(P\ \mathsf{FILTER}\ R) &= \mathsf{let}\ (wd, vs, ws) = \mathsf{is\_wwd}(P)\ \mathsf{in}\\
&\ \mathsf{if}\ ws \cap \mathsf{sort}(\mathsf{vars}(R)) = \emptyset\ \mathsf{then}\ (wd, vs, ws)\ \mathsf{else}\ (\mathit{false}, \emptyset, \emptyset).
\end{aligned}$$

(b)

Fig. 3 Procedures (**a**) rm_toplevel_filters and (**b**) is_wwd

occurring in the second argument of an OPT-subpattern P' of P but not in the first argument of P', and $vs = \mathsf{sort}(\mathsf{vars}(P)) \setminus ws$. Procedure is_wwd can be implemented in quadratic time since sort (which may take time $O(n \log n)$) is only applied to atomic subexpressions and set operations on sorted lists take linear time. $\square$

4 OPT-FILTER-Normal Form and Constraint Pattern Trees

One of the key properties of wd-patterns is that they can always be converted to a so-called OPT-normal form, in which all AND- and FILTER-subpatterns are OPT-free [33]. Also, FILTER-free patterns in OPT-normal form can be naturally represented as trees of a special form [27, 35], which give a good intuition for the evaluation and optimisation of such patterns. In this section, we show that these notions can be generalised to wwd-patterns.

Definition 5 A pattern $P \in \mathcal{P}$ is in OPT-FILTER-*normal form* (or OF-*normal form* for short) if it adheres to the grammar

$$\begin{aligned} P &::= F \mid (P \ \mathsf{FILTER}\ R) \mid (P \ \mathsf{OPT}\ S), \\ S &::= F \mid (S \ \mathsf{OPT}\ S), \\ F &::= (B \ \mathsf{FILTER}\ R), \end{aligned}$$

where B ranges over basic patterns and R over filter constraints.

In other words, the parse tree of a pattern in OF-normal form can be stratified as follows:

1. (occurrences of) basic patterns as the bottom layer,
2. a FILTER on top of each basic pattern as the middle layer,
3. a combination of OPT and FILTER as the top layer;

moreover, each occurrence of a FILTER-pattern in the top layer is top-level (according to Definition 3). Note that our normal form is AND-free: all conjunctions are expressed via basic patterns.

Example 2 None of the four patterns in Example 1 are in OF-normal form. However, the first three of them can be easily normalised by replacing each triple t with $t^\top$, where $P^\top$ is an abbreviation of P FILTER $\top$ for a pattern P. Also, compare the pattern

$$(((?x, a, a)^\top \ \mathsf{OPT}\ (?x, b, ?y)^\top) \ \mathsf{OPT}\ ((?x, b, ?z)^\top \ \mathsf{OPT}\ (?z, c, ?u)^\top)) \\ \mathsf{FILTER}\ ?u \neq ?x, \tag{10}$$

which is in OF-normal form, with the very similar pattern

$$((?x, a, a)^\top \ \mathsf{OPT}\ (?x, b, ?u)^\top) \ \mathsf{OPT} \\ (((?x, b, ?z)^\top \ \mathsf{OPT}\ (?z, c, ?u)^\top) \ \mathsf{FILTER}\ ?u \neq ?z),$$

which is not, because the outer FILTER is in the right argument of the outermost OPT.

As shown by Letelier et al. [27], FILTER-free patterns in OPT-normal form can be represented by means of so-called pattern trees. We next show that this representation can be naturally extended to patterns in OF-normal form.

Definition 6 Let P be a pattern in OF-normal form. The *constraint pattern tree (CPT)* $\mathcal{T}(P)$ of P is the directed, ordered, labelled, rooted tree recursively constructed as follows (in this definition we abuse notation and confuse patterns and their occurrences; strictly speaking, we create a fresh sub-tree for each occurrence, so the resulting object is always a tree):

1. if B is a basic pattern then $\mathcal{T}(B \text{ FILTER } R)$ is a single node v labelled by the pair (B, R);
2. if P' is not a basic pattern then $\mathcal{T}(P' \text{ FILTER } R)$ is obtained by adding a *special* node labelled by R as the last child of the root of $\mathcal{T}(P')$;
3. $\mathcal{T}(P_1 \text{ OPT } P_2)$ is the tree obtained from $\mathcal{T}(P_1)$ and $\mathcal{T}(P_2)$ by adding the root of $\mathcal{T}(P_2)$ as the last child of the root of $\mathcal{T}(P_1)$.

By definition, there is a one-to-one correspondence between patterns in OF-normal form and CPTs. Hence, such trees can be seen as a convenient representation of patterns in OF-normal form. Unlike parse trees, which represent the syntactic shape of patterns, CPTs show the semantic structure of OPT and FILTER nesting. Figure 4 shows how OPT nestings of types (`Opt-R`) and (`Opt-L`) are represented in both formats. Note that CPTs treat different FILTER-subpatterns differently: if the filter is over a basic pattern, the constraint of the FILTER is paired with this pattern; however, if the filter is over an OPT-subpattern, then the constraint is represented by a separate special node. Moreover, since in the second case the FILTER-pattern must be top-level, special nodes can only occur in CPTs as children of the root. For instance, the CPT of the example pattern (10) is given in Fig. 5a.

Proposition 2 *Let P be a pattern in* OF*-normal form. Then every special node in $\mathcal{T}(P)$ is a child of the root.*

Proof Let v be a special node in $\mathcal{T}(P)$. Then v is obtained from a subpattern P' FILTER R where P' is not basic. Hence, by definition of the OF-normal form, P must have the form

$$(\ldots((P' \text{ FILTER } R) \text{ OPT } S_1)\ldots) \text{ OPT } S_n$$

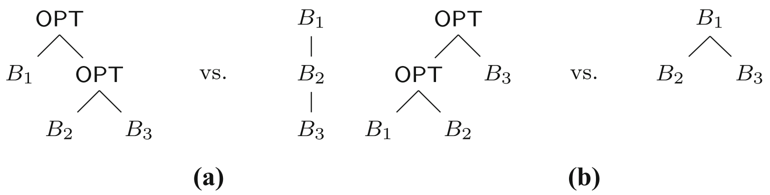

Fig. 4 Parse trees vs. constraint pattern trees for patterns (**a**) B_1 OPT (B_2 OPT B_3) and (**b**) (B_1 OPT B_2) OPT B_3, with B_1, B_2, and B_3 basic patterns

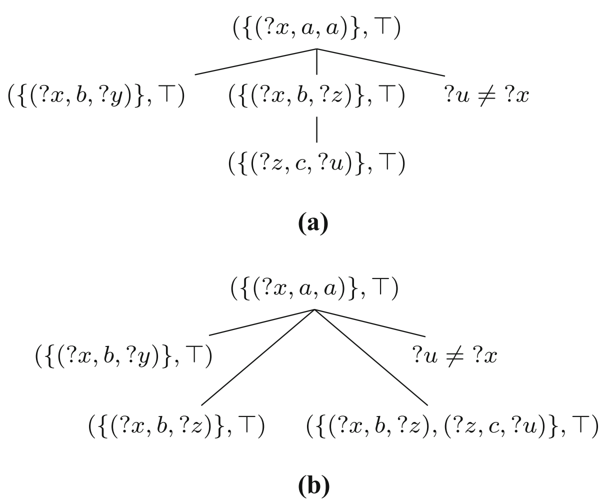

Fig. 5 Constraint pattern trees of (**a**) $(((?x, a, a)^\top$ OPT $(?x, b, ?y)^\top)$ OPT $((?x, b, ?z)^\top$ OPT $(?z, c, ?u)^\top))$ FILTER $?u \neq ?x$ (i.e., pattern (10)) and (**b**) equivalent pattern in "flat" form (13)

(for some $n \geq 0$) where $S_1, \ldots, S_n$ contain only FILTER-subpatterns over basic patterns. Thus, the root of $\mathcal{T}(P')$ is also the root of $\mathcal{T}(P)$, and the claim follows. $\square$

Next we show that each wwd-pattern can be converted to OF-normal form and hence can be represented by a CPT. To prove this statement we make use of a number of equivalences. Formally, a pattern P_1 is *equivalent* to a pattern P_2 (written $P_1 \equiv P_2$) if $[\![P_1]\!]_G = [\![P_2]\!]_G$ holds for any graph G. There are several equivalences, such as associativity and commutativity of AND, as well as filter decompositions, such as P FILTER $(R_1 \wedge R_2) \equiv (P$ FILTER $R_1)$ FILTER R_2, which hold for all patterns (see [38] for an extensive list). Moreover, the key equivalences used in [33] for normalising wd-patterns can easily be adapted to serve our needs.

Proposition 3 *Let P_1, P_2, P_3 be patterns and R a filter constraint such that* $\mathsf{vars}(P_2) \cap \mathsf{vars}(P_3) \subseteq \mathsf{vars}(P_1)$ *and* $\mathsf{vars}(P_2) \cap \mathsf{vars}(R) \subseteq \mathsf{vars}(P_1)$. *Then the following equivalences hold:*

$$(P_1 \text{ OPT } P_2) \text{ AND } P_3 \equiv (P_1 \text{ AND } P_3) \text{ OPT } P_2,$$
$$(P_1 \text{ OPT } P_2) \text{ FILTER } R \equiv (P_1 \text{ FILTER } R) \text{ OPT } P_2.$$

Proof Both equivalences are essentially shown in [33]. While stated for well-designed patterns, the proof only exploits the properties $\mathsf{vars}(P_2) \cap \mathsf{vars}(P_3) \subseteq \mathsf{vars}(P_1)$ and $\mathsf{vars}(P_2) \cap \mathsf{vars}(R) \subseteq \mathsf{vars}(P_1)$, which are satisfied not only by well-designed patterns, but also by weakly well-designed patterns. $\square$

Since all the equivalences preserve weak well-designedness, we obtain the desired result.

Proposition 4 *Each wwd-pattern P is equivalent to a wwd-pattern in* OF*-normal form of size $O(|P|)$.*

Proof We call a pattern P *pre-normal* if it adheres to the grammar that is the same as the one in Definition 5 except that the category F is extended as follows:

$$F ::= B \mid (F \,\mathsf{FILTER}\, R) \mid (F \,\mathsf{AND}\, F).$$

Given a pattern P, let $||P||$ be the sum of the sizes of all AND-subpatterns and all FILTER-subpatterns of P (where different occurrences of each pattern are counted separately). Consider a wwd-pattern P that is not pre-normal. Then P contains a subpattern P' of one of the following two forms (modulo commutativity of AND): $(P_1 \;\mathsf{OPT}\; P_2) \;\mathsf{AND}\; P_3$ and $(P_1 \;\mathsf{OPT}\; P_2) \;\mathsf{FILTER}\; R$ with P' not top-level. In both cases we can rewrite P to a pattern S not increasing $|\cdot|$ and strictly decreasing $||\cdot||$ as follows.

- Let $P' = (P_1 \;\mathsf{OPT}\; P_2) \;\mathsf{AND}\; P_3$. Since P is weakly well-designed and the occurrence of P_3 is not dominated by the occurrence of $P_1 \;\mathsf{OPT}\; P_2$, we have $\mathsf{vars}(P_3) \cap \mathsf{vars}(P_2) \subseteq \mathsf{vars}(P_1)$. Therefore, using the first equivalence in Proposition 3, we can rewrite P to a pattern S by replacing P' with $(P_1 \;\mathsf{AND}\; P_3) \;\mathsf{OPT}\; P_2$. Moreover, we have $|P| = |S|$ and $||P|| > ||S||$.
- Let $P' = (P_1 \;\mathsf{OPT}\; P_2) \;\mathsf{FILTER}\; R$ where the occurrence of P' is not top-level. Since P is weakly well-designed, we then have $\mathsf{vars}(R) \cap \mathsf{vars}(P_2) \subseteq \mathsf{vars}(P_1)$, and thus, with the second equivalence in Proposition 3, we can rewrite P to a pattern S by replacing P' with $(P_1 \;\mathsf{FILTER}\; R) \;\mathsf{OPT}\; P_2$. Moreover, we have $|P| = |S|$ and $||P|| > ||S||$.

Since this rewriting strictly decreases $||\cdot||$, its repeated application to P terminates and yields a pre-normal pattern S equivalent to P with $|S| = |P|$.

Finally, S can be transformed to OF-normal form by replacing every occurrence of an AND-FILTER combination of basic patterns by $B \;\mathsf{FILTER}\; R$ where B consists of all triples in the basic patterns and R is a conjunction of all the filter conditions (if there are no filters in the combination, then R is $\top$). Clearly, this transformation is equivalence-preserving and linear in $|S|$. □

Relying on this proposition, in the rest of the paper we silently assume that all wwd-patterns are in OF-normal form and hence can be represented by CPTs.

We next transfer the notion of weak well-designedness to CPTs. Given a pattern P in OF-normal form, let $\prec$ be the strict topological sorting of the nodes in $\mathcal{T}(P)$ computed by a depth first search traversal visiting the children of a node according to their ordering (i.e., $v \prec u$ holds if v is visited before u).

Lemma 1 *Let P be a pattern in* OF*-normal form and $P' = P_1$* OPT *P_2 be a subpattern of P. Then $v \prec w$ for every two nodes v, w in $\mathcal{T}(P)$ such that v is in the subtree of $\mathcal{T}(P)$ corresponding to P_1 and w is in the subtree corresponding to P_2.*

Proof The claim follows since $\mathcal{T}(P')$ is constructed by attaching $\mathcal{T}(P_2)$ as the last child to the root of $\mathcal{T}(P_1)$. □

In the following proposition, $\mathsf{vars}(u)$ for a node u of a CPT stands for the set of all variables in the label of u.

Proposition 5 *A pattern P in OF-normal form is weakly well-designed if and only if, for each edge (v, u) with non-special u in the CPT $\mathcal{T}(P)$, every variable $?x \in \mathsf{vars}(u) \setminus \mathsf{vars}(v)$ occurs only in nodes w such that $v \prec w$. The pattern is well-designed if and only if for every variable $?x$ in P the set of all nodes v in $\mathcal{T}(P)$ with $?x \in \mathsf{vars}(v)$ is connected.*

Proof For the forward direction of the first statement, suppose P is weakly well-designed. We proceed by induction on the structure of P and consider the following cases.

