SPARQL with Property Paths on the Web

Olaf Hartig\textsuperscript{a,b,*}, Giuseppe Pirrò\textsuperscript{c}

\textsuperscript{a} Hasso Plattner Institute, Universität Potsdam, Germany
\textsuperscript{b} Department of Computer and Information Science (IDA), Linköping University, Sweden
\textsuperscript{c} Italian National Research Council (ICAR-CNR), Rende(CS), Italy

E-mail: olaf.hartig@liu.se
E-mail: pirro@icar.cnr.it

Abstract. Linked Data on the Web represents an immense source of knowledge suitable to be automatically processed and queried. In this respect, there are different approaches for Linked Data querying that differ on the degree of centralization adopted. On one hand, the SPARQL query language, originally defined for querying single datasets, has been enhanced with features to query federations of datasets; however, this attempt is not sufficient to cope with the distributed nature of data sources available as Linked Data. On the other hand, extensions or variations of SPARQL aim to find trade-offs between centralized and fully distributed querying. The idea is to partially move the computational load from the servers to the clients. Despite the variety and the relative merits of these approaches, as of today, there is no standard language for querying Linked Data on the Web. A specific requirement for such a language to capture the distributed, graph-like nature of Linked Data sources on the Web is a support of graph navigation. Recently, SPARQL has been extended with a navigational feature called property paths (PPs). However, the semantics of SPARQL restricts the scope of navigation via PPs to single RDF graphs. This restriction limits the applicability of PPs for querying distributed Linked Data sources on the Web. To fill this gap, in this paper we provide formal foundations for evaluating PPs on the Web, thus contributing to the definition of a query language for Linked Data. We first introduce a family of reachability-based query semantics for PPs that distinguish between navigation on the Web and navigation at the data level. Thereafter, we consider another, alternative query semantics that couples Web graph navigation and data level navigation; we call it context-based semantics. Given these semantics, we find that for some PP-based SPARQL queries a complete evaluation on the Web is not possible. To study this phenomenon we introduce a notion of Web-safeness of queries, and prove a decidable syntactic property that enables systems to identify queries that are Web-safe. In addition to establishing these formal foundations, we conducted an experimental comparison of the context-based semantics and a reachability-based semantics. Our experiments show that when evaluating a PP-based query under the context-based semantics one experiences a significantly smaller number of dereferencing operations, but the computed query result may contain less solutions.

Keywords: Property paths, Web navigational language, Web safeness, SPARQL

1. Introduction

The increasing trend in sharing and interlinking pieces of structured data on the World Wide Web (WWW) is evolving the classical Web—which is focused on hypertext documents and syntactic links among them—into a Web of Linked Data. The Linked Data principles [5] present an approach to extend the scope of Uniform Resource Identifiers (URIs) to new types of resources (e.g., people, places) and represent their descriptions and interlinks by using the Resource Description Framework (RDF) [8] as standard data format. RDF adopts a graph-based data model, which can be queried by using the SPARQL query language [15]. When it comes to Linked Data on the WWW, the common way to provide query-based ac-
cess is via SPARQL endpoints; that is, services that usually answer SPARQL queries over a single dataset. Recently, the original core of SPARQL has been extended with features supporting query federation; it is now possible, within a single query, to target multiple endpoints (via the `SERVICE` operator). However, such an extension is not enough to cope with an unbounded and a priori unknown space of data sources such as the WWW. Moreover, not all Linked Data on the WWW is accessible via SPARQL endpoints. More recent proposals are based on the idea of Linked Data Fragments [39,40] and aim at moving part of the computational load from Web servers to clients.

However, as of today, there exists no standard query language for Linked Data on the WWW, although SPARQL is clearly a candidate. A key feature that such a language should provide is navigation across the unbounded, a priori unknown, graph-like environment represented by distributed Linked Data sources.

While earlier research on using SPARQL for Linked Data is limited to fragments of the first version of the language [6,16,18,38], the version 1.1 of SPARQL introduces a feature called `property paths` (PPs) that equips the language with navigational capabilities [15]. However, the standard definition of PPs is limited to single RDF graphs and, thus, not directly applicable to Linked Data that is distributed over the WWW.

Therefore, toward the definition of a language for accessing Linked Data live on the WWW, the following questions emerge naturally:

**How can PPs be defined over the WWW?**

and

**What are the implications of such a definition?**

Answering these questions is the broad objective of this paper. In particular, we focus on Linked Data on the WWW, by which we mean RDF data that is made available on the WWW as per the Linked Data principles [5] and, thus, can be accessed by looking up HTTP scheme based URLs. In this context we make the following main contributions:

1. We formalize a family of reachability-based query semantics of PP-based SPARQL queries that are meant to be evaluated over Linked Data on the WWW. This formalization approach treats navigation on the Web separate from navigation on the level of data.
2. We also formalize an alternative, context-based query semantics that intertwines Web graph navigation and data level navigation.
3. We study the feasibility of evaluating queries under these semantics. For this study we assume that query engines do not have complete information about the queried Web of Linked Data (as it is the case for the WWW). Our study shows that query evaluation under any reachability-based semantics is possible in practice and that a similarly general statement cannot be made for the context-based semantics; that is, there exist cases in which query evaluation under the context-based semantics is not possible.
4. We establish a decidable syntactic property of queries for which an evaluation under the context-based semantics is possible.
5. We provide an experimental comparison of the context-based and a reachability-based semantics. For this comparison we executed queries directly over the WWW. As its main result, our experiment shows that when evaluating a PP-based query under the context-based semantics, one experiences a significantly smaller number of dereferencing operations, but the computed query result may contain less solutions.

This article extends a preliminary version that appeared in the proceedings of the ESWC 2015 conference [21]. The extension includes: (i) the definition and analysis of a family of reachability-based query semantics for Property Paths on the Web; (ii) an experimental analysis and comparison of the different semantics; (iii) a more detailed description of the main technical results; (iv) further examples to better clarify the terminology and the main concepts of the paper; (v) a more comprehensive discussion of related work.

The paper is organized as follows. Section 2 provides an overview on related work. In Section 3 we introduce the formal framework for this paper, including a data model that captures the notion of Linked Data on the WWW. Section 4 focuses on PPs, isolated from other SPARQL operators. In Section 5 we broaden our view to define PP-based SPARQL graph patterns. In Section 6 we characterize a class of Web-safe patterns and prove their feasibility. Section 7 discusses the experimental evaluation. Finally, in Section 8 we conclude.

### 2. Related Work

There is an extensive body of research on the foundations of querying RDF data. An important work in this context is the investigation of SPARQL provided
by Peréz et al. [30]. Other authors focused on the foundations of SPARQL query optimization [34, 26].

From the perspective of graphs, languages for the navigation and specification of vertices in graphs have a long tradition (see Wood’s survey [41]). For RDF, extensions of SPARQL such as PS sparql [2], nsparql [31], and SPARQLeR [23] introduced navigational features since these were missing in the first version of SPARQL. Only recently, with the addition of property paths (PPs) in version 1.1 [15], SPARQL has been enhanced officially with such features. The final definition of PPs has been influenced by research that studied the computational complexity of an early draft version of PPs [3, 27]. There also already exists a proposal to extend the expressive power of PPs [11].

Other strands of research focus on studying properties of PPs such as containment [25] or supporting recursion in SPARQL [32]. However, the main assumption of all these navigational extensions of SPARQL is to work on a single, centralized RDF graph.

The idea of querying the WWW as a database is not new (see Florescu et al.’s survey [13]). Perhaps the most notable early works in this context are by Konopnicki and Shmueli [24], Abiteboul and Vianu [1], and Mendelzon et al. [28], all of which tackled the problem of evaluating SQL-like queries on the hypertext Web. While such queries included navigational features, the focus was on retrieving specific Web pages, particular attributes of specific pages, or content within them.

Our departure point is different: We aim at defining semantics of SPARQL queries (including property paths) over Linked Data on the WWW; this involves dealing with two graphs of different type; namely, an RDF graph that is distributed over an unbounded number of documents on the WWW and the Web graph in which these documents are interlinked with each other.

To express queries over Linked Data on the WWW, two main strands of research can be identified. The first studies how to extend the scope of SPARQL queries to the WWW, with existing work focusing on basic graph patterns [6, 16, 38] or a more expressive fragment that includes AND, OPT, UNION and FILTER [18]. The second strand of research focuses on emphasizing navigational features, which resulted in new languages such as NautiLOD [10, 12], LDPath [33], and LDQL [20].

These two strands have different departure points. The former employs navigation over the WWW to collect data for answering a given SPARQL query; here navigation is a means to discover query-relevant data. The latter provides explicit navigational features and uses querying capabilities to filter data sources of interest; here navigation (not querying) is the main focus. The context-based query semantics proposed in this paper combines both approaches.

Another line of research slightly related to our proposal is that of focused crawling. The idea is to enhance the behavior of classical Web crawlers, that consider all pages reachable from a given page, to be more selective; selectivity is obtained by considering e.g., a set of predefined topics [36] or meta data within HTML pages [29]. A more recent line of related research looks into building (domain-specific) knowledge graphs by exploiting semantic technologies to reconcile the data continuously crawled from diverse sources [35]. In a way, these approaches mimic the process of filtering performed by our approach but on a less expressive scale due to the limited expressiveness of the filtering mechanism as compared to our language. Nevertheless, our approach could be used to enable a finer-grained information filtering.

3. Formal Framework

This section provides a formal framework for defining semantics of PPs over Linked Data. In particular, we first recall the definition of PPs as per the SPARQL standard [15]. Thereafter, we introduce a data model that captures the notion of Linked Data on the WWW.

3.1. Preliminaries

We assume four pairwise disjoint, countably infinite sets $\mathcal{I}$ (IRIs), $\mathcal{B}$ (blank nodes), $\mathcal{L}$ (literals), and $\mathcal{V}$ (variables, denoted by a leading ‘?’ symbol). An RDF triple (or simply triple) is a tuple from the set $\mathcal{T} = (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{V})$. For any such triple $t = (s, p, o)$ we call $s$ the subject, $p$ the predicate, and $o$ the object, and we write iris($t$) to denote the set of all IRIs in the triple; i.e., iris$(t) = \{s, p, o\} \cap \mathcal{I}$. A set of triples is called an RDF graph.