- Let $P = B$ FILTER R where B is basic. Then the claim is vacuous.
- Let $P = P_1$ FILTER R where P_1 is not basic. By the inductive hypothesis, the claim holds for $\mathcal{T}(P_1)$. Moreover, $\mathcal{T}(P)$ differs from $\mathcal{T}(P_1)$ only in the special node labelled with R, and the claim follows by Proposition 2.
- Let $P = P_1$ OPT P_2. By the inductive hypothesis, the claim holds for $\mathcal{T}(P_1)$ and $\mathcal{T}(P_2)$. Thus, by Lemma 1, it suffices to show that for every edge (v, u) in $\mathcal{T}(P_2)$ (with non-special u by definition), no variable $?x \in \mathsf{vars}(u) \setminus \mathsf{vars}(v)$ occurs in $\mathcal{T}(P_1)$. Suppose for contradiction that this property is violated for some (v, u) and $?x$. Then P_2 has a subpattern $P' = P'_1$ OPT P'_2 such that $\mathcal{T}(P'_1)$ is a subtree of $\mathcal{T}(P)$ rooted at v and $\mathcal{T}(P'_2)$ is the complete subtree of $\mathcal{T}(P)$ rooted at u. Moreover, $?x$ occurs in P_1, and thus outside P'. Since all FILTER-subpatterns in P are safe, we can assume without loss of generality that the occurrence of $?x$ in P_1 is not in a filter constraint. However, this contradicts the assumption that P is weakly well-designed since the occurrence of $?x$ in P_1 is not dominated by the occurrence of P'.

For the backward direction of the first claim, suppose P is not a wwd-pattern. Then P has a subpattern $P' = P'_1$ OPT P'_2, with v the root of $\mathcal{T}(P')$ in $\mathcal{T}(P)$ and u the child of v corresponding to $\mathcal{T}(P'_2)$, and a variable $?x \in \mathsf{vars}(P'_2) \setminus \mathsf{vars}(P'_1)$ such that $?x \in \mathsf{vars}(u)$ and, for some subpattern P_1 OPT P_2 of P, $?x$ occurs in P_1 and P' occurs in P_2. Since $?x \in \mathsf{vars}(P'_2) \setminus \mathsf{vars}(P'_1)$ and $?x \in \mathsf{vars}(u)$, we have $?x \in \mathsf{vars}(u) \setminus \mathsf{vars}(v)$. Thus, by Lemma 1, we have $v \not\prec w$, where w is a node in $\mathcal{T}(P_1)$ with an occurrence of $?x$.

The second claim can be proved analogously. □

Note that if a pattern is FILTER-free, its OF-normal form coincides with the OPT-normal form in [33] (modulo tautological filters), and its CPT is the pattern tree from [27, 35]. In fact, the second part of Proposition 5 generalises an observation from [27] to the case with filters. An important difference to pattern trees is that in our case the order of children of a node is semantically relevant since wwd-patterns do not satisfy the equivalence

$$(P_1 \mathsf{\ OPT\ } P_2) \mathsf{\ OPT\ } P_3 \equiv (P_1 \mathsf{\ OPT\ } P_3) \mathsf{\ OPT\ } P_2. \tag{11}$$

This equivalence, established in [32], holds whenever $(\mathsf{vars}(P_2) \cap \mathsf{vars}(P_3)) \subseteq \mathsf{vars}(P_1)$, which is always the case for wd-patterns but not for wwd-patterns, as can be seen on query (2).

We conclude this section with a property that is unique to wwd-patterns: each wwd-pattern is equivalent to a pattern whose corresponding CPT has depth one.

Definition 7 A pattern in $\mathcal{P}$ is in *depth-one normal form* if it has the structure

$$(\cdots((B \ \mathsf{op}_1 \ S_1) \ \mathsf{op}_2 \ S_2) \cdots) \ \mathsf{op}_n \ S_n, \tag{12}$$

where B is a basic pattern and each $\mathsf{op}_i \ S_i$, $1 \leq i \leq n$, is either $\mathsf{OPT} \ (B_i \ \mathsf{FILTER} \ R_i)$ with B_i a basic pattern and R_i a filter constraint, or just $\mathsf{FILTER} \ R_i$.

To show that each wwd-pattern can be brought to this form, we exploit the following observation in [33].

Lemma 2 (Pérez et al. [33]) *Let P be a pattern in $\mathcal{P}$, G a graph, and μ_1, μ_2 two mappings in $[\![P]\!]_G$. Then $\mu_1 \sim \mu_2$ if and only if $\mu_1 = \mu_2$.*

This lemma allows us to prove the following crucial equivalence.

Proposition 6 *For patterns P_1, P_2 and P_3 in $\mathcal{P}$ with $\mathsf{vars}(P_1) \cap \mathsf{vars}(P_3) \subseteq \mathsf{vars}(P_2)$ it holds that*

$$P_1 \ \mathsf{OPT} \ (P_2 \ \mathsf{OPT} \ P_3) \ \equiv \ (P_1 \ \mathsf{OPT} \ P_2) \ \mathsf{OPT} \ (P_2 \ \mathsf{AND} \ P_3). \tag{13}$$

Proof We first show that any solution to the left hand side is also a solution to the right hand side. Let G be a graph and let $\mu \in [\![P_1 \ \mathsf{OPT} \ (P_2 \ \mathsf{OPT} \ P_3)]\!]_G$. We distinguish three cases.

- Let $\mu \in [\![P_1]\!]_G \bowtie ([\![P_2]\!]_G \bowtie [\![P_3]\!]_G)$. Then, by Lemma 2, we have $[\![P_2]\!]_G = [\![P_2]\!]_G \bowtie [\![P_2]\!]_G$. Consequently, $\mu \in ([\![P_1]\!]_G \bowtie [\![P_2]\!]_G) \bowtie ([\![P_2]\!]_G \bowtie [\![P_3]\!]_G)$, and the claim follows.
- Let $\mu \in [\![P_1]\!]_G \bowtie ([\![P_2]\!]_G \setminus [\![P_3]\!]_G)$. Then $\mu = \mu_1 \cup \mu_2$ such that $\mu_1 \in [\![P_1]\!]_G$, $\mu_2 \in [\![P_2]\!]_G$, and for every $\mu_3 \in [\![P_3]\!]_G$, $\mu_2 \not\sim \mu_3$. Since every mapping in $[\![P_2 \ \mathsf{AND} \ P_3]\!]_G$ is an extension of some mapping in $[\![P_3]\!]_G$, no mapping in $[\![P_2 \ \mathsf{AND} \ P_3]\!]_G$ is compatible with μ_2, and hence with μ. Therefore, $\mu \in ([\![P_1]\!]_G \bowtie [\![P_2]\!]_G) \setminus [\![P_2 \ \mathsf{AND} \ P_3]\!]_G$, and the claim follows.
- Let $\mu \in [\![P_1]\!]_G \setminus [\![P_2 \ \mathsf{OPT} \ P_3]\!]_G$. Then $\mu \in [\![P_1]\!]_G$ and is incompatible with any mapping in $[\![P_2 \ \mathsf{OPT} \ P_3]\!]_G$. Moreover, since $\mathsf{vars}(P_1) \cap \mathsf{vars}(P_3) \subseteq \mathsf{vars}(P_2)$, μ is incompatible with any mapping in $[\![P_2]\!]_G$, and consequently also with any mapping in $[\![P_2 \ \mathsf{AND} \ P_3]\!]_G$. Therefore, $\mu \in ([\![P_1]\!]_G \setminus [\![P_2]\!]_G) \setminus [\![P_2 \ \mathsf{AND} \ P_3]\!]_G$, and the claim follows.

For the other direction, suppose $\mu \in [\![(P_1 \ \mathsf{OPT} \ P_2) \ \mathsf{OPT} \ (P_2 \ \mathsf{AND} \ P_3)]\!]_G$. We distinguish three cases.

- Let $\mu \in ([\![P_1]\!]_G \bowtie [\![P_2]\!]_G) \bowtie ([\![P_2]\!]_G \bowtie [\![P_3]\!]_G)$. Then, by Lemma 2, we have $\mu \in [\![P_1]\!]_G \bowtie ([\![P_2]\!]_G \bowtie [\![P_3]\!]_G)$, and the claim follows.
- Let $\mu \in ([\![P_1]\!]_G \bowtie [\![P_2]\!]_G) \setminus ([\![P_2]\!]_G \bowtie [\![P_3]\!]_G)$. Then $\mu = \mu_1 \cup \mu_2$ such that $\mu_1 \in [\![P_1]\!]_G$, $\mu_2 \in [\![P_2]\!]_G$, and μ is incompatible with every mapping in $[\![P_2]\!]_G \bowtie [\![P_3]\!]_G$. Since $\mathsf{vars}(P_1) \cap \mathsf{vars}(P_3) \subseteq \mathsf{vars}(P_2)$, this implies that $[\![P_2]\!]_G \bowtie [\![P_3]\!]_G$

is empty, that is, μ_2 is incompatible with every mapping in $[\![P_3]\!]_G$. Therefore, $\mu_2 \in [\![P_2]\!]_G \setminus [\![P_3]\!]_G$, and thus $\mu \in [\![P_1]\!]_G \bowtie ([\![P_2]\!]_G \setminus [\![P_3]\!]_G)$. The claim follows.
– Let $\mu \in [\![P_1]\!]_G \setminus [\![P_2]\!]_G$. Since every mapping in $[\![P_2 \text{ OPT } P_3]\!]_G$ extends a mapping in $[\![P_2]\!]_G$, we have that $\mu \in [\![P_1]\!]_G \setminus [\![P_2 \text{ OPT } P_3]\!]_G$, and the claim follows. □

Applied from left to right, equivalence (13) preserves weak well-designedness (but not well-designedness). Each such application transforms a weakly well-designed OPT nesting of type (`Opt-R`) to a nesting of type (`Opt-L`), decreasing the depth of the CPT.

Corollary 1 *Every wwd-pattern is equivalent to a wwd-pattern in depth-one normal form.*

For instance, pattern (10) is equivalent to the pattern

$$((((?x, a, a)^\top \text{ OPT } (?x, b, ?y)^\top) \text{ OPT } (?x, b, ?z)^\top) \text{ OPT } \{(?x, b, ?z), (?z, c, ?u)\}^\top) \text{ FILTER } ?u \neq ?x,$$

represented by the CPT in Fig. 5b. Such “flat” patterns are attractive in practice because of their regular structure. However, “flattening” a pattern can incur an exponential blow-up in size. Hence, in the rest of the paper we consider arbitrary wwd-patterns in OF-normal form rather than restricting our attention to depth-one-normal patterns.

5 Evaluation of wwd-Patterns

In this section, we look at the query answering problem for wwd-patterns and their extensions with union and projection. We show that in all three cases, complexity remains the same as for wd-patterns. To obtain these results, we develop several new techniques.

Formally, we look at the following decision problem for a given SPARQL fragment $\mathcal{L}$.

EVAL($\mathcal{L}$)	
Input:	Graph G, query $Q \in \mathcal{L}$, and mapping μ
Question:	Does μ belong to $[\![Q]\!]_G$?

It is known that EVAL($\mathcal{U}$) for general patterns $\mathcal{U}$ is PSPACE-complete [33], and the result easily propagates to queries with projection (i.e., $\mathcal{S}$) [27]. For wd-patterns, the evaluation problem is coNP-complete, and can be solved by exploiting the following idea of [27].

Suppose we are given a wd-pattern P in OPT-normal form (for simplicity, assume that P is FILTER-free), a graph G, and a mapping μ. First, we look for a subtree of $\mathcal{T}(P)$ that includes the root of $\mathcal{T}(P)$, contains precisely the variables in $\text{dom}\mu$, and “matches” G under μ (i.e., images of all its triples under μ are contained in G). This is doable in polynomial time. If such a subtree does not exist, then μ cannot be a solution. Otherwise, the subtree witnesses that μ is a part of a solution to P. Finally, to verify that μ is a complete solution, we need to check that the subtree is maximal, that is, cannot be extended to any more nodes in $\mathcal{T}(P)$ with a match in G. There are linearly many such nodes to check, and each check can be performed in coNP. So, the overall algorithm runs in coNP.

Inspired by this idea, we next show that the low evaluation complexity of wd-patterns transfers to wwd-patterns by developing a coNP algorithm for $\textsc{Eval}(\mathcal{P}_{\text{wwd}})$.

Let P be a wwd-pattern in OF-normal form. An *r-subtree* of $\mathcal{T}(P)$ is a subtree containing the root of $\mathcal{T}(P)$ and all its special children. Every r-subtree $\mathcal{T}(P')$ of $\mathcal{T}(P)$ is also a CPT representing a wwd-pattern P' that can be obtained from P by dropping the right arguments of some OPT-subpatterns (a transformation known from [33]). A *child* of an r-subtree $\mathcal{T}(P')$ of $\mathcal{T}(P)$ is a node in $\mathcal{T}(P)$ that is not contained in $\mathcal{T}(P')$ but whose parent is.

Definition 8 A mapping μ is a *potential partial solution* (or *pp-solution* for short) to a wwd-pattern P over a graph G if there is an r-subtree $\mathcal{T}(P')$ of $\mathcal{T}(P)$ such that $\mathsf{dom}(\mu) = \mathsf{vars}(P')$, $\mu(\mathsf{triples}(P')) \subseteq G$, and $\mu \models R$ for the constraint R of each ordinary node in $\mathcal{T}(P')$.

A pp-solution μ to P over G can be witnessed by several r-subtrees. However, the union of such r-subtrees is also a witness. Hence, there exists a unique maximal witnessing r-subtree, denoted $\mathcal{T}(P_\mu)$, with P_μ being the corresponding wwd-pattern.

Potential partial solutions generalise "partial solutions" as defined in [33] for wd-patterns. There, every "partial solution" is either a solution or can be extended to one. This is not the case for wwd-patterns. While every solution is clearly a pp-solution, not every pp-solution can be extended to a real one. Real solutions may not just extend pp-solutions by assigning previously undefined variables but can also override variable bindings established in some node v of $\mathcal{T}(P_\mu)$ by extending to a child of $\mathcal{T}(P_\mu)$ that precedes v according to the order $\prec$.

An additional complication is the presence of non-well-designed top-level filters. Note that pp-solutions are only required to satisfy the constraints of ordinary nodes in the corresponding CPT, thus ignoring top-level filters. Indeed, requiring pp-solutions to satisfy constraints of top-level filters would be too strong since real solutions do not generally satisfy this property, as demonstrated by the following example.

Example 3 Consider the graph $G = \{(1, a, 1), (3, a, 3)\}$ and wwd-pattern

$$P = (((?x, a, 1)\ \mathsf{OPT}\ (?y, a, 2))\ \mathsf{FILTER}\, \neg bound(?y))\ \mathsf{OPT}\ (?y, a, 3).$$

The mapping $\mu = \{?x \mapsto 1, ?y \mapsto 3\}$ is a solution to P over G, but $\mu \not\models \neg bound(?y)$.

We now present a characterisation of solutions for wwd-patterns in terms of pp-solutions that (a) takes into account that not every pp-solution can be extended to a real solution and (b) ensures correct treatment of non-well-designed top-level filters. For this we need some more notation. Given a wwd-pattern P, a node v in $\mathcal{T}(P)$, a graph G, and a pp-solution μ to P over G, let $\mu|_v$ be the projection $\mu|_X$ of μ to the set X of all variables appearing in nodes u of $\mathcal{T}(P_\mu)$ such that $u \prec v$. A mapping μ_1 is *subsumed* by a mapping μ_2 (written $\mu_1 \sqsubseteq \mu_2$) if $\mu_1 \sim \mu_2$ and $\mathsf{dom}(\mu_1) \subseteq \mathsf{dom}(\mu_2)$ (this notion is from [5, 33]).

Intuitively, a pp-solution μ needs to satisfy two conditions to be a real solution to a wwd-pattern P. First, $\mu|_v$ (as opposed to μ for wd-patterns) must be non-extendable to v for any child v of $\mathcal{T}(P_\mu)$. Indeed, if such an extension exists, then it is either possible to provide bindings for some variables that are undefined in μ, or some variables from $\mathsf{dom}(\mu)$ can be assigned different values of higher "priority" than the corresponding values in μ. Second, every top-level filter R labelling a node s needs to be satisfied by $\mu|_s$, which is precisely the part of μ bound by the subpattern of P that is paired with R in the FILTER-pattern. The following lemma formalises this intuition.

Lemma 3 *A mapping μ is a solution to a wwd-pattern P over a graph G if and only if*

1. *μ is a pp-solution to P over G;*
2. *for every child v of $\mathcal{T}(P_\mu)$ labelled with (B, R) there is no μ' such that $\mu|_v \sqsubseteq \mu'$, $\mu' \models R$, and $\mu'(B) \subseteq G$;*
3. *$\mu|_s \models R$ for every special node s in $\mathcal{T}(P)$ labelled with R.*

Proof In this proof we write $\mathcal{T}_v$ for the complete subtree of a CPT $\mathcal{T}$ rooted at a node v (i.e., the subtree over all the descendants of v including v itself) and $\mathcal{T}_{\prec v}$ for the subtree of $\mathcal{T}$ consisting of all nodes u such that $u \prec v$.

For the forward direction, suppose μ is a solution to P over G. Clearly, μ is a pp-solution to P over G, so it suffices to show that conditions 2 and 3 hold.

For condition 2, assume for contradiction that v is a child of $\mathcal{T}(P_\mu)$ labelled with (B, R) and μ' a mapping such that $\mu|_v \sqsubseteq \mu'$, $\mu' \models R$, and $\mu'(B) \subseteq G$. Moreover, without loss of generality, let $\mathsf{dom}(\mu') = \mathsf{dom}(\mu) \cup \mathsf{vars}(B)$. Let u be the parent of v in $\mathcal{T}(P)$, and let $\mathcal{T}$ be the largest subtree of $\mathcal{T}(P)$ that is rooted at u and has v as the last child of u. Then $[\![\mathcal{T}]\!]_G = [\![\mathcal{T}_{\prec v}]\!]_G ⟕ [\![\mathcal{T}_v]\!]_G$. Moreover, since u is contained in $\mathcal{T}(P_\mu)$, there is a mapping $\mu_1 \sqsubseteq \mu$ such that $\mu_1 \in [\![\mathcal{T}]\!]_G$. Since v is not contained in $\mathcal{T}(P_\mu)$, we have $\mu_1 \sqsubseteq \mu|_v$ and, since $\mathcal{T}(P_\mu)$ is the largest r-subtree witnessing μ, μ_1 is not compatible with any mapping in $[\![\mathcal{T}_v]\!]_G$. On the other hand, μ' satisfies the label of v, and thus, since $\mathcal{T}_v$ contains no top-level filters, $\mu'|_{\mathsf{vars}(v)}$ can be extended to a mapping of $\mu'' \in [\![\mathcal{T}_v]\!]_G$. Moreover, since P is weakly well-designed, $\mathsf{vars}(\mathcal{T}_v) \cap \mathsf{dom}(\mu|_v) \subseteq \mathsf{vars}(v)$, and hence $\mathsf{dom}(\mu'') \cap \mathsf{dom}(\mu_1) \subseteq \mathsf{dom}(\mu')$. Thus, since $\mu|_v$ is compatible with μ', μ_1 is compatible with μ'', in contradiction to the above observation that μ_1 is not compatible with any mapping in $[\![\mathcal{T}_v]\!]_G$.

For condition 3, let s be a special node in $\mathcal{T}(P)$ labelled with R. Since μ is a solution to P, there is some $\mu_1 \subseteq \mu$ such that $\mu_1 \in [\![\mathcal{T}(P)_{\prec s}]\!]_G$ and $\mu_1 \models R$. Hence, it suffices to show that $\mu_1 = \mu|_s$. Clearly, $\mu_1 \subseteq \mu|_s$ (as $\mu|_s$ is the largest mapping compatible with μ that can occur in $[\![\mathcal{T}(P)_{\prec s}]\!]_G$), so assume for contradiction that there is a variable $?x \in \mathsf{dom}(\mu|_s) \setminus \mathsf{dom}(\mu_1)$. Then there is a node in $\mathcal{T}(P_{\mu|_s}) \cap \mathcal{T}(P)_{\prec s}$ that does not occur in $\mathcal{T}(P_{\mu_1}) \cap \mathcal{T}(P)_{\prec s}$. This yields a contradiction with $\mu_1 \in [\![\mathcal{T}(P)_{\prec s}]\!]_G$ analogously to the case of condition 2.

For the backward direction, suppose that μ satisfies conditions 1–3. We show that $\mu \in [\![P]\!]_G$ by induction on the depth of $\mathcal{T}(P_\mu)$, that is, the maximal number of edges between the root and a leaf.

For the basis of the induction, let the depth of $\mathcal{T}(P_\mu)$ be 0, that is, the root v of $\mathcal{T}(P)$ be the only node of $\mathcal{T}(P_\mu)$. We prove the claim by induction on the number n of children of v in $\mathcal{T}(P)$. If $n = 0$, then $P = B$ FILTER R for some basic pattern B and filter constraint R, and the claim follows since μ is a pp-solution to P over G. For the inductive step, suppose the claim holds for all wwd-patterns P' and mappings μ' satisfying 1–3 provided $\mathcal{T}(P'_{\mu'})$ has depth 0 and $n-1$ children in $\mathcal{T}(P')$. Let P and μ be such that $\mathcal{T}(P_\mu)$ has depth 0 and n children in $\mathcal{T}(P)$. Let u be the last child of (the root v of) $\mathcal{T}(P_\mu)$. Then $\mu|_u$ is a pp-solution to $\mathcal{T}(P)_{\prec u}$ that satisfies conditions 2 and 3 since $(\mu|_u)|_w = \mu|_w$ for every $w \prec u$. Hence, by the inductive hypothesis for the pattern corresponding to $\mathcal{T}(P)_{\prec u}$ and the mapping $\mu|_u$, we have $\mu|_u \in [\![\mathcal{T}(P)_{\prec u}]\!]_G$. We distinguish two cases.

- Let u be a special node labelled with R. Then it suffices to show that $\mu|_u \models R$, which is immediate since $\mu|_u$ satisfies condition 3.
- Let u be an ordinary node labelled with (B, R). We know that u is not in $\mathcal{T}(P_\mu)$. Since v is in $\mathcal{T}(P_\mu)$, by condition 2 there is no mapping μ' such that (a) $\mu|_u \sqsubseteq \mu'$, (b) $\mu' \models R$, and (c) $\mu'(B) \subseteq G$. Since R is safe, it follows that every mapping satisfying (b) and (c) is incompatible with $\mu|_u$. Consequently, every mapping in $[\![\mathcal{T}(P)_u]\!]_G$ is incompatible with $\mu|_u$, and hence $\mu = \mu|_u \in [\![\mathcal{T}(P)_{\prec u}]\!]_G \setminus [\![\mathcal{T}(P)_u]\!]_G$, as required.

For the outer inductive step, let the claim hold for all P' and μ' with $\mathcal{T}(P'_{\mu'})$ of depth $d-1$, for some $d > 0$. Once again, we show the claim for P and μ with $\mathcal{T}(P_\mu)$ of depth d by induction on the number n of children of the root v of $\mathcal{T}(P)$. The basis is vacuous as v cannot have 0 children while $\mathcal{T}(P_\mu)$ has positive depth. The inductive step is the same as for depth 0, except that we have an additional case for the last child u of the root v.

- Let u be an ordinary node labelled with (B', R') that is contained in $\mathcal{T}(P_\mu)$. Then $\mu = \mu|_u \cup \mu_2$ where μ_2 is the projection of μ to the set of variables occurring in the subtree $\mathcal{T}$ of $\mathcal{T}(P_\mu)$ rooted at u (i.e., $\mathcal{T} = \mathcal{T}(P_\mu)_u$). Since u is contained in $\mathcal{T}(P_\mu)$ and contains no special children, μ_2 is a pp-solution to (the subpattern represented by) $\mathcal{T}(P)_u$. Moreover, μ_2 satisfies condition 3 with respect to $\mathcal{T}(P)_u$ since $\mathcal{T}(P)_u$ contains no special nodes. We next show that μ_2 satisfies condition 2 with respect to $\mathcal{T}(P)_u$. Let w be a child of $\mathcal{T}$ (in $\mathcal{T}(P)_u$) labelled with (B, R), and assume for contradiction that there is some μ' such that $\mu_2|_w \sqsubseteq \mu'$, $\mu' \models R$, and $\mu'(B) \subseteq G$. Without loss of generality, $\mathsf{dom}(\mu') = \mathsf{dom}(\mu_2) \cup \mathsf{vars}(B)$. Thus, since P is weakly well-designed, $\mathsf{vars}(B) \cap \mathsf{dom}(\mu|_u) \subseteq \mathsf{vars}(B') \subseteq \mathsf{dom}(\mu_2)$. Hence, μ' is compatible with $\mu|_u$, and $\mu|_w \sqsubseteq \mu|_u \cup \mu'$. Moreover, since μ' and $\mu|_u \cup \mu'$ coincide on $\mathsf{vars}(B)$ and R is safe, we have that $\mu|_u \cup \mu' \models R$ and $(\mu|_u \cup \mu')(B) \subseteq G$, contradicting the assumption for μ. Since μ_2 satisfies conditions 1–3 with respect to $\mathcal{T}(P)_u$, by the outer inductive hypothesis we obtain that $\mu_2 \in [\![\mathcal{T}(P)_u]\!]_G$, and hence $\mu \in [\![\mathcal{T}(P)_{\prec u}]\!]_G \bowtie [\![\mathcal{T}(P)_u]\!]_G$ (as $\mu|_u \in [\![\mathcal{T}(P)_{\prec u}]\!]_G$ holds by the inner inductive hypothesis). The claim follows. □

Checking whether a mapping μ satisfies this characterisation is feasible in coNP, and the matching lower bound follows from the coNP-hardness of evaluation of wd-patterns [33].

Theorem 1 *Problem* EVAL$(\mathcal{P}_{\text{wwd}})$ *is* coNP*-complete.*

Proof The lower bound of this statement is known from [33], and the upper bound can be obtained from Lemma 3 as follows.

First we show that testing whether μ is a pp-solution takes polynomial time, same as computing the maximal witnessing tree $\mathcal{T}(P_\mu)$. We just proceed from the root of the tree down along the branches until we cannot find a match $\mu(\mathsf{triples}(v))$ in G for the basic pattern in the child v which satisfies the condition in the node, and then check that the variables in the resulting tree are exactly $\mathsf{vars}(\mu)$. So, the crucial part is to check that $\mathcal{T}(P_\mu)$ is not extendable to any of its children. But there are only linearly many children, and each check can be done in coNP. Finally, the checks for top-level filters are again polynomial. □

Pérez et al. [33] extended wd-patterns to UNION by considering *unions of wd-patterns*, that is, patterns of the form P_1 UNION ... UNION P_n with all $P_i \in \mathcal{P}_{\text{wd}}$. We denote the resulting fragment by $\mathcal{U}_{\text{wd}}$. This syntactic restriction on the use of UNION in $\mathcal{U}_{\text{wd}}$ is motivated by the fact that any pattern in $\mathcal{U}$ can be equivalently expressed as a union of UNION-free patterns [33]. We denote the fragment of all queries over patterns in $\mathcal{U}_{\text{wd}}$ by $\mathcal{S}_{\text{wd}}$. Similarly, we write $\mathcal{U}_{\text{wwd}}$ for unions of wwd-patterns and $\mathcal{S}_{\text{wwd}}$ for queries over unions of wwd-patterns.

Analogously to the well-designed case, Theorem 1 extends to fragments $\mathcal{U}_{\text{wwd}}$ and $\mathcal{S}_{\text{wwd}}$.

Corollary 2 *Problem* EVAL$(\mathcal{U}_{\text{wwd}})$ *is* coNP*-complete, and* EVAL$(\mathcal{S}_{\text{wwd}})$ *is* Σ_2^p*-complete.*

The coNP-algorithm for $\mathcal{U}_{\text{wwd}}$ is obtained simply by applying the algorithm for $\mathcal{P}_{\text{wwd}}$ to each pattern in the union. Hardness for $\mathcal{S}_{\text{wwd}}$ follows from the hardness of the well-designed case [27], while for membership we just guess the values of the existential variables and then call a coNP-oracle for $\mathcal{U}_{\text{wwd}}$ on the resulting mapping and the normalised body of the query.

Hence, the complexity of evaluation for wwd-patterns is the same as for wd-patterns. We next show that wwd-patterns are, in a certain sense, a maximal extension of wd-patterns that preserves coNP evaluation complexity (under the usual complexity-theoretic assumptions).