A property path pattern (or PP pattern for short) is a tuple $P = (\alpha, path, \beta)$ with $\alpha \in (\mathcal{I} \cup \mathcal{L} \cup \mathcal{V})$, $\beta \in (\mathcal{I} \cup \mathcal{L} \cup \mathcal{V})$, and path is a property path expression (PP expression) that is defined by the following grammar (where $u, u_1, \ldots, u_n \in \mathcal{I}$):

\[
\text{path} = u \mid !(u_1 \ldots u_n) \mid \text{path/path} \mid (\text{path}/\text{path}) \mid (\text{path})^* \mid ^\text{path}
\]

As can be seen from this grammar, we have two base cases for PP expressions, namely, arbitrary IRIs
\( M_1 \times M_2 = (\Omega, \text{card}) \) such that \( \Omega = \{ \mu_1 \cup \mu_2 \mid (\mu_1, \mu_2) \in \Omega_1 \times \Omega_2 \text{ and } \mu_1 \sim \mu_2 \} \) and for every solution mapping \( \mu \in \Omega \) we have \( \text{card(}\mu\text{)} = \sum_{(\mu_1, \mu_2) \in \Omega_1 \times \Omega_2 \text{ s.t. } \mu_1 \cup \mu_2} (\text{card(}\mu_1\text{)} \cdot \text{card(}\mu_2\text{))}. \)

\( M_1 \setminus M_2 = (\Omega, \text{card}) \) such that \( \Omega = \{ \mu_1 \in \Omega_1 \mid \exists \mu_2 \in \Omega_2 : \mu_1 \sim \mu_2 \} \) and for every solution mapping \( \mu \in \Omega \) we have \( \text{card(}\mu\text{)} = \text{card(}\mu_1\text{)}. \)

\( M_1 \cup M_2 = (\Omega, \text{card}) \) such that \( \Omega = \Omega_1 \cup \Omega_2 \) and (i) \( \text{card(}\mu\text{)} = \text{card(}\mu_1\text{)} \) for all solution mappings \( \mu \in \Omega \setminus \Omega_2 \), (ii) \( \text{card(}\mu\text{)} = \text{card(}\mu_2\text{)} \) for all \( \mu \in \Omega \setminus \Omega_1 \), and (iii) \( \text{card(}\mu\text{)} = \text{card(}\mu_1\text{)} + \text{card(}\mu_2\text{)} \) for all \( \mu \in \Omega_1 \cap \Omega_2 \).

\( \pi_V(M_1) = (\Omega, \text{card}) \) such that \( \Omega = \{ \mu \mid \exists \mu' \in \Omega_1 : \mu \sim \mu' \text{ and card}(\mu) = V \cap \text{card}(\mu') \} \) and for every solution mapping \( \mu \in \Omega \) we have \( \text{card}(\mu) = \sum_{\mu' \in \Omega_1, s.t. \mu \sim \mu'} \text{card}(\mu'). \)

Fig. 1. SPARQL algebra operators over multisets of solution mappings, \( M_1 = (\Omega_1, \text{card}_1) \) and \( M_2 = (\Omega_2, \text{card}_2) \).

and expressions of the form \( !(|u_1| \ldots |u_n|) \). PP patterns based on the former are ordinary triple patterns, which, in the context of PPs, represent single navigation steps from the subject to the object of any triple whose predicate is the given IRI. The second base case captures a form of negation that represents a navigation step along any triple whose predicate is not among the IRIs listed. Given these base types of PP expressions, users may combine them via the classical regular expression operators: concatenation \( \cdot \), disjunction \( | \), and recursive concatenation \( (\cdot)^* \); additionally, \( ^{\text{path}} \) represents the inverse of \( \text{path} \) (a formal semantics of PP patterns and PP expressions follows shortly).

The SPARQL standard introduces additional types of PP expressions [15]. Since these are merely syntactic sugar (they are defined in terms of expressions covered by the grammar given above), we ignore them in this paper. As another slight deviation from the standard, we do not permit blank nodes in PP patterns (i.e., \( \alpha, \beta \notin \mathcal{B} \)). However, standard PP patterns with blank nodes can be simulated using fresh variables.

**Example 1.** As an example of a PP pattern consider \( (\text{Tim}, \text{knows}^*)/\text{name}, ?n) \) where \( ?n \in \mathcal{V} \) and \( \text{Tim}, \text{knows}, \text{name} \in \mathcal{I} \). This pattern retrieves the names of persons that can be reached from \( \text{Tim} \) by an arbitrarily long path of \text{knows} \ relationships (which includes \text{Tim}). Another example are the two PP patterns \( (?p, \text{knows}, \text{Tim}) \) and \( (\text{Tim}, ^{\text{path}} \text{knows}, ?p) \), both of which retrieve persons that know \text{Tim}. For further examples we refer to the SPARQL specfication [15, Section 9.2].

In addition to a syntax for the queries of interest, we have to introduce the standard semantics of these queries. The SPARQL specification defines this semantics by an evaluation function (see below) that returns multisets of so-called solution mappings; such a mapping is a partial function \( \mu : \mathcal{V} \rightarrow (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}) \).

To refer to the domain of a solution mapping \( \mu \) (i.e., the set of variables for which \( \mu \) is defined) we write \( \text{dom}(\mu) \). If, for two solution mappings, say \( \mu_1 \) and \( \mu_2 \), we have \( \mu_1(\cdot?) = \mu_2(\cdot?) \) for every variable \( ?v \in (\text{dom}(\mu_1) \cap \text{dom}(\mu_2)) \), then we say that \( \mu_1 \) and \( \mu_2 \) are compatible \( (\mu_1 \sim \mu_2) \). In this case, \( \mu_1 \) and \( \mu_2 \) can be combined into a solution mapping \( \mu = \mu_1 \cup \mu_2 \) such that \( \text{dom}(\mu) = (\text{dom}(\mu_1) \cup \text{dom}(\mu_2)) \cup \mu_1 \sim \mu_2 \). Given a solution mapping \( \mu \) and a PP pattern \( P \), we write \( \mu[P] \) to denote the PP pattern obtained by replacing the variables in \( P \) according to \( \mu \) (where variables for which \( \mu \) is not defined are not replaced).

We represent a multiset of solution mappings by a pair \( M = (\Omega, \text{card}) \) where \( \Omega \) is the underlying set (of solution mappings) and \( \text{card} \) is the corresponding cardinality function; i.e., \( \text{card} : \Omega \rightarrow \{1, 2, \ldots \} \). By abusing notation slightly, we write \( \mu \in M \) for every \( \mu \in \Omega \). Furthermore, to simplify the following definitions we introduce a family of special, parameterized cardinality functions for multisets in which every solution mapping has a cardinality of 1. That is, for any set of solution mappings \( \Omega \), let \( \text{card}^{(1)} : \Omega \rightarrow \{1, 2, \ldots \} \) be the constant-1 cardinality function that is defined by \( \text{card}^{(1)}(\mu) = 1 \) for all \( \mu \in \Omega \).

To define the aforementioned evaluation function we also need to introduce several operators of the SPARQL algebra, which is defined over multisets of solution mappings. That is, for two such multisets, \( M_1 = (\Omega_1, \text{card}_1) \) and \( M_2 = (\Omega_2, \text{card}_2) \), we define the join \((\cdot\cup)\), the difference \((\cdot\setminus)\), the multiset union \((\cdot\sqcup)\), and projection \((\pi_V)\), where \( V \subseteq \mathcal{V} \) is a finite set of variables as given in Figure 1. In addition to these algebra operators, the SPARQL standard introduces auxiliary functions to define the semantics of PP patterns of the form \( \langle \alpha, ^{\text{path}} \beta \rangle \). Figure 2 provides these functions—which we call ALP1 and ALP2—adapted to our formalism (we need a variable \( ?x \) in line 6 since PP patterns in our formalism do not have blank nodes).

We are now ready to define the evaluation function that formalizes the standard semantics of PP patterns.
we have $P$ is defined recursively as given in Figure 3.

RDF graph. The Definition 2.

<table>
<thead>
<tr>
<th>Function ALP1($\gamma$, path, $G$)</th>
<th>Function ALP2($\gamma$, path, Visited, $G$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> $\gamma \in (I \cup B \cup L)$, path is a PP expression, $G$ is an RDF graph.</td>
<td><strong>Input:</strong> $\gamma \in (I \cup B \cup L)$, path is a PP expression, Visited $\subseteq (I \cup B \cup L)$, $G$ is an RDF graph.</td>
</tr>
<tr>
<td>$1$: Visited $:= \emptyset$</td>
<td>$4$: if $\gamma \notin$ Visited then</td>
</tr>
<tr>
<td>$2$: ALP2($\gamma$, path, Visited, $G$)</td>
<td>$5$: add $\gamma$ to Visited</td>
</tr>
<tr>
<td>$3$: return Visited</td>
<td>$6$: for all $\mu \in [{x, path, y}]_G$ such that $\mu(x) = \gamma$ and $x, y \in V$ do</td>
</tr>
<tr>
<td></td>
<td>$7$: ALP2($\mu(y)$, path, Visited, $G$)</td>
</tr>
</tbody>
</table>

Fig. 2. Auxiliary functions used for defining the semantics of PP expressions of the form path$^*$.

$$[[\langle \alpha, \beta \rangle]]_G = \langle \{ \mu \mid \text{dom}(\mu) = (\{\alpha, \beta\} \cap V) \text{ and } \mu(\alpha, \beta) \in G \} \rangle, \quad \text{card}_1(\Omega)$$

$$[[\langle \alpha, !\{u_1, \ldots, u_n\}, \beta \rangle]]_G = \langle \{ \mu \mid \text{dom}(\mu) = (\{\alpha, \beta\} \cap V) \text{ and there exists an IRI } u \in I \text{ such that } u \notin \{u_1, \ldots, u_n\} \text{ and } \mu(\alpha, u, \beta) \in G \} \rangle, \quad \text{card}_1(\Omega)$$

$$[[\alpha, path, \beta]]_G = \langle \{ \mu \mid \text{dom}(\mu) = (\{\alpha, \beta\} \cap V) \text{ and there exists an IRI } u \in I \text{ such that } u \notin \{u_1, \ldots, u_n\} \text{ and } \mu(\alpha, u, \beta) \in G \} \rangle, \quad \text{card}_1(\Omega)$$

$$[[\alpha, path_1/path_2, \beta]]_G = \pi_{\langle \alpha, \beta \rangle \cap V} ([[\alpha, path_1, ?v]]_G \times [[?v, path_2, \beta]]_G)$$

$$[[\langle x_L, \langle path \rangle^*, ?v_R \rangle]]_G = \langle \{ \mu \mid \text{dom}(\mu) = \{?v_R\} \text{ and } \mu(?v_R) \in \text{ALP1}(x_L, path, G) \} \rangle, \quad \text{card}_1(\Omega)$$

$$[[\langle ?v_L, \langle path \rangle^*, ?v_R \rangle]]_G = \langle \{ \mu \mid \text{dom}(\mu) = \{?v_L, ?v_R\} \text{ and } \mu(?v_L) \in \text{terms}(G) \text{ and } \mu(?v_R) \in \text{ALP1}(?v_L, path, G) \} \rangle, \quad \text{card}_1(\Omega)$$

$$[[\langle ?v_L, \langle path \rangle^*, x_R \rangle]]_G = \langle \{ \mu \mid \text{dom}(\mu) = \{?v_L, ?v_R\} \text{ and } \mu(?v_R) \in \text{ALP1}(\mu(?v_L), path, G) \} \rangle, \quad \text{card}_1(\Omega)$$

Fig. 3. Standard query semantics of SPARQL Property Paths, where $\alpha, \beta \in (I \cup L \cup V)$: $u, u_1, \ldots, u_n \in I; x_L, x_R \in (I \cup L); ?v_L, ?v_R \in V; ?v \in V$ is a fresh variable; and $\mu_\emptyset$ is the empty solution mapping with $\text{dom}(\mu_\emptyset) = \emptyset$.