The definition of weakly well-designed patterns suggests two intuitive ways in which it could be relaxed. Given an occurrence i of an OPT-subpattern P_1 OPT P_2, one could allow variables in $\mathsf{vars}(P_2) \setminus \mathsf{vars}(P_1)$ to occur in

- some subpatterns whose occurrences are not dominated by i, or
- constraints of some non-top-level occurrences of FILTER-patterns.

We next show that either relaxation immediately makes the evaluation problem Π_2^p-hard.

For the first relaxation, the arguably simplest special case would be to allow for some non-well-designed OPT-nesting of type (Opt-R). Consider the fragment $\mathcal{P}_{\text{opt-r}}$ of patterns of the form B_1 OPT $(B_2$ OPT $B_3)$, where B_1, B_2 and B_3 are basic patterns. Intuitively, $\mathcal{P}_{\text{opt-r}}$ allows for the most simple form of non-well-designed nesting of type (Opt-R).

Theorem 2 *Problem* EVAL($\mathcal{P}_{\text{opt-r}}$) *is* Π_2^p*-complete.*

Proof This theorem is a corollary of [38, Theorem 4] for their class $\mathcal{E}_{\leq 3}$, but without UNION. □

Now suppose we allow for some non-well-designed non-top-level filters, as suggested by the second relaxation. As we will see next, even a very restricted fragment of patterns allowing for such filters is Π_2^p-complete. This implies that the requirement that special nodes be children of the root, while it may look somewhat ad-hoc, cannot be substantially relaxed. Consider the fragment $\mathcal{P}_{\text{filter-2}}$ of patterns of the form

$$B_1 \text{ OPT } ((B_2 \text{ OPT } B_3) \text{ FILTER } R),$$

where B_1, B_2 and B_3 are basic patterns such that $\mathsf{vars}(B_3) \cap \mathsf{vars}(B_1) \subseteq \mathsf{vars}(B_2)$, and R is a filter constraint. Intuitively, $\mathcal{P}_{\text{filter-2}}$ allows for the simplest form of "second-level" filters.

Theorem 3 *Problem* EVAL*($\mathcal{P}_{\text{filter-2}}$) is* Π_2^p*-complete.*

Proof This problem allows for a reduction from a restriction of EVAL($\mathcal{P}_{\text{opt-r}}$). Indeed, from the proof of [38, Theorem 4] it follows that it is already Π_2^p-hard to check whether $\mu \in [\![P]\!]_G$ for P of the form B_1 OPT $(B_2$ OPT $B_3)$ with $\mathsf{dom}(\mu) = \mathsf{vars}(B_1)$ and $\mathsf{vars}(B_2) \setminus \mathsf{vars}(B_1) \neq \emptyset$. Let P and μ be such a pattern and such a mapping, respectively. Consider the pattern

$$\begin{aligned} P' &= B_1 \text{ OPT } (((B_2 \cup B_1) \text{ OPT } B_3')\text{FILTER } R), \text{ where} \\ R &= \neg bound(?x_1') \vee ((?x_1' = ?x_1) \wedge \cdots \wedge (?x_n' = ?x_n)), \end{aligned}$$

with B_3' a basic pattern obtained from B_3 by replacing all the variables $?x_1, \ldots, ?x_n$ in $(\mathsf{vars}(B_3) \cap \mathsf{vars}(B_1)) \setminus \mathsf{vars}(B_2)$ by their fresh copies $?x_1', \ldots, ?x_n'$ (if no such variables exist, that is, if the original pattern is well-designed, we just set R to $\top$). Clearly, $P' \in \mathcal{P}_{\text{filter-2}}$, so it suffices to show, for every G and μ with $\mathsf{dom}(\mu) = \mathsf{vars}(B_1)$, that $\mu \in [\![P]\!]_G$ if and only if $\mu \in [\![P']\!]_G$.

For the forward direction, suppose $\mathsf{dom}(\mu) = \mathsf{vars}(B_1)$ and $\mu \in [\![P]\!]_G$. Since $\mathsf{vars}(B_2) \setminus \mathsf{vars}(B_1) \neq \emptyset$, we must have $\mu \in [\![B_1]\!]_G \setminus [\![B_2 \text{ OPT } B_3]\!]_G$. Thus, $\mu \in [\![B_1]\!]_G$ and for every $\mu' \in [\![B_2 \text{ OPT } B_3]\!]_G$ we have $\mu \not\sim \mu'$. Since $\mu \in [\![B_1]\!]_G$, to show $\mu \in [\![P']\!]_G$ it suffices to verify that μ is not compatible with any

$\mu' \in [\![((B_2 \cup B_1) \text{ OPT } B_3') \text{ FILTER } R]\!]_G$, for which we distinguish the following two cases.

- If $\mu' \in [\![B_2 \cup B_1]\!]_G \bowtie [\![B_3']\!]_G$, then $\mu' \models (?x_1' = ?x_1) \wedge \cdots \wedge (?x_n' = ?x_n)$. Hence, $\mu'|_{\mathsf{vars}(B_2) \cup \mathsf{vars}(B_3)} \in [\![B_2 \text{ OPT } B_3]\!]_G$. Consequently, by assumption, $\mu \not\sim \mu'|_{\mathsf{vars}(B_2) \cup \mathsf{vars}(B_3)}$, and thus $\mu \not\sim \mu'$.
- If $\mu' \in [\![B_2 \cup B_1]\!]_G \setminus [\![B_3']\!]_G$, then $\mu'|_{\mathsf{vars}(B_2)} \in [\![B_2]\!]_G \setminus [\![B_3']\!]_G = [\![B_2]\!]_G \setminus [\![B_3]\!]_G \subseteq [\![B_2 \text{ OPT } B_3]\!]_G$. Therefore, by assumption, $\mu \not\sim \mu'|_{\mathsf{vars}(B_2)}$, and hence $\mu \not\sim \mu'$.

For the backward direction, suppose $\mathsf{dom}(\mu) = \mathsf{vars}(B_1)$ and $\mu \in [\![P']\!]_G$. Again, since $\mathsf{vars}(B_2) \setminus \mathsf{vars}(B_1) \neq \emptyset$, we have $\mu \in [\![B_1]\!]_G \setminus [\![((B_2 \cup B_1) \text{ OPT } B_3') \text{ FILTER } R]\!]_G$. Thus, $\mu \in [\![B_1]\!]_G$ and $\mu \not\sim \mu'$ for every $\mu' \in [\![((B_2 \cup B_1) \text{ OPT } B_3') \text{ FILTER } R]\!]_G$. Since $\mu \in [\![B_1]\!]_G$, it follows that $[\![((B_2 \cup B_1) \text{ OPT } B_3') \text{ FILTER } R]\!]_G = \emptyset$. To show $\mu \in [\![P]\!]_G$ it suffices to verify that μ is not compatible with any $\mu' \in [\![B_2 \text{ OPT } B_3]\!]_G$. Assume for the sake of contradiction that this is not the case and there is a compatible μ'. We distinguish the following two cases.

- Suppose $\mu' \in [\![B_2]\!]_G \bowtie [\![B_3]\!]_G$. Then there is some μ'' such that $\mu \cup \mu' \cup \mu'' \in [\![B_1]\!]_G \bowtie [\![B_2]\!]_G \bowtie [\![B_3']\!]_G$ and $\mu \cup \mu' \cup \mu'' \models (?x_1' = ?x_1) \wedge \cdots \wedge (?x_n' = ?x_n)$. Thus, $\mu \cup \mu' \cup \mu'' \in [\![((B_2 \cup B_1) \text{ OPT } B_3') \text{ FILTER } R]\!]_G$, which is a contradiction.
- Suppose $\mu' \in [\![B_2]\!]_G \setminus [\![B_3]\!]_G = [\![B_2]\!]_G \setminus [\![B_3']\!]_G$. Then $\mu \cup \mu' \in [\![B_1]\!]_G \cup B_2 \setminus [\![B_3']\!]_G$ and $\mu \cup \mu' \models \neg \mathit{bound}(?x_1')$, and hence $\mu \cup \mu' \in [\![((B_2 \cup B_1) \text{ OPT } B_3') \text{ FILTER } R]\!]_G$, which is again a contradiction. □

Theorems 2 and 3 suggest that $\mathcal{P}_{\text{wwd}}$ is a maximal fragment of $\mathcal{P}$ that does not impose structural restrictions on basic patterns or filter constraints and has a CONP evaluation algorithm (assuming CONP $\neq \Pi_2^p$). Hence, going beyond wwd-patterns while preserving good computational properties requires more refined restrictions, possibly in the spirit of [27, Section 4].

6 Expressivity of wwd-Patterns and Their Extensions

In this section, we analyse the expressive power of our fragments.

Definition 9 A language $\mathcal{L}_1$ is *strictly less expressive* than a language $\mathcal{L}_2$ (written $\mathcal{L}_1 < \mathcal{L}_2$) if for every query Q_1 in $\mathcal{L}_1$ there is a query Q_2 in $\mathcal{L}_2$ such that $Q_1 \equiv Q_2$, and there is a query Q_2 in $\mathcal{L}_2$ such that $Q_1 \not\equiv Q_2$ for every query Q_1 in $\mathcal{L}_1$.

We begin with UNION-free patterns, establishing that $\mathcal{P}_{\text{wd}} < \mathcal{P}_{\text{wwd}} < \mathcal{P}$, and then proceed to unions, showing that $\mathcal{U}_{\text{wd}} < \mathcal{U}_{\text{wwd}} < \mathcal{U}$, and queries, showing that $\mathcal{S}_{\text{wd}} < \mathcal{S}_{\text{wwd}} < \mathcal{S}$.

Following [5, 33], a set of mappings Ω_1 is *subsumed* by a set of mappings Ω_2 (written $\Omega_1 \sqsubseteq \Omega_2$) if for every $\mu_1 \in \Omega_1$ there exists a mapping $\mu_2 \in \Omega_2$ such that $\mu_1 \sqsubseteq \mu_2$. A query Q is *weakly monotone* if $[\![Q]\!]_{G_1} \sqsubseteq [\![Q]\!]_{G_2}$ for any two graphs G_1 and G_2 with $G_1 \subseteq G_2$, and a fragment $\mathcal{L}$ is *weakly monotone* if it contains only weakly monotone queries. Arenas and Pérez [5] showed that, unlike $\mathcal{P}$, the fragment $\mathcal{P}_{\text{wd}}$ is weakly monotone, and hence $\mathcal{P}_{\text{wd}} < \mathcal{P}$.

Example 4 (Pérez et al. [33]) Consider the non-well-designed pattern

$$P = (?x, a, 1) \text{ OPT } ((?y, a, 2) \text{ OPT } (?x, a, 3))$$

as well as graphs $G_1 = \{(1, a, 1), (2, a, 2)\}$ and $G_2 = G_1 \cup \{(3, a, 3)\}$. Then $\mu_1 = \{?x \mapsto 1, ?y \mapsto 2\}$ is the only mapping in $[\![P]\!]_{G_1}$ while $\mu_2 = \{?x \mapsto 1\}$ is the only mapping in $[\![P]\!]_{G_2}$. Hence $[\![P]\!]_{G_1} \not\sqsubseteq [\![P]\!]_{G_2}$, meaning that P is not weakly monotone.

Analogously, we show that $\mathcal{P}_{\text{wd}} < \mathcal{P}_{\text{wwd}}$ by observing that $\mathcal{P}_{\text{wwd}}$ is not weakly monotone.

Proposition 7 *Fragment $\mathcal{P}_{\text{wwd}}$ is not weakly monotone.*

Proof Consider a wwd-pattern

$$P = ((?x, a, 1) \text{ OPT } (?y, a, 2)) \text{ OPT } (?y, a, 3),$$

as well as graphs $G_1 = \{(1, a, 1), (3, a, 3)\}$ and $G_2 = G_1 \cup \{(2, a, 2)\}$. Then $[\![P]\!]_{G_1} = \{\{?x \mapsto 1, ?y \mapsto 3\}\} \not\sqsubseteq \{\{?x \mapsto 1, ?y \mapsto 2\}\} = [\![P]\!]_{G_2}$. □

An alternative proof of $\mathcal{P}_{\text{wd}} < \mathcal{P}_{\text{wwd}}$ can be obtained by adapting Theorem 3.5 in [6], which exhibits a weakly well-designed, weakly monotone pattern that is not equivalent to any well-designed pattern.

To distinguish $\mathcal{P}_{\text{wwd}}$ from $\mathcal{P}$ we need a different property.

Definition 10 A query Q is *non-reducing* if for any two graphs G_1, G_2 such that $G_1 \subseteq G_2$ and any mapping $\mu_1 \in [\![Q]\!]_{G_1}$ there is no $\mu_2 \in [\![Q]\!]_{G_2}$ such that $\mu_2 \sqsubset \mu_1$ (i.e., $\mu_2 \sqsubseteq \mu_1$ and $\mu_2 \neq \mu_1$). A fragment is *non-reducing* if it contains only non-reducing queries.

Intuitively, for a non-reducing query extending a graph cannot result in a previously bound answer variable becoming unbound. All weakly monotone queries are non-reducing but not vice versa. Moreover, all wwd-patterns are non-reducing.

Proposition 8 *Fragment $\mathcal{P}_{\text{wwd}}$ is non-reducing.*

Proof Let $P \in \mathcal{P}_{\text{wwd}}$ and let G_1, G_2 be two graphs such that $G_1 \subseteq G_2$. We show that $\mu_2 \not\sqsubset \mu_1$ for any $\mu_1 \in [\![P]\!]_{G_1}$ and $\mu_2 \in [\![P]\!]_{G_2}$ by induction on the structure of P, proving, in parallel, that if all filters in P are over basic patterns, then for every mapping $\mu_1 \in [\![P]\!]_{G_1}$ there is a mapping $\mu_2 \in [\![P]\!]_{G_2}$ such that $\mu_1|_{\mathsf{vars}(v)} = \mu_2|_{\mathsf{vars}(v)}$ for v the root of $\mathcal{T}(P)$.

For the base case, suppose $P = B \text{ FILTER } R$ for some basic pattern B and filter constraint R. Then, P is *monotone* in the sense of [5], that is, satisfies $[\![P]\!]_{G_1} \subseteq [\![P]\!]_{G_2}$. Moreover, P contains no OPT, and hence every two distinct mappings in $[\![P]\!]_{G_2}$ have the same domain and are thus incompatible. These facts imply both claims.