**Definition 2.** Let $P$ be a PP pattern and let $G$ be an RDF graph. The evaluation of $P$ over $G$, denoted by $[P]_G$, is a multiset of solution mappings $(\Omega, \text{card})$ that is defined recursively as given in Figure 3.

**Example 3.** Consider the following RDF graph:

$$G_{\text{ex}} = \{ \langle \text{Suzi}, \text{knows}, \text{Eve} \rangle, \langle \text{Eve}, \text{knows}, \text{Charlie} \rangle, \langle \text{Suzi}, \text{knows}, \text{Alice} \rangle, \langle \text{Alice}, \text{knows}, \text{Charlie} \rangle, \langle \text{Alice}, \text{knows}, \text{Eve} \rangle \}.$$ 

Then, for the PP pattern $P_a = \langle \text{Suzi}, \text{knows}, ?x \rangle$ we have $[P_a]_{G_{\text{ex}}} = \{\Omega_a, \text{card}_a\}$ with $\Omega_a = \{\mu_{a1}, \mu_{a2}\}$.

$$\mu_{a1}(?x) = \text{Charlie} \quad \text{where card}_a(\mu_{a1}) = 2, \quad \text{and} \quad \mu_{a2}(?x) = \text{Eve} \quad \text{where card}_a(\mu_{a2}) = 1.$$ 

Note that the result contains the solution mapping $\mu_{a1}$ twice because Charlie can be reached from Suzi by two different paths that match the PP expression knows/ knows (namely, one via Eve, the other via Alice).

**Example 4.** As another example, consider PP pattern $P_b = \langle \text{Suzi}, \langle \text{knows} \rangle^*, ?x \rangle$, for which we have:

$$[P_b]_{G_{\text{ex}}} = \{\Omega_b, \text{card}_b\}, \text{where}$$

$$\mu_{b1}(?x) = \text{Suzi}, \quad \mu_{b2}(?x) = \text{Eve}, \quad \mu_{b3}(?x) = \text{Alice}, \quad \mu_{b4}(?x) = \text{Charlie},$$

and $\text{card}_b(\mu_{bi}) = 1$ for all $i \in \{1, 2, 3, 4\}$. The latter may be surprising at first. However, for the PP pattern $P_b$, as for every PP pattern whose PP expression is of the form $(\text{path})^*$, the SPARQL specification digresses from the standard bag semantics of other PP patterns.
to an existential semantics where every solution mapping is counted only once, even if there exist multiple matching paths with the same target node (the procedural definition represented by function ALP2 achieves this effect by ignoring already visited elements; cf. line 4 in Figure 2).

3.2. Data Model

The standard query semantics of PP patterns—as introduced in the SPARQL specification and presented in the previous section—defines the result expected from evaluating such a pattern over a (single) RDF graph. Since the WWW is not an RDF graph, this standard definition is insufficient as a formal foundation for evaluating PP patterns over Linked Data on the WWW. As a basis for providing a suitable definition we need a data model that captures the notion of a Web of Linked Data. To this end, we adopt the data model introduced in our earlier work [18].

For this model we assume an infinite set \( D \) that is disjoint from the aforementioned sets \( I \) (IRIs), \( B \) (blank nodes), \( L \) (literals), and \( V \) (variables). Elements in this set \( D \) represent the concept of Web documents from which Linked Data can be extracted; hereafter, we call each \( d \in D \) a Linked Data document, or document for short. Moreover, we assume a function \( data : D \rightarrow 2^T \) that maps every document \( d \in D \) to a finite set of triples \( data(d) \subseteq T \). As prescribed by the RDF data model [8], we require that the triples of each document use a unique set of blank nodes; i.e., for any pair of distinct documents \( d, d' \in D \), there does not exist two triples \( t = \langle s, p, o \rangle \) and \( t' = \langle s', p', o' \rangle \) such that \( t \in data(d), t' \in data(d') \), and \( \{s, p, o\} \cap \{s', p', o'\} \cap B = \emptyset \). Given these preliminaries, we define a Web of Linked Data as follows.

**Definition 5.** Assume a special symbol \( \bot \) such that \( \bot \notin (D \cup I \cup B \cup L \cup V) \). A Web of Linked Data is a tuple \( W = (D, adoc) \) with the following two elements:

1. \( D \subseteq D \) is a set of documents; and
2. \( adoc \) is a function that maps every IRI \( u \in I \) either to a document \( d \) in \( D \) or to the symbol \( \bot \) (i.e., \( adoc : I \rightarrow D \cup \{\bot\} \)) such that for every \( d \in D \), there exists an IRI \( u \in I \) with \( adoc(u) = d \).

Observe that the function \( adoc \) captures the concept of obtaining documents by looking up (HTTP) IRIs on the WWW (also referred to as dereferencing). IRIs that cannot be looked up, or whose look up does not result in retrieving a document (even after following HTTP-based redirection pointers) are mapped to the special symbol \( \bot \). In this paper we assume that in any Web of Linked Data \( W = (D, adoc) \) the set of documents \( D \) is finite, in which case we say \( W \) is finite (for a discussion of infiniteness refer to our earlier work [18]).

For the subsequent discussion we introduce a few additional concepts: Given a Web of Linked Data \( W = (D, adoc) \), we write \( dom^D(adoc) \) to denote the set of IRIs that function \( adoc \) maps to a document; i.e., \( dom^D(adoc) = \{u \in I \mid adoc(u) \neq \bot\} \) (hence, this set corresponds to what is also referred to as “dereferenceable IRIs”). Moreover, for any two documents \( d, d' \in D \), we say that document \( d \) has a data link to \( d' \) if there exists some triple \( t = \langle s, p, o \rangle \) in the data of \( d \) (i.e., \( t \in data(d) \)) such that \( t \) contains an IRI that can be used to obtain \( d' \), i.e., \( adoc(u) = d' \) for some \( u \in \{s, p, o\} \). Such data links establish the link graph of the Web of Linked Data \( W \), that is, a directed graph \((D, E)\) in which the edges \( E \) are all pairs \((d, d') \in D \times D\) for which \( d \) has a data link to \( d' \). We emphasize that the link graph of \( W \) is a different type of graph than the RDF “graph” whose triples are distributed over the documents in \( W \).

**Example 6.** As a running example for the remainder of this paper, we assume a small Web of Linked Data \( W_{ex} = (D_{ex}, adoc_{ex}) \) consisting of seven documents.

\[
D_{ex} = \{d_A, d_B, d_C, d_D, d_E, d_F, d_P\}, \text{ with data that describes a project, denoted by IRI PrjX \in I, and people, denoted by Alice, Bob, Charlie, Dody, Eve, Suzi \in I. Figure 4 presents this data and illustrates the link graph of } W_{ex} \text{ assuming function adoc}_{ex} \text{ is given as follows:}
\]

\[
\begin{align*}
adoc_{ex}(Alice) &= d_A, & adoc_{ex}(Eve) &= d_E, \\
adoc_{ex}(Bob) &= d_B, & adoc_{ex}(Suzi) &= d_S, \\
adoc_{ex}(Charlie) &= d_C, & adoc_{ex}(PrjX) &= d_P, \\
adoc_{ex}(Dody) &= d_D, & \text{and } adoc_{ex}(u) &= \bot \\
&\text{for every other IRI } u.
\end{align*}
\]

We emphasize that the link graph, as well as the two elements \( D \) and \( adoc \), typically are not available directly to systems that aim to compute queries over the Web of Linked Data captured by \( W = (D, adoc) \). In particular, the set \( dom^D(adoc) \)—i.e., all IRIs that can be used to retrieve some document—is unknown to such systems and can only be disclosed partially (by trying to look up IRIs). This inherent lack of complete information about a queried Web of Linked Data has an impact on the feasibility of answering specific types of queries completely as we shall see in Section 6.
4. Web-aware Semantics of Property Paths

This section introduces three alternative query semantics, each of which defines an expected query result for any PP pattern over any Web of Linked Data.

4.1. Full-Web Query Semantics

As a first approach we may assume a semantics that is based on the standard evaluation function for PP patterns (cf. Definition 2) and defines expected query results in terms of all data in a queried Web of Linked Data. The following definition captures this approach, which we call a “full-Web query semantics” [18].

Definition 7. Let \( P \) be a PP pattern, \( W = (D, adoc) \) be a Web of Linked Data, and \( G_\text{all} \) be the RDF graph for which it holds that \( G_\text{all} = \bigcup_{d \in D} \text{data}(d) \). The evaluation of \( P \) over \( W \) under full-Web semantics, denoted by \( [P]^{FW}_{W} \), is defined by \( [P]^{FW}_{W} = [P]^{G_\text{all}} \).