For the inductive step, suppose first that $P = P_1$ OPT P_2 and both claims hold for P_1 and P_2. Let $\mu_1 \in [\![P]\!]_{G_1}$. We first prove that $\mu_2 \not\sqsubset \mu_1$ for any $\mu_2 \in [\![P]\!]_{G_2}$. We distinguish two cases.

– Let $\mu_1 = \mu_1^1 \cup \mu_1^2$ where $\mu_1^i \in [\![P_i]\!]_{G_1}$. Assume for contradiction that $\mu_2 \sqsubset \mu_1$ for some $\mu_2 \in [\![P]\!]_{G_2}$. We begin by showing that μ_2 must be of the form $\mu_2^1 \cup \mu_2^2$ where $\mu_2^i \in [\![P_i]\!]_{G_2}$, for which it suffices to show that $\mu_2|_{\mathsf{vars}(P_1)}$ is compatible with some mapping in $[\![P_2]\!]_{G_2}$. On the one hand, since $\mu_2 \sqsubset \mu_1$, μ_1^2 is compatible with $\mu_2|_{\mathsf{vars}(P_1)}$. On the other hand, since all filters in P_2 are over basic patterns, the inductive hypothesis tells us that $[\![P_2]\!]_{G_2}$ contains a mapping μ' that coincides with μ_1^2 on the set of variables X in the root of $\mathcal{T}(P_2)$; moreover, since P is weakly well-designed, $\mathsf{dom}(\mu') \cap \mathsf{vars}(P_1) \subseteq X$, and hence μ' is compatible with $\mu_2|_{\mathsf{vars}(P_1)}$. Thus, $\mu_2 = \mu_2^1 \cup \mu_2^2$ where $\mu_2^i \in [\![P_i]\!]_{G_2}$. Then, however, we must have that $\mu_2^1 \sqsubset \mu_1^1$ or $\mu_2^2 \sqsubset \mu_1^2$, contradicting the inductive hypothesis for P_1 or P_2, respectively.
– Let $\mu_1 \in [\![P_1]\!]_{G_1} \setminus [\![P_2]\!]_{G_1}$, and let μ_2 be an arbitrary mapping in $[\![P]\!]_{G_2}$. Then μ_2 extends some $\mu' \in [\![P_1]\!]_{G_2}$. By the inductive hypothesis for claim 2, we have that $\mu' \not\sqsubset \mu_1$, and hence $\mu_2 \not\sqsubset \mu_1$.

Suppose now that all filters in P are over basic patterns. We need to prove that there is $\mu_2 \in [\![P]\!]_{G_2}$ such that $\mu_1|_{\mathsf{vars}(v)} = \mu_2|_{\mathsf{vars}(v)}$. We know that μ_1 extends some $\mu_1' \in [\![P_1]\!]_{G_1}$. Thus, by the inductive hypothesis, there is some $\mu_2' \in [\![P_1]\!]_{G_2}$ that coincides with μ_1' on the variables in the root of $\mathcal{T}(P_1)$. The claim follows since μ_2' can be extended to a mapping μ_2 for P that coincides with μ_2' on the variables in the root of $\mathcal{T}(P_1)$, and, by construction, the root of $\mathcal{T}(P_1)$ and the root of $\mathcal{T}(P)$ have the same label.

Consider now the inductive step for the case when $P = P_1$ FILTER R. Since P_1 is not a basic pattern, we only need to show that $\mu_2 \not\sqsubset \mu_1$ for any $\mu_1 \in [\![P]\!]_{G_1}$ and $\mu_2 \in [\![P]\!]_{G_2}$. This holds by the inductive hypothesis, because $\mu_1 \in [\![P_1]\!]_{G_1}$ and $\mu_2 \in [\![P_1]\!]_{G_2}$ for any such μ_1 and μ_2. □

In contrast to Proposition 8, patterns in $\mathcal{P}$ do not generally satisfy non-reducibility. For instance, consider again pattern P, graphs G_1, G_2, and mappings μ_1, μ_2 from Example 4. Pattern P is not non-reducing since $\mu_1 \in [\![P]\!]_{G_1}$ and $\mu_2 \in [\![P]\!]_{G_2}$ but $\mu_2 \sqsubset \mu_1$. Therefore, we have the following theorem.

Theorem 4 *It holds that* $\mathcal{P}_{\text{wd}} < \mathcal{P}_{\text{wwd}} < \mathcal{P}$.

We next compare $\mathcal{U}_{\text{wwd}}$ to $\mathcal{U}_{\text{wd}}$ and $\mathcal{U}$, as well as $\mathcal{S}_{\text{wwd}}$ to $\mathcal{S}_{\text{wd}}$ and $\mathcal{S}$ (note that neither UNION nor projection via SELECT can be expressed by means of the other operators [40], so adding either construct makes each fragment strictly more expressive). It is easily seen that $\mathcal{U}_{\text{wd}}$ and $\mathcal{S}_{\text{wd}}$ inherit weak monotonicity from $\mathcal{P}_{\text{wd}}$ [27, 33], and hence $\mathcal{U}_{\text{wd}} < \mathcal{U}_{\text{wwd}}$ and $\mathcal{S}_{\text{wd}} < \mathcal{S}_{\text{wwd}}$. Non-reducibility, however, does not propagate to unions.

Example 5 Consider the following $\mathcal{U}_{\text{wd}}$-pattern with G_1, G_2 and μ_1, μ_2 from Example 4:

$$P = ((?x, a, 1) \text{ OPT } (?y, a, 2)) \text{ UNION } (?x, a, 1).$$

We have $\mu_1 \in [\![P]\!]_{G_1}$ and $\mu_2 \in [\![P]\!]_{G_2}$ but $\mu_2 \sqsubset \mu_1$, which is due to the fact that μ_2 is already contained in $[\![P]\!]_{G_1}$ along with μ_1. This is only possible in the presence of UNION since all mappings in the evaluation of a UNION-free pattern are mutually non-subsuming (see Lemma 2).

Thus, to account for UNION, we introduce the following, more delicate property.

Definition 11 A query Q is *extension-witnessing* (*e-witnessing*) if for any two graphs $G_1 \subseteq G_2$ and mapping $\mu \in [\![Q]\!]_{G_2}$ such that $\mu \notin [\![Q]\!]_{G_1}$ there is a triple t in Q such that $\mathsf{vars}(t) \subseteq \mathsf{dom}(\mu)$ and $\mu(t) \in G_2 \setminus G_1$. A fragment is *e-witnessing* if so are all of its queries.

Informally, a query Q is e-witnessing if whenever an extension of a graph leads to a new answer, this answer is justified by a triple pattern in Q which maps to the extension. Unions of wwd-patterns can be shown e-witnessing.

Proposition 9 *Fragment $\mathcal{U}_{\text{wwd}}$ is e-witnessing.*

Proof Let $P \in \mathcal{U}_{\text{wwd}}$ and let G_1, G_2 be graphs such that $G_1 \subseteq G_2$. Let μ be a mapping in $[\![P]\!]_{G_2}$ but not in $[\![P]\!]_{G_1}$. We show that there is some $t \in \mathsf{triples}(P)$ such that $\mu(t) \in G_2 \setminus G_1$.

Since P is a union of wwd-patterns, there is some wwd-pattern P' in the union such that $\mu \in [\![P']\!]_{G_2}$. It suffices to show $\mu(\mathsf{triples}(P'_\mu)) \cap (G_2 \setminus G_1) \neq \emptyset$, where P'_μ is the pattern corresponding to the maximal r-subtree of P witnessing μ in G_2 (i.e., the part of P in the image of μ, see Definition 8). We know that $\mu(\mathsf{triples}(P'_\mu)) \subseteq G_2$. Assume, for contradiction, that $\mu(\mathsf{triples}(P'_\mu)) \subseteq G_1$. Then μ is a pp-solution to P' over G_1. We next show that μ is a real solution to P' over G_1. By Lemma 3, it suffices to show that (a) for any child u of $\mathcal{T}(P'_\mu)$ labelled with (B, R), there is no mapping μ' such that $\mu|_u \sqsubseteq \mu'$, $\mu' \models R$, and $\mu'(B) \subseteq G_1$, and (b) $\mu|_s \models R$ for any special node s in $\mathcal{T}(P')$ labelled with R. Claim (a) holds since $\mu \in [\![P']\!]_{G_2}$ and $G_1 \subseteq G_2$ while (b) holds since $\mu \in [\![P']\!]_{G_2}$ and the claim does not depend on the graph over which the evaluation is computed. Consequently, $\mu \in [\![P']\!]_{G_1}$, and hence $\mu \in [\![P]\!]_{G_1}$, in contradiction to the assumption. □

On the other hand, $\mathcal{U}$ is not e-witnessing, as can be seen on the pattern and graphs in Example 4. Hence, we obtain the following theorem.

Theorem 5 *It holds that $\mathcal{U}_{\text{wd}} < \mathcal{U}_{\text{wwd}} < \mathcal{U}$.*

Next we move to the fragments that allow for projection. As already mentioned, we have $\mathcal{S}_{\text{wd}} < \mathcal{S}_{\text{wwd}}$ since $\mathcal{S}_{\text{wd}}$ is weakly monotone while $\mathcal{S}_{\text{wwd}}$ is not. However, $\mathcal{S}_{\text{wwd}}$ is not e-witnessing, so we cannot apply the technique of Theorem 5 to establish $\mathcal{S}_{\text{wwd}} < \mathcal{S}$; instead, we make use of the following lemma.

Lemma 4 *Let Q be a query in $\mathcal{S}_{\text{wwd}}$ and G be a graph. For every graph G_1 with $G \subseteq G_1$ and every $\mu \in [\![Q]\!]_{G_1}$, there is a graph G_2 with $G \subseteq G_2$ such that $\mu \in [\![Q]\!]_{G_2}$ and $|G_2| \leq |G| + |\mathsf{triples}(Q)|$.*

Proof Let $Q = \mathsf{SELECT}\ X\ \mathsf{WHERE}\ P$, for P a union of wwd-patterns, and let G, G_1 and μ be as required. Then there is a wwd-pattern P' in the union P such that $\mu' \in [\![P']\!]_{G_1}$ for some μ' with $\mu'|_X = \mu$. Let $G_2 = G \cup \mu'(\mathsf{triples}(P'_{\mu'}))$. Clearly, $|G_2| \leq |G| + |\mathsf{triples}(Q)|$, so it suffices to show that $\mu' \in [\![P']\!]_{G_2}$.

By construction, μ' is a pp-solution to P' over G_2. Moreover, since μ' is a solution to P' over G_1, we have that $\mu|_s \models R$ for every special node s in $\mathcal{T}(P')$ labelled with R. Finally, suppose for contradiction that there is a child v of $\mathcal{T}(P'_{\mu'})$ labelled with (B, R) and a mapping μ'' such that $\mu'|_v \sqsubseteq \mu''$, $\mu'' \models R$, and $\mu''(B) \subseteq G_2$. However, since $G_2 \subseteq G_1$, we then have $\mu''(B) \subseteq G_1$, which contradicts the fact that $\mu' \in [\![P']\!]_{G_1}$. □

This lemma is the base of the last result of the section.

Theorem 6 *It holds that* $\mathcal{S}_{\text{wd}} < \mathcal{S}_{\text{wwd}} < \mathcal{S}$.

Proof As observed before, the inclusion $\mathcal{S}_{\text{wd}} < \mathcal{S}_{\text{wwd}}$ holds since $\mathcal{S}_{\text{wd}}$ is weakly monotone [27, 33] and $\mathcal{S}_{\text{wwd}}$ is not.

As for the second inclusion, consider the family of graphs

$$G_n = \{(a, a, a), (d_1, b, b), \ldots, (d_n, b, b), (d_1, c, c)\},$$

for pairwise distinct IRIs $a, b, c, d_1, \ldots, d_n$, and the query

$$Q = \mathsf{SELECT}\ \{?x, ?y\}\ \mathsf{WHERE}\ (?x, a, a)\ \mathsf{DIFF}\ ((?y, b, b)\ \mathsf{DIFF}\ (?y, c, c)).$$

By equivalence (5) in Section 3, the operator DIFF can be expressed via OPT, AND and FILTER, so we can assume that $Q \in \mathcal{S}$. On the other hand, it is easily seen that $Q \notin \mathcal{S}_{\text{wwd}}$. The mapping $\mu = \{?x \mapsto a\}$ is an answer to Q over G_1 but not an answer over any G_n with $n \geq 2$. Moreover, it is easily seen that any extension G of G_n such that $\mu \in [\![Q]\!]_G$ requires the addition of at least $n - 1$ triples, namely $\{(d_2, c, c), \ldots, (d_n, c, c)\}$. Consequently, $\mu \in [\![Q]\!]_G$ implies $|G| \geq |G_1| + n - 1$.

Suppose for contradiction there is a query Q' in $\mathcal{S}_{\text{wwd}}$ such that $Q' \equiv Q$. Let $n = |\mathsf{triples}(Q')| + 2$. Then, by Lemma 4, $\mu \in [\![Q']\!]_G$ for some G with $|G| \leq |G_n| + |\mathsf{triples}(Q')| = |G_n| + n - 2$, which contradicts the above observation for Q. □

7 Static Analysis of wwd-Patterns

In this section, we look at the general static analysis problems of query equivalence and containment. Formally, equivalence for a language $\mathcal{L}$ is defined as follows.

EQUIVALENCE($\mathcal{L}$)	
Input:	Queries Q and Q' in $\mathcal{L}$
Question:	Is $Q \equiv Q'$?

This problem is commonly generalised to CONTAINMENT($\mathcal{L}$), in which one checks whether Q is *contained* in Q', written $Q \subseteq Q'$, that is, whether $[\![Q]\!]_G \subseteq$

$[\![Q']\!]_G$ holds for every graph G. We have $Q \equiv Q'$ if and only if Q and Q' contain each other.

These problems have been studied for FILTER-free wd-patterns in [27, 35], establishing NP-completeness of equivalence and containment. Moreover, both problems are Π_2^p-complete for unions of FILTER-free wd-patterns, and undecidable for fragments with projection. Finally, from the results in [41] it follows that containment is undecidable for $\mathcal{U}$. On the other hand, nothing seems to be known so far for well-designed patterns with FILTER.