Example 8. Recall our example Web \( W_{ex} \) (cf. Example 6 and Figure 4). The expected result of evaluating PP pattern \( P_{a} = \langle \text{Suzi knows, knows} \rangle \langle ?x \rangle \) over \( W_{ex} \) under full-Web semantics is the multiset of solution mappings \( [P_{a}]^{FW}_{W_{ex}} = \{ \{ a_{3}, a_{2}, a_{4}, a_{5} \} \} \) for which the following properties hold:

\[ \mu_{a_{1}}(\langle ?x \rangle) = \text{Charlie and } \text{card}_{a}^{FW}(\mu_{a_{1}}) = 1 \text{ (because } \text{Suzi has a “knows/knows connection” to Charlie via Alice by using triples from documents } d_{5} \text{ and } d_{6} \text{);} \]

\[ \mu_{a_{2}}(\langle ?x \rangle) = \text{Eve and } \text{card}_{a}^{FW}(\mu_{a_{2}}) = 1 \text{ (connection via Alice with triples from documents } d_{5} \text{ and } d_{6} \text{);} \]

\[ \mu_{a_{3}}(\langle ?x \rangle) = \text{Alice and } \text{card}_{a}^{FW}(\mu_{a_{3}}) = 1 \text{ (via Dody by using only triples from } d_{5} \text{);} \]

\[ \mu_{a_{4}}(\langle ?x \rangle) = \text{Suzi and } \text{card}_{a}^{FW}(\mu_{a_{4}}) = 2 \text{ (connections via Dody, see } d_{5}, \text{ and Bob, see } d_{6} \text{);} \]

\[ \mu_{a_{5}}(\langle ?x \rangle) = \text{Dody and } \text{card}_{a}^{FW}(\mu_{a_{5}}) = 1 \text{ (via Bob).} \]

We emphasize that the full-Web query semantics is mostly of theoretical interest. In practice, that is, for a Web of Linked Data \( W' = \langle D', adoc' \rangle \) that represents the “real” WWW (as deployed on the Internet), there cannot exist any system that guarantees to compute the given evaluation function \( [P]^{FW}_{W'} \) over \( W' \) using an algorithm that both terminates and returns complete query results. Our earlier work provides a formal proof of such a limitation of a full-Web query semantics for other types of SPARQL graph patterns, including triple patterns [18]. It is trivial to carry this result over to the full-Web semantics of PP patterns (i.e., Definition 7) because any PP pattern \( P = \langle \alpha, \text{path}, \beta \rangle \) with PP expression \( \text{path} \) being an IRI \( u \in \mathcal{I} \) is a triple pattern \( \langle \alpha, u, \beta \rangle \). Informally, we explain this negative result by the fact that the two structures \( D^* \) and \( adoc^* \) that capture the queried Web formally, are not available for the WWW. Consequently, to enumerate the set of all triples in \( W^* \) (denoted by \( G_\text{all} \) in Definition 7), a query execution system would have to discover all documents of the set \( D^* \); given that mapping \( adoc^* \) is not available to such a system (in particular, \( \text{dom}(adoc^*) \)—the set of all IRIs whose lookup retrieves a document—is, at best, partially known), the only guarantee to discover all documents is to look up any possible (HTTP) IRI. Since these are infinitely many [9], the enumeration process cannot terminate.

4.2. Reachability-Based Query Semantics

Given the limited practical applicability of the full-Web semantics, our earlier work introduces reachability-based semantics that restrict the scope of queries and expected results to “reachable” documents [18]. In the following, we adapt this idea for PP patterns.
Informally, a set of reachable documents of a Web of Linked Data \( W \) contains all the documents that can be reached by traversing recursively a well-defined set of data links in the link graph of \( W \). To specify what data links belong to such a set, we introduce the notion of a reachability criterion [18], which we define formally as a function \( c : \mathcal{T} \times \mathcal{I} \times \mathcal{P} \to \{ \text{true}, \text{false} \} \) where \( \mathcal{P} \) denotes the infinite set of all PP patterns (and, as introduced before, \( \mathcal{T} \) and \( \mathcal{I} \) are the sets of all triples and all IRIs, respectively). Then, given such a reachability criterion, we define reachability of documents as follows.

**Definition 9.** Let \( P \) be a PP pattern, let \( S \subseteq \mathcal{I} \) be a finite set of IRIs (which serve as a seed), and let \( c \) be a reachability criterion, we define reachability of documents as follows.

1. There exists an IRI \( u \in S \) such that \( \text{adoc}(u) = d \) (in which case we call \( d \) a “seed document”); or
2. There exists (another) document \( d' \in D \), a triple \( t \), and an IRI \( u \) such that
   (a) \( d' \) is \( (S, c, P) \)-reachable in \( W \),
   (b) \( t \in \text{data}(d') \),
   (c) \( u \in \text{iris}(t) \),
   (d) \( c(t, u, P) = \text{true} \), and
   (e) \( \text{adoc}(u) = d \).

Notice how the second condition restricts the notion of reachability by ignoring any data link that does not satisfy the given reachability criterion. In earlier work we define several concrete reachability criteria [18], including \( c_{\text{All}} \) that, for each tuple \((t, u, P)\in\mathcal{T}\times\mathcal{I}\times\mathcal{P}\), is defined by \( c_{\text{All}}(t, u, P) = \text{true} \); hence, \( c_{\text{All}} \) does not place any restrictions on data links.

Another, more restrictive criterion that is commonly used in practice [19,38], is \( c_{\text{Match}} \) [18]; this criterion ignores all data links that do not match any triple pattern contained in the given SPARQL query. While our earlier formal definition of \( c_{\text{Match}} \) assumes that SPARQL queries are constructed from triple patterns [18], we may adapt the idea of this criterion for the PP-based patterns in this paper and define a corresponding reachability criterion that we call \( \text{cppMatch} \).

**Definition 10.** For any triple \( t = (s, p, o) \), IRI \( u \), and PP pattern \( P \), \( \text{cppMatch}(t, u, P) = \text{true} \) if and only if \( p \) is an IRI that is mentioned in the PP expression of PP pattern \( P \) except for those IRIs that appear only in subexpressions of the forms \( !\{u_1\ | \ldots | u_n\} \).

**Example 11.** By using our previous example pattern \( P_a = \langle \text{Suzi, knows/knows, ?x} \rangle \) and \( S_{\text{ex}} = \{ \text{Suzi} \} \), the following documents are \( (S_{\text{ex}}, \text{cppMatch}, P_a) \)-reachable in our example Web \( W_{\text{ex}} \) (cf. Example 6 and Figure 4): \( d_S, d_K, d_C, \) and \( d_E \). If we consider the less restrictive reachability criterion \( c_{\text{All}} \) instead, then we have these four documents and, additionally, \( d_P \) and \( d_D \) as being \( (S_{\text{ex}}, c_{\text{All}}, P_a) \)-reachable in \( W_{\text{ex}} \) (i.e., all but \( d_B \)).

Given the notion of reachability criteria, we define a family of reachability-based semantics for PP patterns:

**Definition 12.** Let \( P \) be a PP pattern, let \( S \subseteq \mathcal{I} \) be a finite set of IRIs, and let \( c \) be a reachability criterion. Furthermore, let \( W \) be a Web of Linked Data, let \( D_{\text{w}} \) be the set of all documents that are \( (S, c, P) \)-reachable in \( W \), and let \( G_R \) be the RDF graph for which it holds that \( G_R = \bigcup_{D \in D_{\text{w}}} \text{data}(d) \). Then, the \( S \)-seeded evaluation of \( P \) over \( W \) under \( c \)-semantics, denoted by \( \llbracket P \rrbracket_{W}^{c,S} \), is defined by \( \llbracket P \rrbracket_{W}^{c,S} = \llbracket P \rrbracket_{G_R} \) where \( \llbracket P \rrbracket_{G_R} \) uses the standard evaluation function for PP patterns (cf. Definition 2).

**Example 13.** Consider \( P_a = \langle \text{Suzi, knows/knows, ?x} \rangle \) and \( S_{\text{ex}} = \{ \text{Suzi} \} \), then, under \( c_{\text{All}} \)-semantics, we have \( \llbracket P_a \rrbracket_{W_{\text{ex}}}^{c_{\text{All}},S_{\text{ex}}} = \{ \langle \mu_{a_1}, \mu_{a_2}, \mu_{a_3}, \mu_{a_4} \rangle \} \), \( \text{card}_{a}^{c_{\text{All}},S_{\text{ex}}} = \{ \langle \mu_{a_1} \rangle \} \) as in Example 8 and \( \text{card}_{a}^{c_{\text{All}},S_{\text{ex}}} = \{ \langle \mu_{a_1} \rangle \} \) for all \( i \in \{ 1, 2, 3, 4 \} \). Note that solution mapping \( \mu_{a_5} \) (cf. Example 8) is not a solution in this case because computing it requires triples from document \( d_B \), but \( d_B \) is not \( (S_{\text{ex}}, c_{\text{All}}, P_a) \)-reachable in \( W_{\text{ex}} \) (cf. Example 11); due to the same reason we have \( \text{card}_{a}^{c_{\text{Match}},S_{\text{ex}}} = \{ \langle \mu_{a_1} \rangle \} \) (under full-Web semantics it is \( \text{card}_{a}^{c_{\text{Match}},S_{\text{ex}}} = \{ \langle \mu_{a_1} \rangle \} \) = 2; cf. Example 8).

**Example 14.** Under \( \text{cppMatch} \)-semantics, we only expect the following result for \( P_a \) (and \( S_{\text{ex}} \)) over \( W_{\text{ex}} \):

\[
\llbracket P_a \rrbracket_{W_{\text{ex}}}^{\text{cppMatch},S_{\text{ex}}} = \{ \langle \mu_{a_1}, \mu_{a_2} \rangle \} \text{ and } \text{card}_{a}^{\text{cppMatch},S_{\text{ex}}} = \{ \langle \mu_{a_1}, \mu_{a_2} \rangle \} .
\]

As mentioned in Example 8, solution mapping \( \mu_{a_3} \) requires document \( d_D \), which is not \( (S_{\text{ex}}, \text{cppMatch}, P_a) \)-reachable in \( W_{\text{ex}} \) (cf. Example 11); similarly, for \( \mu_{a_4} \).

### 4.3. Context-Based Query Semantics

Reachability-based query semantics as introduced in the previous section impose a clear conceptual separation between navigation over the link graph of a queried Web of Linked Data—which serves the purpose of discovering and retrieving reachable documents—and standard PP-based navigation over the data obtained from all reachable documents. That is, there exists no correlation between paths of triples that match PP expressions and paths of data links that connect reachable documents to seed documents.
At this point it is interesting to also explore an alternative approach in which navigation on the link graph correlates with PP patterns in queries. To this end, we introduce another semantics that interprets PP patterns as a language for navigation over Linked Data on the WWW (i.e., along the lines of earlier navigational languages for Linked Data such as NautilOD [10]). We refer to this semantics as context-based.

The main idea of this query semantics is to restrict the scope of searching for any next triple of a potentially matching path to specific data within specific documents on the queried Web of Linked Data.

To formalize these restrictions we introduce the notion of a context selector. Informally, for each IRI that can be used to retrieve a document, the context selector returns a specific subset of the data within that document; this subset contains only those triples that have the given IRI as their subject (such a subset of triples resembles Harth and Speiser’s notion of “subject authoritative triples” [16]). Formally, for any Web of Linked Data \(W = \langle D, adoc \rangle\), the context selector of \(W\) is a function \(C^W: (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L} \cup \mathcal{V}) \rightarrow 2^T\) that, for every IRI \(u \in \mathcal{I}\) with \(u \in \text{dom}^W(adoc)\), is defined by

\[
C^W(u) = \{ (s, p, o) \in \text{data}(adoc(u)) \mid u = s \},
\]

and for any other \(\gamma \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L} \cup \mathcal{V}) \setminus \text{dom}^W(adoc)\) we have \(C^W(\gamma) = \emptyset\) (by extending the definition of \(C^W\) to handle any such \(\gamma\), we can simplify the following formalization of the context-based query semantics).

Informally, the context-based semantics uses the notion of a context selector to restrict the scope of PP patterns over a Web of Linked Data as follows. Assume a sequence of triples \(\langle s_1, p_1, o_1 \rangle, \ldots, \langle s_k, p_k, o_k \rangle\) that presents a path that already matches a sub-expression of a given PP expression. Under the previously defined reachability-based query semantics, the next triple for such a path can be searched for in any reachable document in the queried Web of Linked Data \(W\). By contrast, under the context-based query semantics that we formalize in the following Definition 15, the next triple has to be searched for only in \(C^W(o_k)\).

**Definition 15.** Given a PP pattern \(P\) and a Web of Linked Data \(W = \langle D, adoc \rangle\), the evaluation of \(P\) over \(W\) under context-based semantics, denoted by \(\llbracket P \rrbracket_{W,ctx}^{ctx}\), is a multiset of solution mappings \((\Omega, \text{card})\) that is defined recursively as given in Figure 5.

Note how Definition 15 uses the context selector to restrict the data that has to be searched to find matching triples (e.g., consider the first line in Figure 5).

**Example 16.** Coming back to the example PP pattern \(P_a = \langle \text{suzi, knows}/\text{knows}, ?x \rangle\), and \(W_{ex}\) (cf. Example 6 and Figure 4), under the context-based semantics we obtain \(\llbracket P_a \rrbracket_{W_{ex},ctx}^{ctx} = \langle \{ \mu_{a1} \}, \text{card}_{a}^{ctx}(\mu_{a1}) \rangle\) with \(\mu_{a1}\) as before (cf. Example 8) and \(\text{card}_{a}^{ctx}(\mu_{a1}) = 1\).

There are two points worth emphasizing regarding Definition 15: First, we define the context-based semantics such that it resembles the standard semantics of PP patterns in Section 3.1 as close as possible. To this end, the part of our definition that covers PP patterns of the form \(\langle \alpha, \text{path}, \beta \rangle\) also uses auxiliary functions, namely, ALPW1 and ALPW2 (cf. Figure 6).

These functions evaluate the sub-expression \(\text{path}\) recursively over the queried Web of Linked Data (instead of using a fixed RDF graph as done in the standard semantics in Figure 2). Second, the two base cases with a variable in the subject position (i.e., the third and the sixth case in Figure 5) require an enumeration of all IRIs. Such a requirement is necessary to both, remain consistent with the standard semantics and preserve commutativity of operators that can be defined on top of PP patterns (such as the \(\text{AND}\) operator in SPARQL; cf. Section 5).

However, due to this requirement, there exist PP patterns whose (complete) evaluation under context-based semantics is infeasible when querying the WWW. The following example describes such a case.

**Example 17.** Consider the following PP pattern \(P_{E17}\), which retrieves the IRIs of people that know Tim:

\[
P_{E17} = \langle ?v, \text{knows}, \text{Tim} \rangle.
\]

Under context-based semantics, any IRI \(u\) can be used to generate a correct solution mapping for the pattern as long as a lookup of that IRI results in retrieving a document whose data contains the triple \(\langle u, \text{knows}, \text{Tim} \rangle\). While, for any Web of Linked Data that is finite, there exists only a finite number of such IRIs, determining these IRIs and guaranteeing completeness requires enumerating the infinite set of all possible IRIs and checking each of them—unless one knows the complete (and finite) subset of all IRIs that can be used to retrieve some document, which, due to the infiniteness of possible HTTP-scheme IRIs, cannot be achieved for the WWW.

It is not difficult to see that the issue illustrated in the example exists for any triple pattern that has a variable in the subject position. On the other hand, triple patterns whose subject is an IRI do not have this issue. However, having an IRI in the subject position is
A question that arises is whether there exists a (decidably) property of PP patterns that can be used to distinguish between patterns that do not have this issue (i.e., evaluating them over any Web of Linked Data is feasible under the context-based semantics).
and those that do. Another question is whether any of the aforementioned reachability-based semantics has a similar problem, and, more generally, how do these semantics compare to the context-based semantics?

We come back to these questions in Sections 6 and 7, after introducing the more general case of PP-based SPARQL queries in the next section.

5. PP-based SPARQL Queries for the Web

After considering PP patterns in isolation, we now turn to a more expressive fragment of SPARQL that embeds PP patterns as the basic building block and uses additional operators on top. In this section, we define the resulting PP-based SPARQL queries; we specify their syntax and formalize Web-aware semantics that extend the above defined semantics of PP patterns.

By using the algebraic syntax of SPARQL [30], we define a graph pattern recursively as follows:

- Any PP pattern \( \langle \alpha, \text{path}, \beta \rangle \) is a graph pattern.
- If \( P_1 \) and \( P_2 \) are graph patterns, then so are \( (P_1 \text{ AND} P_2) \), \( (P_1 \text{ UNION} P_2) \), and \( (P_1 \text{ OPT} P_2) \).

For any graph pattern \( P \), we write \( \text{vars}(P) \) to denote the set of all variables in \( P \); that is, if \( P \) is a PP pattern \( \langle \alpha, \text{path}, \beta \rangle \), we have \( \text{vars}(P) = \{ \alpha, \beta \} \cap X \), and if \( P \) is of the form \( (P_1 \text{ AND} P_2) \), \( (P_1 \text{ UNION} P_2) \), or \( (P_1 \text{ OPT} P_2) \), we have \( \text{vars}(P) = \text{vars}(P_1) \cup \text{vars}(P_2) \).

Example 18. An example of a graph pattern that combines two PP patterns using the OPT operator is given as follows: \( \langle \text{Tim}, \text{knows/known}, ?, p \rangle \text{ OPT} \langle ?, \text{name}, ?n \rangle \) This pattern retrieves persons known by acquaintances of Tim and, if available, the names of these persons.

By using PP patterns as the basic building block of graph patterns, we can readily carry over any of the above defined query semantics to graph patterns. To this end, let \( \Sigma \) be a set of symbols that denote these semantics; in particular, we have \( \Sigma^\varphi \subseteq \Sigma \) that denotes the full-Web semantics (cf. Section 4.1), \( \Sigma^\varphi(c, S) \subseteq \Sigma \) denotes the (reachability-based-\( c \)-semantics with a set \( S \) of seed IRIs (cf. Section 4.2), and \( \text{ctx} \subseteq \Sigma \) denotes the context-based semantics (cf. Section 4.3). We extend these semantics to cover graph patterns as follows.

\[ \text{Definition 19. Let } P \text{ be a graph pattern and let } W \text{ be a Web of Linked Data. For any } \varphi \in \Sigma, \text{ the evaluation of } P \text{ over } W \text{ under the semantics denoted by } \varphi \text{ is a multiset of solution mappings, denoted by } [P]^\varphi_W, \text{ that is defined recursively as follows:}^2 \]

\[ - \text{ If } P \text{ is a PP pattern } \langle \alpha, \text{path}, \beta \rangle, \text{ then } [P]^\varphi_W \text{ is defined in the } \varphi \text{-specific subsection of Section 4.} \]

\[ - \text{ If } P \text{ is of the form } (P_1 \text{ AND} P_2), \text{ then } [P]^\varphi_W = [P_1]^\varphi_W \times [P_2]^\varphi_W. \]

\[ - \text{ If } P \text{ is of the form } (P_1 \text{ UNION} P_2), \text{ then } [P]^\varphi_W = [P_1]^\varphi_W \cup [P_2]^\varphi_W. \]

\[ - \text{ If } P \text{ is of the form } (P_1 \text{ OPT} P_2), \text{ then } [P]^\varphi_W = ([P_1]^\varphi_W \times [P_2]^\varphi_W) \cup ([P_1]^\varphi_W \setminus [P_2]^\varphi_W). \]

6. Web-Safeness

Given the different semantics for evaluating (PP-based) graph patterns over a Web of Linked Data, we now study formally whether such evaluations are possible in practice over Linked Data on the WWW.

To this end, we first recall from Section 4.1 that, under full-Web semantics, evaluating PP patterns over the WWW is not possible in practice because, for the tuple \( W = \langle D, \text{adoc} \rangle \) with which we formalize the notion of Linked Data on the WWW, the sets \( D \) and \( \text{dom}^2(\text{adoc}) \) cannot be assumed to be available completely to any algorithm [18]. Without complete knowledge of these two sets, an algorithm designed to answer PP patterns completely under full-Web semantics would have to enumerate the infinite set of all possible (HTTP-scheme) IRIs and look up each of them.

Based on this observation, we define a notion of Web-safeness of graph patterns; with this notion we capture whether it is possible for a graph pattern to be evaluated completely over Linked Data on the WWW under a given semantics.

\[ \text{Definition 20. For any } \varphi \in \Sigma, \text{ a graph pattern } P \text{ under the semantics denoted by } \varphi \text{ is Web-safe if there exists an algorithm that, for any finite Web of Linked Data } W = \langle D, \text{adoc} \rangle, \text{ has the following properties:} \]

1. The algorithm computes \([P]^\varphi_W\).
2. During its execution, the algorithm looks up only a finite number of IRIs (that is, conceptually, the algorithm invokes function adoc only a finite number of times).

\[ ^2 \text{Note that the definition uses the algebra defined in Figure 1.} \]
3. Neither the set \( D \) nor the set \( \text{dom}^k(\text{adoc}) \) is required as input for the algorithm (hence, the algorithm does not require any a priori information about \( W \)).

Unsurprisingly, as already discussed in Section 4.1, it follows from the results in our earlier work [18] that, under full-Web semantics, none of the graph patterns considered in this paper is Web-safe.

In the following, we study Web-safeness of graph patterns under the other Web-aware query semantics.

6.1. Web-Safeness of Reachability-Based Semantics

Independent of what reachability criterion (and seed IRIs) one chooses, for every reachability-based semantics we can show the following positive result.

**Theorem 21.** Given an arbitrary reachability criterion \( c \) and any finite set \( S \subseteq \mathcal{I} \) of IRIs, every graph pattern is Web-safe under \( c \)-semantics with \( S \) as seed IRIs.

As a basis to prove Theorem 21, we first focus on PP patterns, for which we show the following lemma.

**Lemma 22.** Given an arbitrary reachability criterion \( c \) and any finite set \( S \subseteq \mathcal{I} \) of IRIs, every PP pattern is Web-safe under \( c \)-semantics with \( S \) as seed IRIs.

**Proof (Lemma 22).** We prove the lemma by providing Algorithm 1. It is easily verified that this algorithm has the desired properties (as listed in Definition 20). Note that the execution of this algorithm consists of two consecutive phases: a data retrieval phase (lines 1 to 12) and a standard result computation phase (line 13). During the data retrieval phase the algorithm incrementally discovers all documents that are \((S, c, P)\)-reachable in the queried Web, and collects their data in RDF graph \( G_R \). The second condition in line 11 ensures that any other document is ignored during the data retrieval phase. Hence, when the execution of the algorithm reaches line 13, we have \( G_R = \bigcup_{d \in D_R} \text{data}(d) \) where \( D_R \) is the set of all \((S, c, P)\)-reachable documents. Due to the finiteness of the queried Web of Linked Data, both \( D_R \) and \( G_R \) are finite. Therefore, there exists a finite upper bound on the number of different IRIs that the algorithm has to look up; in the worst case this upper bound is the number of all IRIs in the final version of \( G_R \) (in practice, the upper bound may be smaller depending on the reachability criterion \( c \)). The existence of this upper bound and the first condition in line 11 ensure that the data retrieval phase terminates.