We next show that equivalence and containment are both Π_2^p-complete for $\mathcal{P}_{\text{wwd}}$ and $\mathcal{U}_{\text{wwd}}$ (whereas they are undecidable for $\mathcal{S}_{\text{wwd}}$ by the results in [35]). As the following lemma shows, the upper bound for containment follows from a small counterexample property: if $P \not\subseteq P'$ for some P and P' from $\mathcal{U}_{\text{wwd}}$, then there is a witnessing mapping and graph of size $O(|P| + |P'|)$. Given this property, a Π_2^p algorithm for containment is straightforward—we guess a mapping μ and a graph G of linear size, check that $\mu \notin [\![P']\!]_G$, and then call a CONP oracle for checking $\mu \in [\![P]\!]_G$. As a corollary, EQUIVALENCE($\mathcal{U}_{\text{wwd}}$) is also in Π_2^p.

Lemma 5 *Let P and P' be two patterns from $\mathcal{U}_{\text{wwd}}$. If $P \not\subseteq P'$ then there exists a mapping μ and a graph G of size $O(|\mathsf{triples}(P)| + |\mathsf{triples}(P')|)$ such that $\mu \in [\![P]\!]_G$ but $\mu \notin [\![P']\!]_G$.*

Proof Without loss of generality, let us assume that

$$\begin{aligned} P &= P_1 \text{ UNION } P_2 \text{ UNION } \ldots \text{ UNION } P_n,\\ P' &= P'_1 \text{ UNION } P'_2 \text{ UNION } \ldots \text{ UNION } P'_m,\end{aligned}$$

where all P_i and P'_j are wwd-patterns in OF-normal form.

Since $P \not\subseteq P'$, there exists a graph G', a mapping μ, and a pattern P_i, $1 \leq i \leq n$, such that $\mu \in [\![P_i]\!]_{G'}$, but $\mu \notin [\![P'_j]\!]_{G'}$ for every P'_j. Hence, μ is a pp-solution to P_i over G' with corresponding r-subtree $\mathcal{T}((P_i)_\mu)$ of the CPT $\mathcal{T}(P_i)$. Let $G_0 = \mu(\mathsf{triples}((P_i)_\mu))$. By construction, we have that $G_0 \subseteq G'$ and $|G_0| \leq |\mathsf{triples}((P_i)_\mu)| \leq |\mathsf{triples}(P_i)| \leq |\mathsf{triples}(P)|$. Moreover, $\mu \in [\![P_i]\!]_{G_0}$, because all the matches and constraints, including the ones on the top-level, stay unchanged. In fact, $\mu \in [\![P_i]\!]_{G''}$ for any G'' such that $G_0 \subseteq G'' \subseteq G'$.

If $\mu \notin [\![P'_j]\!]_{G_0}$ for every j, then G_0 satisfies all the properties required from G. Otherwise, there exists P'_j among $P'_1, \ldots, P'_m$ such that $\mu \in [\![P'_j]\!]_{G_0}$. Since $G_0 \subseteq G'$, μ is a pp-solution to P'_j over G'. Consider the corresponding pattern $(P'_j)_\mu$ (i.e., the maximal pattern witnessing μ in G' obtained from P'_j by dropping the right arguments of some OPT operators), the r-subtree $\mathcal{T}((P'_j)_\mu)$ of the CPT $\mathcal{T}(P'_j)$, and the “image” $\mu(\mathsf{triples}((P'_j)_\mu))$. Note that we may have $\mu(\mathsf{triples}((P'_j)_\mu)) \subseteq G_0$ or not: the latter is possible because the maximal r-subtree of $\mathcal{T}(P'_j)$ witnessing μ in G_0 may be different from $\mathcal{T}((P'_j)_\mu)$, which is maximal in G'. Let $G'_1 = G_0 \cup \mu(\mathsf{triples}((P'_j)_\mu))$. We define G_1 depending on whether $\mu \in [\![P'_j]\!]_{G'_1}$ or not. If $\mu \notin [\![P'_j]\!]_{G'_1}$, then let $G_1 = G'_1$. Otherwise, since $\mu \notin [\![P'_j]\!]_{G'}$ by assumption, there exists a child v of $\mathcal{T}((P'_j)_\mu)$ and a mapping μ_0 such that $\mu|_v \sqsubset \mu_0$

and $\mu_0(\mathsf{triples}(v)) \subseteq G'$. Then the graph $G_1 = G'_1 \cup \mu_0(\mathsf{triples}(v))$ is such that $\mu \notin [\![P'_j]\!]_{G_1}$. In either case, $\mu \in [\![P_i]\!]_{G_1}$ because $G_0 \subseteq G_1 \subseteq G'$. Moreover, we have $\mu \notin [\![P'_j]\!]_{G''}$ for every G'' such that $G_1 \subseteq G'' \subseteq G'$. To see this, suppose for contradiction that $\mu \in [\![P'_j]\!]_{G''}$ for a graph G'' as above. Then there must be a child v' of $\mathcal{T}((P'_j)_{\mu_0})$ such that $v' \prec v$, $\mu_o|_{v'} \sqsubset \mu$ and $\mu(\mathsf{triples}(v')) \subseteq G''$. Since $\mathcal{T}((P'_j)_{\mu_0})$ and $\mathcal{T}((P'_j)_{\mu})$ are identical restricted to nodes preceding v with respect to $\prec$, v' is a child of $\mathcal{T}((P'_j)_{\mu})$. Thus, v' is not contained in $\mathcal{T}((P'_j)_{\mu})$, which contradicts maximality of $\mathcal{T}((P'_j)_{\mu})$ since $\mu(\mathsf{triples}(v')) \subseteq G'' \subseteq G'$.

If $\mu \notin [\![P'_j]\!]_{G_1}$ for all other j as well, then G_1 satisfies all the properties required from G. Otherwise we can extend G_1 to a graph G_2 on the base of some other P'_j with $\mu \in [\![P'_j]\!]_{G_1}$ in the same way as G_1 extends G_0. We then have $G_2 \subseteq G'$, $\mu \in [\![P]\!]_{G_2}$, and $\mu \notin [\![P'_j]\!]_{G_2}$ for j from both steps. Repeating the extension step until there are no P'_j having μ as a solution on the resulting graph, we obtain a graph that satisfies all the properties required from G; in particular, for each j the number of added triples to the graph is bounded by $|\mathsf{triples}(P'_j)|$. □

Hardness of equivalence is established in the following lemma by a reduction of ∀∃3SAT, while containment is Π_2^p-hard by the results in [35]. Note that both results hold even for fragments without FILTER.

Lemma 6 *Problem* EQUIVALENCE*($\mathcal{O}_{\text{wwd}}$) is Π_2^p-hard for the fragment $\mathcal{O}_{\text{wwd}}$ of* FILTER*-free wwd-patterns.*

Proof We proceed by reduction of the ∀∃3SAT problem, that is, the problem of checking whether a formula of the form

$$\forall \bar{x}\, \exists \bar{y}\, \psi, \tag{14}$$

holds for a conjunction ψ of clauses $t_1 \vee t_2 \vee t_3$ with t_i propositional literals, that is, propositional variables from $\bar{x} \cup \bar{y}$ or their negations. Without loss of generality, we assume that ψ contains no tautologous clauses and no clauses with duplicate literals. Let ϕ be a formula of the form (14). Starting from ϕ, we construct FILTER-free wwd-patterns P and P' in OF-normal form, and then show that ϕ is true if and only if $P \equiv P'$. Let $\bar{x} = x_1, \ldots, x_n$ and $\bar{y} = y_1, \ldots, y_m$.

For each clause $\gamma = t_1 \vee t_2 \vee t_3$, there are exactly 7 assignments to the variables in t_1, t_2, t_3 making γ true, and exactly one assignment making γ false (since γ is assumed to be non-tautologous and contain no duplicate literals). Let, for each such γ in ψ, each ℓ, $1 \leq \ell \leq 7$, and each j, $1 \leq j \leq 3$, $val(\gamma, \ell, j) = \top$ if the variable of literal t_j evaluates to true in the ℓ'th assignment making γ true, and $val(\gamma, \ell, j) = \bot$, otherwise; here $\top$ and $\bot$ are fresh IRIs. Let also, for every clause γ in ψ, $cl^1_\gamma, \ldots, cl^7_\gamma$ and $lit^1_\gamma, lit^2_\gamma, lit^3_\gamma$ be fresh IRIs. We define, for each γ and $1 \leq \ell \leq 7$, a basic pattern

$$B^\ell_\gamma = \{(cl^\ell_\gamma, lit^1_\gamma, val(\gamma, \ell, 1)), (cl^\ell_\gamma, lit^2_\gamma, val(\gamma, \ell, 2)), (cl^\ell_\gamma, lit^3_\gamma, val(\gamma, \ell, 3))\},$$

and a basic pattern

$$B^*_\gamma = B^1_\gamma \cup \cdots \cup B^7_\gamma$$

(note that these patterns do not have any variables).

Let, for each propositional variable $z \in \bar{x} \cup \bar{y}$, iri_z be a fresh IRI and $?z$ be a fresh SPARQL variable. For each γ, let also $?c_\gamma$, $?v_\gamma^1$, $?v_\gamma^2$, $?v_\gamma^3$ be fresh variables. Let

$$\begin{aligned} B_\gamma = \{&(?c_\gamma, lit_\gamma^1, ?v_\gamma^1), (?c_\gamma, lit_\gamma^2, ?v_\gamma^2), (?c_\gamma, lit_\gamma^3, ?v_\gamma^3), \\ &(var_\gamma^1, iri_\gamma^1, ?v_\gamma^1), (var_\gamma^2, iri_\gamma^2, ?v_\gamma^2), (var_\gamma^3, iri_\gamma^3, ?v_\gamma^3)\}, \end{aligned}$$

where each var_γ^j and iri_γ^j, $1 \leq j \leq 3$, are the variable and the IRI corresponding to the variable of literal t_j in γ; that is, $var_\gamma^j = ?z$ and $iri_\gamma^j = iri_z$ if $t_j = z$ or $t_j = \neg z$.

Let $?u$ and $?s$ be fresh variables and r, $cy^\perp$, $cy^\top$ fresh IRIs. We define

$$\begin{array}{lll} B_{\text{base}} & = \{(?u, iri_{x_1}, \top), \dots, (?u, iri_{x_n}, \top)\}, & \\ B_i^\perp & = \{(?x_i, iri_{x_i}, \perp)\}, & \text{for all } 1 \leq i \leq n, \\ B_i^\top & = \{(?x_i, iri_{x_i}, \top)\}, & \text{for all } 1 \leq i \leq n, \\ B_{\text{valid}} & = \{(?s, r, ?s), & \\ & \quad (cy^\perp, iri_{y_1}, \perp), \dots, (cy^\perp, iri_{y_m}, \perp), & \\ & \quad (cy^\top, iri_{y_1}, \top), \dots, (cy^\top, iri_{y_m}, \top)\} \cup & \\ & \quad B_{\gamma_1}^* \cup \dots \cup B_{\gamma_k}^*, & \text{where } \psi = \gamma_1 \wedge \dots \wedge \gamma_k, \\ B_\psi & = \{(?s, r, ?s)\} \cup B_{\gamma_1} \cup \dots \cup B_{\gamma_k}, & \text{where } \psi = \gamma_1 \wedge \dots \wedge \gamma_k. \end{array}$$

For example, a visualisation of these patterns for

$$\phi = \forall x_1, x_2 \, \exists y_1, y_2 \, (\neg x_1 \vee y_1 \vee y_2) \wedge (\neg y_1 \vee \neg y_2 \vee x_2)$$

is shown in Fig. 6.

Finally, let

$$P = ((\dots((B_{\text{base}} \text{ OPT } B_1^\perp) \text{ OPT } B_1^\top) \text{ OPT } \dots \text{ OPT } B_n^\perp) \text{ OPT } B_n^\top) \text{ OPT } B_\psi,$$

and

$$\begin{aligned} P' = (((\dots((B_{\text{base}} \text{ OPT } B_1^\perp) \text{ OPT } B_1^\top) \text{ OPT } \dots \text{ OPT } B_n^\perp) \text{ OPT } B_n^\top) \\ \text{OPT } B_{\text{valid}}) \text{ OPT } B_\psi \end{aligned}$$

be two FILTER-free wwd-patterns in OF-normal form.

We next show that ϕ is true if and only if P is equivalent to P', starting with the forward direction.

Let ϕ be true, yet, for the sake of contradiction, P is not equivalent to P'. Then there is a graph G and mapping μ such that $\mu \in [\![P]\!]_G$, but $\mu \notin [\![P']\!]_G$. Since patterns P and P' have the same root B_{base}, which contains $?u$ as the only variable, we conclude that $?u \in \mathsf{dom}(\mu)$. Each $?x_i$ is also in $\mathsf{dom}(\mu)$ by the construction of P, since there is a homomorphism from the corresponding leaf $B_i^\top$ to the root B_{base}. However, it is not necessary that $(\mu(?x_i), iri_{x_i}, \top)$ is in G because if G contains a triple of the form $(c, iri_{x_i}, \perp)$ for some IRI c, we will have $(\mu(?x_i), iri_{x_i}, \perp) \in G$. Note also that nothing prevents G from containing both a triple $(c, iri_{x_i}, \perp)$ and a triple $(c, iri_{x_i}, \top)$ for some i. Depending on whether $?s \in \mathsf{dom}(\mu)$ or not, we have two cases.

Case 1 Let $?s \in \mathsf{dom}(\mu)$, that is, there is a homomorphism from B_ψ to G that aligns with the previous assignment of all $?x_i$. In particular, this means that $\mathsf{dom}(\mu) =$

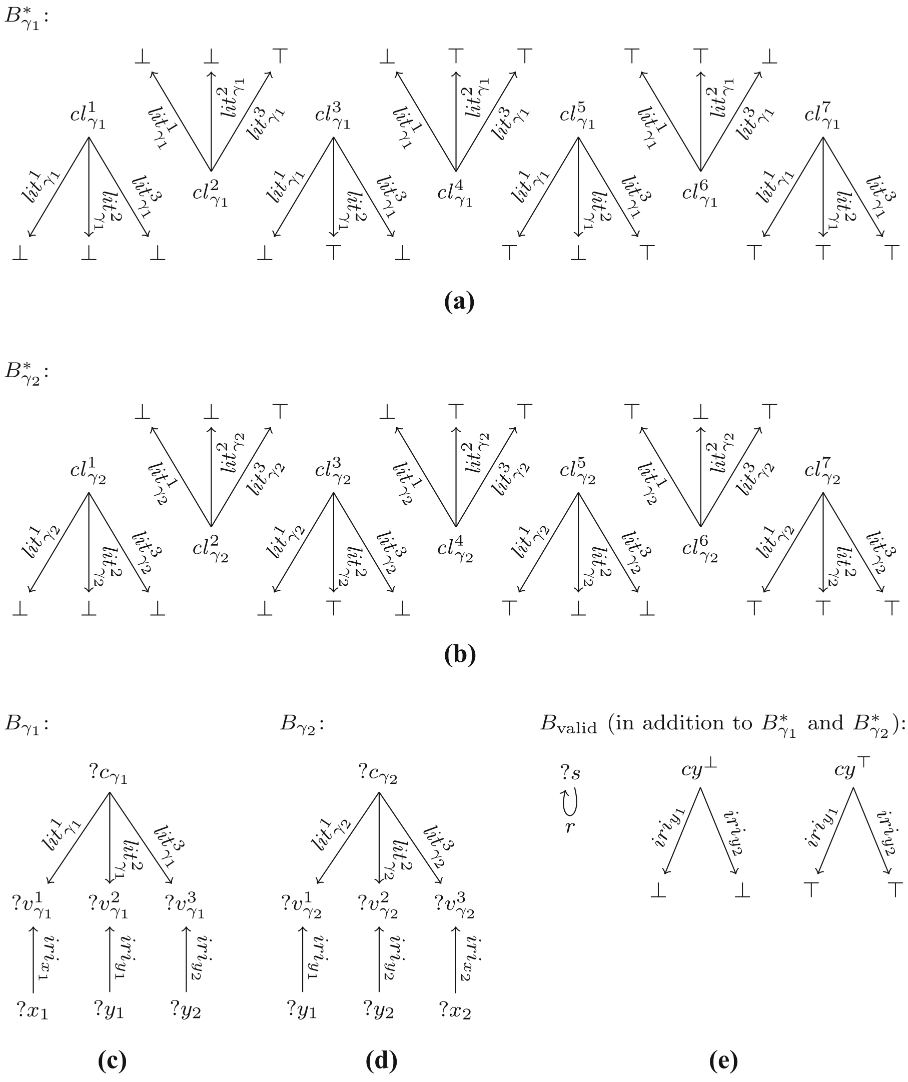
$B^*_{\gamma_1}$:
$cl^1_{\gamma_1}$ $cl^2_{\gamma_1}$ $cl^3_{\gamma_1}$ $cl^4_{\gamma_1}$ $cl^5_{\gamma_1}$ $cl^6_{\gamma_1}$ $cl^7_{\gamma_1}$
$lit^1_{\gamma_1}$ $lit^2_{\gamma_1}$ $lit^3_{\gamma_1}$
(a)
$B^*_{\gamma_2}$:
$cl^1_{\gamma_2}$ $cl^2_{\gamma_2}$ $cl^3_{\gamma_2}$ $cl^4_{\gamma_2}$ $cl^5_{\gamma_2}$ $cl^6_{\gamma_2}$ $cl^7_{\gamma_2}$
$lit^1_{\gamma_2}$ $lit^2_{\gamma_2}$ $lit^3_{\gamma_2}$
(b)
B_{γ_1}:
$?c_{\gamma_1}$ $?v^1_{\gamma_1}$ $?v^2_{\gamma_1}$ $?v^3_{\gamma_1}$
iri_{x_1} iri_{y_1} iri_{y_2}
$?x_1$ $?y_1$ $?y_2$
(c)
B_{γ_2}:
$?c_{\gamma_2}$ $?v^1_{\gamma_2}$ $?v^2_{\gamma_2}$ $?v^3_{\gamma_2}$
iri_{y_1} iri_{y_2} iri_{x_2}
$?y_1$ $?y_2$ $?x_2$
(d)
B_{valid} (in addition to $B^*_{\gamma_1}$ and $B^*_{\gamma_2}$):
$?s$ r
$cy^\bot$ $cy^\top$ iri_{y_1} iri_{y_2}
(e)

Fig. 6 Visualisation of the patterns (**a**) $B^*_{\gamma_1}$, (**b**) $B^*_{\gamma_2}$, (**c**) B_{γ_1}, (**d**) B_{γ_2}, and (**e**) B_{valid} used in the proof of Lemma 6 on the example formula $\forall x_1, x_2\, \exists y_1, y_2\; \gamma_1 \wedge \gamma_2$ with $\gamma_1 = \neg x_1 \vee y_1 \vee y_2$ and $\gamma_2 = \neg y_1 \vee \neg y_2 \vee x_2$

$\mathsf{vars}(P) = \mathsf{vars}(P')$. If there is no homomorphism from B_{valid} to G, then $\mu \in [\![P']\!]_G$, because B_ψ is the last leaf of P' as well, and nothing prevents it from matching. But this contradicts the assumption. However, even if there is a homomorphism h from B_{valid} to G, we still have a contradiction because $\mu \in [\![P']\!]_G$ still holds. Indeed, $?s$ is the only variable in B_{valid} and is essentially isolated in B_{valid}, so if h is a homomorphism from B_{valid} to G, then h', which maps $?s$ to $\mu(?s)$, is also such a homomorphism (in other words, since by the assumptions of this case, we know that $(?s, r, ?s)$ has a match in G, the existence of h just means that all the ground triples of B_{valid} are in G). This means, however, that nothing prevents B_ψ from matching in P', implying $\mu \in [\![P']\!]_G$.

Case 2 Let $?s \notin \mathsf{dom}(\mu)$. Since $\mu \notin [\![P']\!]_G$, there is no homomorphism from B_ψ to G but there is one from B_{valid} to G, that is, all ground triples of B_{valid} are in G (the non-existence of a homomorphism from B_ψ to G is immediate since $?s \notin \mathsf{dom}(\mu)$ and $\mu \in [\![P]\!]_G$; the existence of a homomorphism from B_{valid} to G then follows since otherwise we would have $\mu \in [\![P']\!]_G$). Consider now a truth assignment α of variables $\bar{x}$ such that if $\alpha(x_i)$ is true then $(\mu(?x_i), iri_{x_i}, \top) \in G$ and if $\alpha(x_i)$ is false then $(\mu(?x_i), iri_{x_i}, \bot) \in G$ (as we mentioned earlier, α may be not unique, but the argument does not depend on its uniqueness). Since ϕ is true, we know that α can be extended to the variables $\bar{y}$ in such a way that each clause in ψ holds. Let α' be such an extension, and let μ' be an extension of μ to the variables in B_ψ such that, for all j, $\mu'(?y_j) = cy^\top$ if $\alpha'(y_j)$ is true and $\mu'(?y_j) = cy^\bot$ otherwise. Then, for every clause γ in ψ, the IRIs $\mu'(?v_\gamma^1)$, $\mu'(?v_\gamma^2)$, $\mu'(?v_\gamma^3)$ correspond to the values $\alpha'(z_1)$, $\alpha'(z_2)$, $\alpha'(z_3)$, respectively, where z_1, z_2, z_3 are the variables in the literals t_1, t_2, t_3 of γ. Moreover, $\mu'(?c_\gamma) = cl_\gamma^\ell$ for ℓ the number of the assignment $\alpha'(z_1), \alpha'(z_2), \alpha'(z_3)$; this assignment makes γ true by the choice of α' (in other words, for every γ there is some ℓ such that $(\mu'(?c_\gamma), lit_\gamma^i, \mu'(?v_\gamma^i)) \in B_\gamma^\ell$ for all $1 \leq i \leq 3$). Hence, the extension μ' is contained in $[\![P]\!]_G$, and hence $\mu \notin [\![P]\!]_G$, which, however, contradicts the original assumption.

Since both cases yield a contradiction, we conclude that P is equivalent to P'.

We continue with the backward direction of the equivalence. Suppose that $P \equiv P'$, yet, for the sake of contradiction, ϕ is false. Then, there is a truth assignment α of the variables $\bar{x}$ such that for each extension α' of α to the variables $\bar{y}$ there is a clause γ in ψ that evaluates to false under α'. Fix such an α and consider the graph G consisting of the following triples:

– the triple $(u, iri_{x_i}, \top)$, for each x_i in $\bar{x}$ and a fresh IRI u;
– the triple $(c_{x_i}, iri_{x_i}, \top)$, for each x_i in $\bar{x}$ with $\alpha(x_i)$ true, and the triple $(c_{x_i}, iri_{x_i}, \bot)$, for each x_i in $\bar{x}$ with $\alpha(x_i)$ false, where $c_{x_1}, \ldots, c_{x_n}$ are fresh IRIs;
– all ground triples from B_{valid}, that is, all of its triples except $(?s, r, ?s)$;
– the triple (s, r, s) for a fresh IRI s.

Consider also the mapping μ such that

– $\mu(?u) = u$;
– $\mu(?x_i) = c_{x_i}$, for each x_i in $\bar{x}$.

Clearly, μ is a pp-solution to both P and P' over G. However, by the construction of G, the mapping $\mu' = \mu \cup \{?s \mapsto s\}$ is a pp-solution to P' over G as well. Thus, μ is not a solution to P' over G, and, since $P \subseteq P'$, we also have $\mu \notin [\![P]\!]_G$. Consequently, it must be possible to further extend μ' to a mapping μ'' that is both in $[\![P]\!]_G$ and in $[\![P']\!]_G$, and is defined on all variables in B_ψ. Essentially, this means that there is a homomorphism from B_ψ to B_{valid} that preserves $?s$. Consider now the extension α' of α to the variables $\bar{y}$ such that $\mu''(?y_j) = cy^\top$ if $\alpha'(y_j)$ is true and $\mu''(?y_j) = cy^\bot$ otherwise. By the construction of B_{valid}, this assignment validates all clauses in ψ, which, however, contradicts the assumption that ϕ is false.

Thus, we have shown that ϕ is true if and only if $P \equiv P'$. □

Theorem 7 *Problems* EQUIVALENCE($\mathcal{L}$) *and* CONTAINMENT($\mathcal{L}$) *are both* Π_2^p-*complete for any* $\mathcal{L} \in \{\mathcal{P}_{\text{wwd}}, \mathcal{U}_{\text{wwd}}\}$.

Proof The existence of a Π_2^p algorithm for containment immediately follows from Lemma 5: to show that $P \not\sqsubseteq P'$, for $P, P' \in \mathcal{P}_{\text{wwd}}$, we just need to guess, in NP, a graph G of linear size as well as a mapping μ, check that $\mu \notin [\![P']\!]_G$, and then call for a CONP oracle for checking that $\mu \in [\![P]\!]_G$. The claim for patterns in $\mathcal{U}_{\text{wwd}}$ is similar, but involves guessing a disjunct P_1 of P with $\mu \in [\![P_1]\!]_G$ and checking $\mu \notin [\![P_1']\!]_G$ for every disjunct P_1' of P'. Since $P \equiv P'$ if and only if containment holds in both directions, the problem EQUIVALENCE($\mathcal{U}_{\text{wwd}}$) is also in Π_2^p.

Hardness follows by the results in [35] for containment and by Lemma 6 for equivalence. □

Hence, for UNION- and FILTER-free patterns, the step from well-designed to weakly well-designed OPT incurs a complexity jump for containment and equivalence. However, for the fragments with UNION or projection complexity remains the same in both cases. As far as we are aware, these are the first decidability results on query equivalence and related problems for SPARQL fragments with OPT and FILTER.

8 Analysis of DBpedia Logs

In this section, we present an analysis of query logs over DBpedia, which suggests that the step from wd-patterns to wwd-patterns makes a dramatic difference in real life: while only about half of the queries with OPT have well-designed patterns, almost all of these patterns fall into the weakly well-designed fragment.

DBpedia [26] is a project providing access to RDF data extracted from Wikipedia via a SPARQL endpoint. DBpedia query logs are well suited for analysing the structure of real-life SPARQL queries as they contain a large amount of general-purpose knowledge base queries, generated both manually and automatically. DBpedia query logs have been analysed by Picalausa and Vansummeren [34], who reported that, over a period in 2010, about 46.38% of a total of 1344K distinct DBpedia queries used OPT. However, only 47.80% of the queries with OPT had well-designed patterns. Another analysis of DBpedia logs from the USEWOD 2011 data set performed by Arias Gallego et al. [7] concluded that 16.61% of about 5166K queries contained OPT; however, detailed structure of queries was not analysed.

We considered query logs over DBpedia 3.9 from USEWOD 2015 [30] and USEWOD 2016 [29]. The USEWOD 2015 DBpedia dataset is a random selection of almost 14M queries from the first half of 2014 while the USEWOD 2016 dataset contains 35M queries from the second half of 2015. We removed syntactically incorrect queries as well as queries outside of $\mathcal{S}$ (in particular, queries using operators specific to SPARQL 1.1). Also, we rewrote the patterns of the remaining queries to unions of UNION-free patterns as proposed in [33] and eliminated duplicates, which left us with 6.6M queries in USEWOD 2015 and 9.1M queries in USEWOD 2016. Finally, we isolated queries involving OPT and counted how many of their patterns were in $\mathcal{U}_{\mathrm{wd}}$ and in $\mathcal{U}_{\mathrm{wwd}}$.

The results are given in Table 1. They confirm that a non-negligible number of DBpedia queries use OPT (over 17%). However, by far not all queries with OPT are well-designed (only about 44% for USEWOD 2015 and 52% for USEWOD 2016), which is consistent with the results in [34]. On the other hand, almost all of the patterns with OPT (over 99.9% in both cases) are weakly well-designed, which we consider as the main practical justification for wwd-patterns.

What about the remaining 0.05% of queries with OPT? We looked at a number of such queries and identified what we believe to be the three most common sources of non-weakly-well-designedness in query patterns. The first and seemingly most common such source is joins between an OPT subpattern and another pattern on a variable that only occurs in the right argument of the OPT subpattern. The following query is an example of such a join:

$$\begin{aligned}&\mathsf{SELECT}\ ?\ell, ?u\ \mathsf{WHERE}\\ &\qquad ((?s, \texttt{label}, ?\ell)\ \mathsf{OPT}\ (?s, \texttt{type}, ?t))\ \mathsf{AND}\ (?t, \texttt{subClassOf}, ?u).\end{aligned} \tag{15}$$

We believe that the vast majority if not all such queries are erroneous as they are highly unlikely to yield meaningful answers in case the optional part fails to match. Intuitively, one would expect an answer to query (15) to contain the label of an object in variable $?\ell$ together with one of its supertypes in variable $?u$. And indeed, this is the answer returned by the query on graph G in Fig. 7a (see Fig. 7b). However, if an object in $?s$ is not assigned any type, which is explicitly allowed by the use of OPT, the query does not just return its label in $?\ell$ leaving $?u$ unbound, as one would expect; instead, it returns the cross product of the label and all types in the graph, which are, of course, completely unrelated to the object in $?s$ (see Fig. 7c and d).