Given Lemma 22, it is trivial to prove Theorem 21.

**Algorithm 1** Computation of the \( S \)-seeded evaluation of a PP pattern \( P \) over any Web of Linked Data under \( c \)-semantics (where \( S \subseteq \mathcal{I} \) is a finite set of IRIs and \( c \) is a reachability criterion).

\[
\begin{align*}
1: & \ G_R := \emptyset \quad // \text{an initially empty RDF graph} \\
2: & \ \text{Visited} := \emptyset \quad // \text{an initially empty set of IRIs} \\
3: & \text{Create a list of IRIs called } \text{Open} \text{ and add every IRI } u \in S \text{ to this list (in an arbitrary order)} \\
4: & \text{while } \text{Open} \text{ is not empty do} \\
5: & \quad \text{Remove the first IRI, say } u, \text{ from } \text{Open}, \text{ add this IRI to } \text{Visited}, \text{ and look up this IRI} \\
6: & \quad \text{if the lookup of IRI } u \text{ results in retrieving a document, say } d, \text{ and } d \text{ contains triples then} \\
7: & \quad \quad G := \text{the set of triples in } d \quad (\text{use a fresh set of blank node identifiers when parsing } d) \\
8: & \quad \quad \text{Add } G \text{ to } G_R \text{ (i.e., } G_R := G_R \cup G) \\
9: & \quad \quad \text{for all } t \in G \text{ do} \\
10: & \quad \quad \quad \text{for all } u' \in \text{iris}(t) \text{ do} \\
11: & \quad \quad \quad \quad \text{if } u' \notin \text{Visited} \text{ and } c(t, u', P) = \text{true then} \\
12: & \quad \quad \quad \quad \quad \text{Add } u' \text{ to } \text{Open} \\
13: & \quad \quad \text{Compute the query result } [P]_{G_R} \text{ (by using an arbitrary algorithm that implements the standard SPARQL evaluation function for PP patterns)} \\
14: & \quad \text{return } [P]_{G_R} \\
\end{align*}
\]

**Proof (Theorem 21).** Theorem 21 is a direct consequence of Definition 19 and Lemma 22. That is, given multisets of solution mappings computed for PP patterns, combining such multisets as per the algebra operators does not require any more URI lookups (or any other kind of access to the queried Web of Linked Data) and can be done by any algorithm that implements these algebra operators.

We emphasize that, while Algorithm 1 is sufficient for proving Lemma 22 and, thus, Theorem 21, it is perhaps not a very efficient algorithm to use in practice. Systems might instead implement traversal-based execution approaches to evaluate PP patterns under reachability-based semantics [19,38]; the processing of IRIs from the Open list (used in the algorithm) can be parallelized by a multi-threaded implementation; additionally, assuming a suitable invalidation policy, documents may be cached and reused for later queries [17].

6.2. Web-Safeness of Context-Based Semantics

After finding that under any reachability-based semantics all graph patterns are Web-safe, we now come back to the context-based semantics for which we know from Example 17 that Web-safeness cannot be assumed in general. We begin our analysis by providing the following example, which extends Example 17.
Example 23. Consider the following graph pattern:

\[ P_{E23} = \{(\text{Bob}, \text{knows}, ?v) \text{ AND } (?v, \text{knows}, \text{Tim})\}. \]

The right sub-pattern \( P_{E17} = \{(?v, \text{knows}, \text{Tim})\} \) is not Web-safe because evaluating it completely over the WWW is not possible under context-based semantics (cf. Example 17). However, the larger pattern \( P_{E23} \) is Web-safe under context-based semantics: A possible algorithm may first evaluate the left sub-pattern, \( \langle \text{Bob}, \text{knows}, ?v \rangle \), which is possible because it requires the lookup of a single IRI only (the IRI Bob). Thereafter, the evaluation of the right sub-pattern \( P_{E17} \) can be reduced to looking up a finite number of IRIs only, namely the IRIs bound to variable \(?v\) in solution mappings obtained in the first step for the left sub-pattern. Although any other IRI, say \( u^* \), might also be used to discover triples for \( P_{E17} \), each of these triples has IRI \( u^* \) as its subject (which is a consequence of restricting retrieved data based on the context selector introduced in Section 4.3). Therefore, possible solution mappings resulting from such triples cannot be compatible with any solution for the left sub-pattern and, thus, do not satisfy the join condition established by the semantics of \texttt{AND} in pattern \( P_{E23} \).

The example illustrates that some graph patterns are Web-safe under context-based semantics even if some of their sub-patterns are not. Consequently, we are interested in a decidable property that enables us to identify Web-safe patterns under context-based semantics, including those whose sub-patterns are not Web-safe.

Buil-Aranda et al. study a similar problem in the context of SPARQL federation where graph patterns of the form \((\text{SERVICE} ?v P)\) are allowed [7]. For such a pattern \( P_S = (\text{SERVICE} ?v P) \), variable \(?v\) ranges over a possibly large set of IRIs, each of which represents the address of a (remote) SPARQL service that needs to be called to assemble the complete result of \( P_S \). However, many service calls may be avoided if \( P_S \) is embedded in a larger graph pattern that allows for an evaluation during which \(?v\) can be bound before evaluating \( P_S \). To identify such cases, Buil-Aranda et al. introduce a notion of strongly boundedness of variables in graph patterns and use it to show a notion of safeness for the evaluation of patterns like \( P_S \) within larger graph patterns. The idea behind the notion of strongly bound variables has already been used in earlier work (e.g., “certain variables” [34], “output variables” [37]), and it is tempting to adopt it for our problem. To this end, we first define the notion of strongly bound variables for our PP-based graph patterns:

Definition 24. The set of strongly bound variables in a graph pattern \( P \), denoted by \( \text{sbvars}(P) \), is defined recursively as follows (recall that \( \text{vars}(P) \) is the set of all variables in \( P \)):

- If \( P \) is a PP pattern, then \( \text{sbvars}(P) = \text{vars}(P) \).
- If \( P \) is of the form \((P_1 \text{ AND } P_2)\), then \( \text{sbvars}(P) = \text{sbvars}(P_1) \cup \text{sbvars}(P_2) \).
- If \( P \) is of the form \((P_1 \text{ UNION } P_2)\), then \( \text{sbvars}(P) = \text{sbvars}(P_1) \cap \text{sbvars}(P_2) \).
- If \( P \) is of the form \((P_1 \text{ OPT } P_2)\), then \( \text{sbvars}(P) = \text{sbvars}(P_1) \).

Given the definition of strongly bound variables, we observe that one cannot identify Web-safe graph patterns by using only this notion of strong boundedness.

Example 25. Consider graph pattern \( P_{E23} \) from Example 23. We know that (i) \( P_{E23} \) is Web-safe and that (ii) \( \text{vars}(P_{E23}) = \{?v\} \) and also \( \text{sbvars}(P_{E23}) = \{?v\} \). Then, one might hypothesize that a graph pattern \( P \) is Web-safe if \( \text{sbvars}(P) = \text{vars}(P) \). However, the PP pattern \( P_{E17} = \{(?v, \text{knows}, \text{Tim})\} \) disproves such a hypothesis because, even if \( \text{sbvars}(P_{E17}) = \text{vars}(P_{E17}) \), pattern \( P_{E17} \) is not Web-safe (cf. Example 17). Alternatively, one might also hypothesize that if a graph pattern \( P \) is Web-safe, then \( \text{sbvars}(P) = \text{vars}(P) \). However, this hypothesis can be disproved by using pattern \( P_{E25} = ((\text{Bob}, \text{knows}, ?x) \text{ OPT } (?x, \text{knows}, ?y)) \). It can easily be verified that \( P_{E25} \) is Web-safe (e.g., it is not difficult to adjust the algorithm for pattern \( P_{E23} \) in Example 23 accordingly). However, in contradiction to the hypothesis we have \( \text{sbvars}(P_{E25}) \neq \text{vars}(P_{E25}) \).

We conjecture the following reason why strongly boundedness cannot be used directly for our problem. Consider the types of graph patterns that combine two sub-patterns (by using operators such as \texttt{AND}). For such a pattern, the sets of strongly bound variables of its sub-patterns are defined independent from each other, whereas the algorithm outlined in Example 23 leverages a specific relationship between sub-patterns. More precisely, the algorithm leverages the fact that the same variable that is the subject of the right sub-pattern is also the object of the left sub-pattern.

Based on this observation, we introduce the notion of conditionally bound variables, which is based on particular relationships between sub-patterns due to which the result of one sub-pattern may be used to evaluate another sub-pattern in a more well-behaved
manner (along the lines of Example 23). This notion shall turn out to be suitable for our case.

**Definition 26.** Let \( X \subseteq V \) be a set of variables. The conditionally bound variables in a graph pattern \( P \) w.r.t. \( X \), denoted by \( \text{cbvars}(P \mid X) \), is a subset of the variables in \( P \) (i.e., \( \text{cbvars}(P \mid X) \subseteq \text{vars}(P) \)) that is defined recursively as given in Table 1.

Example 27. The conditionally bound variables in the PP pattern \( P_{E17} = \langle \text{Bob} \text{, knows} \text{, Tim} \rangle \) w.r.t. the empty set of variables can be determined based on line 2 in Table 1, and we obtain: \( \text{cbvars}(P_{E17} \mid \emptyset) = \emptyset \). However, if we use the set \( \{ ?v \} \) instead, then, by line 1 in Table 1, we obtain: \( \text{cbvars}(P_{E17} \mid \{ ?v \}) = \{ ?v \} \).

Example 28. As another example consider the graph pattern \( P_{E23} = \langle \langle \text{Bob} \text{, knows} \rangle \text{AND} \langle ?u \text{, knows} \rangle \rangle \) for which we obtain \( \text{cbvars}(P_{E23} \mid \emptyset) = \emptyset \) by using line 10 in Table 1 and the following facts:

1. \( \text{cbvars}(\langle \text{Bob} \text{, knows} \rangle \mid \emptyset) = \{ ?v \} \),
2. \( \text{cbvars}(\langle ?u \text{, knows} \rangle \mid \{ ?v \}) = \{ ?v \} \),
3. \( \text{cbvars}(\langle ?u \text{, knows} \rangle \mid \emptyset) = \{ ?v \} \).

We note that for the pattern \( P_{E17} \), which is not Web-safe under context-based semantics (as discussed in Example 17), we have \( \text{cbvars}(P_{E17} \mid \emptyset) \neq \text{vars}(P_{E17}) \), whereas for the pattern \( P_{E23} \), which is Web-safe under context-based semantics (cf. Example 23), we have \( \text{cbvars}(P_{E23} \mid \emptyset) = \text{vars}(P_{E23}) \). This example seems to suggest that, if all variables of a graph pattern are conditionally bound w.r.t. the empty set of variables, then the graph pattern is Web-safe under context-based semantics. The following result verifies this hypothesis.

**Theorem 29.** A graph pattern \( P \) is Web-safe under context-based semantics if \( \text{cbvars}(P \mid \emptyset) = \text{vars}(P) \).

Before proving Theorem 29 in the remainder of this section, we emphasize the following observation.

**Note 30.** Due to the recursive nature of Definition 26, the condition \( \text{cbvars}(P \mid \emptyset) = \text{vars}(P) \) (as used in Theorem 29) is decidable for any graph pattern \( P \).

To prove Theorem 29 we aim to provide an algorithm that evaluates graph patterns recursively by passing (intermediate) solution mappings to recursive calls. To capture the desired results of each recursive call formally, we introduce a special evaluation function for a graph pattern \( P \) over a Web of Linked Data \( W \) that takes a solution mapping \( \mu \) as input and returns only the solutions of \( P \) over \( W \) that are compatible with \( \mu \) (recall from Section 3.1 that the compatibility of two solution mappings, \( \mu_1 \) and \( \mu_2 \), is denoted by \( \mu_1 \sim \mu_2 \)).

**Definition 31.** Let \( P \) be a graph pattern, let \( W \) be a Web of Linked Data, and let \( (\Omega, \text{card}) = \llbracket P \rrbracket_W^{\text{ctx}} \). Given a solution mapping \( \mu \), the \( \mu \)-restricted evaluation of \( P \) over \( W \) under context-based semantics, denoted by \( \llbracket P \mid \mu \rrbracket_W^{\text{ctx}} \), is the multiset of solution mappings \( (\Omega', \text{card}') \) with \( \Omega' = \{ \mu' \in \Omega \mid \mu' \sim \mu \} \) and \( \text{card}' \) is the restriction of \( \text{card} \) to \( \Omega' \), i.e., for every solution mapping \( \mu' \in \Omega' \) we have \( \text{card}'(\mu') = \text{card}(\mu)^{\Omega'} \).

The following lemma shows the existence of the aforementioned recursive algorithm.

**Lemma 32.** Let \( P \) be a graph pattern and \( \mu\text{bec} \) be a solution mapping. If \( \text{cbvars}(P \mid \text{dom}(\mu)\text{bec}) = \text{vars}(P) \), then there exists an algorithm that, for any finite Web of Linked Data \( W = (D, \text{adoc}) \), has the following three properties:

1. The algorithm computes \( \llbracket P \mid \mu\text{bec} \rrbracket_W^{\text{ctx}} \).
2. During its execution, the algorithm looks up only a finite number of IRIs (that is, conceptually, the algorithm invokes function \( \text{adoc} \) only a finite number of times).
3. Neither the set \( D \) nor the set \( \text{dom}(\mu) \text{bec} \) is required as input for the algorithm (hence, the algorithm does not require any a priori information about \( W \)).

Before proving the lemma (and Theorem 29), we point out two important properties of Definition 31. First, it is easily seen that, for any graph pattern \( P \) and Web of Linked Data \( W \), \( \llbracket P \mid \mu_0 \rrbracket_W^{\text{ctx}} = \llbracket P \rrbracket_W^{\text{ctx}} \), where \( \mu_0 \) is the empty solution mapping with \( \text{dom}(\mu_0) = \emptyset \). Consequently, given an algorithm, say \( A \), that, for \( P \) and \( \mu_0 \), has the properties of the algorithm described by Lemma 32, a trivial algorithm that can be used to prove Theorem 29 may simply call algorithm \( A \) and return the result of this call (a more detailed discussion of this approach follows in the proof of Theorem 29 below). Second, for any PP pattern \( (\alpha, \text{path}, \beta) \) and Web of Linked Data \( W \), if \( \alpha \) is a variable and \( \text{path} \) is a PP expression that corresponds to one of the first two cases in the grammar in Section 3.1 (i.e., the two base cases), then \( \llbracket P \mid \mu \rrbracket_W^{\text{ctx}} \) is empty for every solution mapping \( \mu \) that binds (variable) \( \alpha \) to a literal or a blank node. Formally, we show the latter as follows.

**Lemma 33.** Let \( ?v \in \mathcal{V} \) be a variable, \( P \) be a PP pattern of the form \( \langle ?v, u, \beta \rangle \) or \( \langle ?v, \langle !u_1, \ldots, !u_n \rangle, \beta \rangle \) with \( u, u_1, \ldots, u_n \in \mathcal{L} \), and \( \mu \) be a solution mapping. If \( ?v \in \text{dom}(\mu) \) and \( \mu(?v) \in (B \cup L) \), then, for any Web of Linked Data \( W \), \( \llbracket P \mid \mu \rrbracket_W^{\text{ctx}} \) is the empty multiset (of solution mappings).
If $P$ is: 

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) $(\alpha, u, \beta)$ or $(\alpha, [(u_1</td>
<td>\ldots</td>
<td>u_n), \beta)$ such that $\alpha \in {\mathcal{T} \cup \mathcal{L}}$ or $\alpha \notin X$</td>
</tr>
<tr>
<td>2) $(\alpha, u, \beta)$ or $(\alpha, [(u_1</td>
<td>\ldots</td>
<td>u_n), \beta)$ such that $\alpha \notin {\mathcal{T} \cup \mathcal{L}}$ and $\alpha \notin X$</td>
</tr>
<tr>
<td>3) $(\alpha, (\text{path})^*, \beta)$ such that $\alpha \in V$ and $\beta \notin V$</td>
<td>$\text{vars}({\beta, (\text{path})^*, \alpha \mid X})$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>4) $(\alpha, (\text{path})^*, \beta)$ such that $\alpha \notin V$ or $\beta \in V$, and for any two variables $?x, ?y \in V$ it holds that $\text{vars}({?x, \text{path}, ?y \mid {?x}}) = {?x, ?y}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>5) $(\alpha, (\text{path})^*, \beta)$ such that none of the above</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

Table 1

Cases of the recursive definition of the conditionally bound variables of a graph pattern $P$ w.r.t. a set of variables $X \subseteq V$.

**Proof (Lemma 33).** Recall that for any IRI $u$ and any Web of Linked Data $W$, every triple in the context $\mathcal{C}^W(u)$ has IRI $u$ as its subject. As a consequence, for any Web of Linked Data $W$, every solution mapping in $\mathbb{P}^W_{\mathcal{C}^W}$ binds variable $?v$ to some IRI (and not to a literal or a blank node); that is, formally, for every $?v \in \mathbb{P}^W_{\mathcal{C}^W}$ we have $\mu(?v) \in \mathcal{I}$. Therefore, if $\mu \in \mathbb{P}^W_{\mathcal{C}^W}$ and $\mu(?v) \in (\mathcal{B} \cup \mathcal{L})$, then none of the solution mappings in $\mathbb{P}^W_{\mathcal{C}^W}$ is compatible with $\mu$, and, thus, $\mathbb{P}^W_{\mathcal{C}^W}$ is empty. □

We use Lemma 33 to prove Lemma 32 as follows.

**Proof idea (Lemma 32).** We prove Lemma 32 by induction on the possible structure of graph pattern $P$. To this end, we provide Algorithm 2 and show that this (recursive) algorithm has the desired properties for any possible graph pattern (i.e., any case of the induction, including the base case). In this paper we focus on a fragment of the algorithm and highlight essential properties thereof. This fragment covers the base case (lines 1-11) and one pivotal case of the induction step, namely, graph patterns of the form $(P_1 \cup P_2)$. The complete version of the algorithm and the full proof can be found in our technical report [22].

For the base case (i.e., PP patterns of the form $(\alpha, u, \beta)$ or $(\alpha, [(u_1 | \ldots | u_n), \beta)$), Algorithm 2 looks up at most one IRI (cf. lines 2-5). The crux of showing that the returned result is sound and complete is Lemma 33 and the fact that a triple $(s, p, o)$ with $s \in \mathcal{I}$ can be found only in the context $\mathcal{C}^W(s)$.

For PP patterns of the form $(P_1 \cup P_2)$ consider lines 57-72. For sub-patterns $P_1$ and $P_2$ as used in this part of the algorithm, we may use Definition 26 to show that (i) $\text{vars}(P_1 | \text{dom}(\mu^\text{in})) = \text{vars}(P_1)$ and (ii) $\text{vars}(P_2 | \text{dom}(\mu^\text{in}) \cup \text{dom}(\mu)) = \text{vars}(P_2)$ for all $\mu \in \Omega^P$. Therefore, by induction, any recursive call of the algorithm in line 61 and line 63 looks up a finite number of IRIs and returns the expected (sound and complete) result; that is, $(\Omega^P_1, \text{card}^P_1) = \mathbb{P}^W_{\mathcal{C}^W}[\mu^\text{in}]$ and $(\Omega^P_2, \text{card}^P_2) = \mathbb{P}^W_{\mathcal{C}^W}[\mu^\text{in} \cup \mu]$ for all $\mu \in \Omega^P$. Then, since every $\mu \in \Omega^P$ is compatible with every $\mu^\text{in} \in \Omega^\mu$ and all processed solution mappings are compatible with $\mu^\text{in}$, it is easily verified that the computed result is $\mathbb{P}^W_{\mathcal{C}^W}[(P_1 \cup P_2) | \mu^\text{in}]$.

We are now ready to prove Theorem 29.

**Proof (Theorem 29).** Suppose $P$ is a graph pattern such that $\text{vars}(P | \emptyset) = \text{vars}(P)$. Then, by using the empty solution mapping $\mu_\emptyset$ with $\text{dom}(\mu_\emptyset) = \emptyset$, we have $\text{vars}(P | \text{dom}(\mu_\emptyset)) = \text{vars}(P)$. Therefore, by Lemma 32, there exists an algorithm, say $A$, that, for any finite Web of Linked Data $W = \langle D, \text{adoc} \rangle$, computes $\mathbb{P}^W_{\mathcal{C}^W}$ by looking up a finite number of IRIs only without using the set $D$ or the set.
While the condition given in Theorem 29 is sufficient to identify graph patterns that are Web-safe under context-based semantics, the question that remains is whether it is a necessary condition (i.e., whether it can be used to decide Web-safeness of all graph patterns under context-based semantics). Unfortunately, the answer is no as the following example shows.