Table 1 Structure of query patterns in DBpedia logs from USEWOD 2015 and 2016

	USEWOD 2015			USEWOD 2016		
	Unique patterns	Fraction of total	Fraction of patterns with OPT	Unique patterns	Fraction of total	Fraction of patterns with OPT
Total	6 606 201	100%		9 119 492	100%	
Patterns with OPT	1 147 704	17.37%	100%	1 582 698	17.36%	100%
Unions of wd-patterns	500 676	7.58%	43.62%	816 276	8.95%	51.57%
Unions of wwd-patterns	1 147 135	17.36%	99.95%	1 582 339	17.35%	99.98%

(S1, label, L1)
(S1, type, T1)
(T1, subClassOf, T1)

(a)

$?\ell$	$?u$
L1	T1

(b)

(S1, label, L1)
(S1, type, T1)
(T1, subClassOf, T1)
(S2, label, L2)

(c)

$?\ell$	$?u$
L1	T1
L2	T1

(d)

Fig. 7 (**a**) Graph G; (**b**) answers to query (15) over G; (**c**) graph G'; (**d**) answers over G'

A second source of non-weakly-well-designed patterns is joins between an OPT subpattern and another pattern where the left argument of the OPT subpattern is empty. The following query illustrates this source:

$$\textsf{SELECT } ?b, ?h, ?c \textsf{ WHERE} \\ (?b, \texttt{height}, ?h) \textsf{ AND } (\emptyset \textsf{ OPT } (?b, \texttt{city}, ?c)). \tag{16}$$

Intuitively, query (16) computes the join between the answers to the pattern $(?b, \texttt{height}, ?h)$ and those to $(?b, \texttt{city}, ?c)$, provided the second set is non-empty; otherwise, the query just returns the answers to $(?b, \texttt{height}, ?h)$. Hence, query (16) is equivalent to the following query in $\mathcal{S}_{\text{wwd}}$:

$$\begin{aligned} &\textsf{SELECT } ?b, ?h, ?c \textsf{ WHERE} \\ &\quad (((?b, \texttt{height}, ?h) \textsf{ OPT } (?b', \texttt{city}, ?c')) \textsf{ FILTER} \neg \mathit{bound}(?c')) \\ &\qquad \textsf{UNION } ((?b, \texttt{height}, ?h) \textsf{ AND } (?b, \texttt{city}, ?c)). \end{aligned}$$

We conclude that, while queries such as (16) may make sense in practice, they can be easily and intuitively restated using wwd-patterns.

Our final, and most interesting, source of real-life non-wwd-patterns is UNION in the right argument of OPT. For instance, consider the pattern

$$(?p, \texttt{type}, \texttt{person}) \textsf{ OPT } ((?p, \texttt{son}, ?a) \textsf{ UNION } (?p, \texttt{daughter}, ?a)). \tag{17}$$

The pattern is quite intuitive and it is easy to imagine similar patterns being useful in various applications. However, the normalisation algorithm in [33], which "pushes" unions outside, converts (17) to a pattern that is inherently non-well-designed (due to the two occurrences of $?a$ in the third disjunct):

$$\begin{aligned} &((?p, \texttt{type}, \texttt{person}) \textsf{ AND } (?p, \texttt{son}, ?a)) \\ &\quad \textsf{UNION} \\ &((?p, \texttt{type}, \texttt{person}) \textsf{ AND } (?p, \texttt{daughter}, ?a)) \\ &\quad \textsf{UNION} \\ &((((?p, \texttt{type}, \texttt{person}) \textsf{ OPT } ((?p, \texttt{son}, ?a) \textsf{ AND } (?x_1, ?y_1, ?z_1))) \textsf{ AND} \\ &\quad ((?p, \texttt{type}, \texttt{person}) \textsf{ OPT } ((?p, \texttt{daughter}, ?a) \textsf{ AND } (?x_2, ?y_2, ?z_2)))) \\ &\qquad \textsf{FILTER } \neg \mathit{bound}(?x_1) \wedge \neg \mathit{bound}(?x_2)). \end{aligned}$$

We believe that this behaviour is unavoidable in general as we expect query answering to become Π_2^p-hard over patterns that contain UNION in the right argument of OPT. Yet, for certain classes of patterns, generalising (17), one may be able to obtain a coNP evaluation algorithm by accounting for UNION natively rather than relying on the normalisation in [33]. A detailed study of such patterns, however, is outside the scope of the present paper.

9 Conclusion and Future Work

In this paper, we introduced a new fragment of SPARQL patterns called weakly well-designed patterns. This fragment extends the widely studied well-designed fragment by allowing variables from the optional side of an OPT-subpattern that are not "guarded" by the mandatory side to occur in certain positions outside of the subpattern. We showed that queries with wwd-patterns enjoy the same low complexity of evaluation as well-designed queries but cover almost all real-life queries. Moreover, our fragment is the maximal coNP fragment that does not impose structural restrictions on basic patterns and filter conditions. We studied the expressive power of the fragment and the complexity of its query optimisation problems.

For future work, we want to extend wwd-patterns to allow for non-top-level occurrences of UNION and projection. As we have seen in the previous section, this promises to be a challenging task since a naive extension of our definitions to such constructs is likely to increase reasoning complexity. Also, we want to take into account features of SPARQL 1.1 [17] such as GRAPH, NOT EXISTS and property paths. Finally, we would like to implement our ideas in a prototype system and compare its performance with existing SPARQL engines.

Acknowledgements This work was supported by the Engineering and Physical Sciences Research Council [grant numbers EP/J020214/1, EP/L012138/1, EP/N014359/1].

References

1. Ahmetaj, S., Fischl, W., Pichler, R., Simkus, M., Skritek, S.: Towards reconciling SPARQL and certain answers. In: Gangemi, A., Leonardi, S., Panconesi, A. (eds.) Proceedings of the 24th International Conference on World Wide Web, WWW 2015, pp. 23–33. ACM (2015)
2. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W., Thirunarayan, K. (eds.) ISWC 2008, LNCS, vol. 5318, pp. 114–129. Springer (2008)
3. Arenas, M., Conca, S., Pérez, J.: Counting beyond a Yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In: Mille, A., Gandon, F.L., Misselis, J., Rabinovich, M., Staab, S. (eds.) Proceedings of the 21st World Wide Web Conference, WWW 2012, pp. 629–638. ACM (2012)
4. Arenas, M., Gottlob, G., Pieris, A.: Expressive languages for querying the semantic web. In: Hull, R., Grohe, M. (eds.) Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2014, pp. 14–26. ACM (2014)
5. Arenas, M., Pérez, J.: Querying semantic web data with SPARQL. In: Lenzerini, M., Schwentick, T. (eds.) Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, pp. 305–316. ACM (2011)

6. Arenas, M., Ugarte, M.: Designing a query language for RDF: marrying open and closed worlds. In: Milo, T., Tan, W. (eds.) Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, pp. 225–236. ACM (2016)
7. Arias Gallego, M., Fernández, J.D., Martínez-Prieto, M.A., de la Fuente, P.: An empirical study of real-world SPARQL queries. In: Proceedings of the 1st International Workshop on Usage Analysis and the Web of Data, USEWOD 2011. arXiv:1103.5043 (2011)
8. Barceló, P., Pichler, R., Skritek, S.: Efficient evaluation and approximation of well-designed pattern trees. In: Milo, T., Calvanese, D. (eds.) Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, pp. 131–144. ACM (2015)
9. Bischof, S., Krötzsch, M., Polleres, A., Rudolph, S.: Schema-agnostic query rewriting in SPARQL 1.1. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) ISWC 2014, Part I, LNCS, vol. 8796, pp. 584–600. Springer (2014)
10. Buil Aranda, C., Arenas, M., Corcho, Ó.: Semantics and optimization of the SPARQL 1.1 federation extension. In: Antoniou, G., Grobelnik, M., Simperl, E.P.B., Parsia, B., Plexousakis, D., Leenheer, P.D., Pan, J.Z. (eds.) ESWC 2011, Part II, LNCS, vol. 6644, pp. 1–15. Springer (2011)
11. Buil Aranda, C., Polleres, A., Umbrich, J.: Strategies for executing federated queries in SPARQL 1.1. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) ISWC 2014, Part II, LNCS, vol. 8797, pp. 390–405. Springer (2014)
12. Chekol, M.W., Euzenat, J., Genevès, P., Layaïda, N.: SPARQL query containment under RDFS entailment regime. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012, LNCS, vol. 7364, pp. 134–148. Springer (2012)
13. Chekol, M.W., Euzenat, J., Genevès, P., Layaïda, N.: SPARQL query containment under SHI axioms. In: Hoffmann, J., Selman, B. (eds.) Proceedings of the 26th AAAI Conference on Artificial Intelligence, AAAI 2012, pp. 10–16. AAAI Press (2012)
14. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C. http://www.w3.org/TR/rdf11-concepts/ (2014)
15. Geerts, F., Unger, T., Karvounarakis, G., Fundulaki, I., Christophides, V.: Algebraic structures for capturing the provenance of SPARQL queries. J. ACM **63**(1), 7:1–7:63 (2016)
16. Halpin, H., Cheney, J.: Dynamic provenance for SPARQL updates. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) ISWC 2014, Part I, LNCS, vol. 8796, pp. 425–440. Springer (2014)
17. Harris, S., Seaborne, A.: SPARQL 1.1 query language. W3C recommendation, W3C. http://www.w3.org/TR/sparql11-query/ (2013)
18. Hayes, P.J., Patel-Schneider, P.F.: RDF 1.1 semantics. W3C recommendation, W3C. http://www.w3.org/TR/rdf11-mt/ (2014)
19. Kaminski, M., Kostylev, E.V.: Beyond well-designed SPARQL. In: Martens, W., Zeume, T. (eds.) Proceedings of the 19th International Conference on Database Theory, ICDT 2016, LIPIcs, vol. 48, pp. 5:1–5:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016)
20. Kaminski, M., Kostylev, E.V., Cuenca Grau, B.: Semantics and expressive power of subqueries and aggregates in SPARQL 1.1. In: Bourdeau, J., Hendler, J., Nkambou, R., Horrocks, I., Zhao, B.Y. (eds.) Proceedings of the 25th International Conference on World Wide Web, WWW 2016, pp. 227–238. ACM (2016)
21. Kontchakov, R., Kostylev, E.V.: On expressibility of non-monotone operators in SPARQL. In: Baral, C., Delgrande, J.P., Wolter, F. (eds.) Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning, KR 2016, pp. 369–379. AAAI Press (2016)
22. Kontchakov, R., Rezk, M., Rodriguez-Muro, M., Xiao, G., Zakharyaschev, M.: Answering SPARQL queries over databases under OWL 2 QL entailment regime. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) ISWC 2014, Part I, LNCS, vol. 8796, pp. 552–567. Springer (2014)
23. Kostylev, E.V., Cuenca Grau, B.: On the semantics of SPARQL queries with optional matching under entailment regimes. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) ISWC 2014, Part II, LNCS, vol. 8797, pp. 374–389. Springer (2014)
24. Kostylev, E.V., Reutter, J.L., Romero, M., Vrgoč, D.: SPARQL with property paths. In: Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P.T., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) ISWC 2015, Part I, LNCS, vol. 9366, pp. 3–18. Springer (2015)

25. Kostylev, E.V., Reutter, J.L., Ugarte, M.: CONSTRUCT queries in SPARQL. In: Arenas, M., Ugarte, M. (eds.) Proceedings of the 18th International Conference on Database Theory, ICDT 2015, LIPIcs, vol. 31, pp. 212–229. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015)
26. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web **6**(2), 167–195 (2015)
27. Letelier, A., Pérez, J., Pichler, R., Skritek, S.: Static analysis and optimization of semantic web queries. ACM Trans. Database Syst. **38**(4), 25 (2013)
28. Losemann, K., Martens, W.: The complexity of evaluating path expressions in SPARQL. In: Benedikt, M., Krötzsch, M., Lenzerini, M. (eds.) Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, pp. 101–112. ACM (2012)
29. Luczak-Rösch, M., Aljaloud, S., Berendt, B., Hollink, L.: USEWOD 2016 research dataset. doi:10.5258/SOTON/385344 (2016)
30. Luczak-Rösch, M., Berendt, B., Hollink, L.: USEWOD 2015 research dataset. doi:10.5258/SOTON/379407 (2015)
31. Manola, F., Miller, E., McBride, B.: RDF 1.1 primer. W3C working group note, W3C. http://www.w3.org/TR/rdf11-primer/ (2014)
32. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: Cruz, I.F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) ISWC 2006, LNCS, vol. 4273, pp. 30–43. Springer (2006)
33. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. **34**(3), 16 (2009)
34. Picalausa, F., Vansummeren, S.: What are real SPARQL queries like? In: Virgilio, R.D., Giunchiglia, F., Tanca, L. (eds.) Proceedings of the 3rd International Workshop on Semantic Web Information Management, SWIM 2011, pp. 7:1–7:6. ACM (2011)
35. Pichler, R., Skritek, S.: Containment and equivalence of well-designed SPARQL. In: Hull, R., Grohe, M. (eds.) Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2014, pp. 39–50. ACM (2014)
36. Polleres, A., Wallner, J.P.: On the relation between SPARQL 1.1 and answer set programming. J. Appl. Non-Classical Log. **23**(1–2), 159–212 (2013)
37. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C recommendation, W3C. http://www.w3.org/TR/rdf-sparql-query/ (2008)
38. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL query optimization. In: Segoufin, L. (ed.) Proceedings of the 13th International Conference on Database Theory, ICDT 2010, pp. 4–33. ACM (2010)
39. Zhang, X., Van den Bussche, J.: On the power of SPARQL in expressing navigational queries. Comput. J. **58**(11), 2841–2851 (2015)
40. Zhang, X., Van den Bussche, J.: On the primitivity of operators in SPARQL. Inf. Process. Lett. **114**(9), 480–485 (2014)
41. Zhang, X., Van den Bussche, J., Picalausa, F.: On the satisfiability problem for SPARQL patterns. J. Artif. Intell. Res. (JAIR) **56**, 403–428 (2016)