Example 34. For the graph pattern \(P = (P_1 \cup P_2)\) with \(P_1 = \langle (u_1, p_1, ?x) \rangle\) and \(P_2 = \langle (u_2, p_2, ?y) \rangle\) we note that \(cbvars(P_1 | \emptyset) = \{ ?x \}\) and \(cbvars(P_2 | \emptyset) = \{ ?y \}\), and, thus, \(cbvars(P | \emptyset) = \emptyset\). Hence, the pattern does not satisfy the condition in Theorem 29. Nonetheless, it is easy to see that there exists a (sound and complete) algorithm that, for any finite Web of Linked Data \(W\), computes \(\llbracket P \rrbracket_W^{ctx}\) by looking up a finite number of IRIs only. For instance, such an algorithm, say \(A\), may first use two other algorithms that compute \(\llbracket P_1 \rrbracket_W^{ctx}\) and \(\llbracket P_2 \rrbracket_W^{ctx}\) by looking up a finite number of IRIs, respectively. Such algorithms exist by Theorem 29, because \(cbvars(P_1 | \emptyset) = \text{vars}(P_1)\) and \(cbvars(P_2 | \emptyset) = \text{vars}(P_2)\). Finally, algorithm \(A\) can generate the (sound and complete) query result \(\llbracket P \rrbracket_W^{ctx}\) by computing the multiset union \(\llbracket P_1 \rrbracket_W^{ctx} \cup \llbracket P_2 \rrbracket_W^{ctx}\), which requires no additional IRI lookups.

The example illustrates that “only if” cannot be shown in Theorem 29. It remains an open question whether there exists an alternative condition for Web-safeness that is both sufficient and necessary (and decidable) and, thus, can be used to decide Web-safeness of all graph patterns under context-based semantics.

7. Experimental Comparison

In the previous section we have shown that, when querying Linked Data on the WWW, it is possible for PP-based graph patterns to be evaluated completely under any reachability-based semantics, and, similarly, under the context-based semantics (assuming, for the latter, we use only patterns that have been identified to be Web-safe). Hence, we have shown that—based on these semantics—one can build a system that answers PP-based SPARQL queries over the WWW in a well-defined manner. At this point, a natural question that arises is:

\textit{How do these query semantics compare when actually used in practice?}

To achieve empirical insights related to this question we conducted an experimental comparison of the
context-based semantics and a reachability-based semantics. For this comparison we selected \( \text{CPPMatch} \)-semantics as an exemplar of the family of reachability-based semantics; as argued in Section 4.2, \( \text{CPPMatch} \) is very close in nature to the reachability criterion \( \text{cMatch} \) [18] which is commonly used in the literature on Linked Data query execution approaches [19,38] (note that \( \text{cMatch} \) is defined for SPARQL queries constructed from triple patterns, instead of PP patterns).

In the remainder of this section, we specify the experimental setup, describe the experiments, present the measurements, and discuss the experimental results.

7.1. Metrics and Experimental Setup

The objective of the experimental comparison is to identify the differences between the studied semantics in terms of (i) number of dereferencing operations performed to evaluate a query and (ii) number of solutions in the respective query results, including duplicates (which are possible in our bag semantics as Example 13 illustrates). Hereafter, we refer to these metrics as (i) \( n_{\text{nderef}} \) and (ii) \( r_{\text{ressize}} \), respectively. Since this paper focuses on possible query semantics rather than on efficient techniques to implement such semantics, performance-related metrics such as query execution time are out of scope of our study.

For the experiments, which we conducted during the days of November 16–28, 2015, we used a prototypical implementation of the studied semantics to execute PP-based SPARQL queries directly on the WWW. To avoid overloading Web servers we introduced a delay of 3 seconds between dereferencing operations. While we did not use any client-side caching of retrieved documents, there may have been Web caches (proxy servers) between our prototypical query clients and the Web servers that host the data discovered and retrieved during the execution of our test queries. Measurements reported in the following are the average of five executions with rounding to the next integer.

7.2. Experiments and Measurements

We conducted two different experiments considering two different topical domains of Linked Data on the WWW, namely, distributed social network data (D1) and encyclopedic data about influence relationships between people (D2). Within these domains we focus on navigational queries that we express using PP patterns. The particular queries used for the experiments can be found in Appendix A. In the following, we describe the experiments and the queries in more detail, and we present the measurements.

7.2.1. Experiment on D1

In our first experiment we considered the distributed social network of FOAF profiles [14]. Such FOAF profiles typically are RDF documents that people make available online to provide Linked Data that describes themselves in terms of their interests, their works, and, most important for our experiment, references to other people they know. Such references are expressed using triples with the IRI\(^3\) “foaf:knows” as predicate and the persons’ IRIs as subject and object (i.e., along the lines of our example Web in Figure 4). Hence, such triples establish data links between different people’s FOAF profiles. The resulting network of such “foaf:knows links” is thus a part of the Web of Linked Data, and it is the focus of our first experiment. We point out that this experiment is particularly significant due to the truly distributed nature of the FOAF profiles, which typically reside (and get updated) on different servers. Indeed, there is no SPARQL endpoint to query the live version of this kind of distributed social network.

In this experiment we use the IRI of Nuno Lopes\(^4\) in his FOAF profile as a starting point for six queries that

---

\(^3\)For the compact representation of IRIs in this section we use the following two prefixes. foaf: <http://xmlns.com/foaf/> and dbo: <http://dbpedia.org/ontology/>

\(^4\)http://nunolopes.org/foaf.rdf#me
retrieve Lopes’ acquaintances from distances 1 to 6, respectively. The measurements obtained by executing these queries under both the context-based semantics and the (reachability-based) _CPPMatch_ -semantics are reported in the charts in Figure 7; the x-axes list the six queries and the y-axes represent our metrics, _nderef_ and _ressize_, respectively (reported in log-scale).

By looking at Figure 7 (a), we notice that, under the context-based semantics, _nderef_ is increasing steadily with the (increasing) distance selected in the queries. In contrast, under _CPPMatch_ -semantics, _nderef_ is almost the same for all six queries, and it is significantly higher than under the context-based semantics, even for the distance-6 query. Considering the definition of the reachability criterion _CPPMatch_ (cf. Definition 10), this observation is not unexpected: While the PP patterns of the six queries differ, they all mention the same IRI in their PP expressions, namely, foaf:knows. Consequently, in all six cases, the same set of documents is reachable by applying _CPPMatch_ as reachability criterion (and using the same seed IRI). Essentially, this set of documents represents the complete strongly connected component of FOAF profiles that contains the profile of the seed IRI. Recall that the data of all these documents must be retrieved to compute query results that are guaranteed to be complete under _CPPMatch_ -semantics; this explains the comparable high number of dereferencing operations. The slight variations of these numbers across the six queries are due to occasional timeouts of dereferencing operations and Web servers that did not always respond during each query execution.

The effect of taking into account more data can be observed by looking at our _ressize_ measurements in Figure 7 (b). Clearly, under _CPPMatch_ -semantics we obtain query results that have a much greater size than the results under the context-based semantics, in particular, for the higher distance queries. This effect, again, is not unexpected. Instead, it can also be seen, on a much smaller scale, in the examples in Section 4 (compare in particular Examples 14 and 16). However, we note that the greater _ressize_ per query under _CPPMatch_ -semantics is due not only to finding paths to additional persons in the data retrieved under _CPPMatch_ -semantics, but also to a greater number of duplicates, which result from finding a greater number of alternative paths to some persons (cf. Example 3).

The only exception, where the query result under both semantics is the same, is the distance-1 query. This query consists only of a single triple pattern with the seed IRI as subject, foaf:knows as predicate, and a variable as object. In the given case of using Nuno Lopes’ IRI as seed, all triples that match this pattern happen to be in the same document (Lopes’ FOAF profile) and, thus, all other documents retrieved under _CPPMatch_ -semantics turn out to not contribute to the query result (which may be different for other seeds).

### 7.2.2. Experiment on D2

For our second experiment we considered influence relationships between people described in Linked Data that is made available by the DBpedia project [4]. In particular, we focused on the relationships expressed by triples with the IRI dbo:influencedBy as predicate, and we used the IRI of Veno Taufer\(^\text{5}\) as starting point for six queries that obtain influences of Taufer at distance 1 to 6, respectively. These queries are of the same form as the queries used in the first experiment. However, the main difference w.r.t. the first experiment is that the "dbo:influencedBy links" point only to data in DBpedia. In other words, every document that is reachable according to the reachability criterion _CPPMatch_ (and, thus, has to be retrieved under _CPPMatch_ -semantics) comes from the DBpedia Linked Data server. Hence, with this second experiment we wanted to capture a more dataset-centric scenario.

\(^5\)http://dbpedia.org/resource/Veno_Taufer
while the first experiment has captured a scenario in which the data to be discovered during query execution is truly distributed all over the WWW. Another important difference is that the `dbo:influencedBy` links are bidirectional; that is, any triple with predicate `dbo:influencedBy` can be found in both the document for the subject IRI of the triple and the document for the object IRI.

Due to the availability of these bidirectional data links, the query results under both semantics are the same for each of the six queries (cf. Figure 8). In contrast, the `nderef` measurements differ significantly and present the same pattern as observed in the first experiment. In fact, the number of dereferencing operations necessary to guarantee complete results under `CPPMatch`-semantics is even higher in the second experiment. We explain this observation by the fact that the strongly connected component established by the `dbo:influencedBy` links is bigger than the component of FOAF profiles in the first experiment. Apparently, this “fact” is known only after the corresponding traversal processes have been performed.

7.3. Discussion of the Experimental Results

Our experiments indicate that choosing one of the two tested query semantics over the other may have a significant impact in practice. Considering the size of query results first, our experiments show that there are cases in which the query result computed under the context-based semantics is smaller than under the (reachability-based) `CPPMatch`-semantics. We explain this finding by two important properties that distinguish the context-based semantics from reachability-based semantics such as the `CPPMatch`-semantics.

First, since it is based on the context selector (cf. Section 4.3), the context-based semantics ignores all the triples from any given document that have a subject IRI different from the IRI whose lookup resulted in retrieving the document. Ignoring such triples significantly decreases the number of paths (of triples) that can be found to match a given PP expression.

Second, the context-based semantics is designed to be very selective in the way the queried Web of Linked Data has to be traversed. More precisely, every traversal step is the result of first discovering a triple in the data of the current context document such that this triple can be used as a next step along a path that eventually may match the given PP expression. As a consequence of enforcing such a behavior, the traversal may not reach some documents that are reached under the `CPPMatch`-semantics, and some of these documents may happen to contain triples that can be used to compute additional solutions under the `CPPMatch`-semantics.

Our first experiment shows that this may happen in particular if the region of the Web that a query focuses on has a very heterogeneous link structure with many unidirectional links. On the other hand, if the link structure is more homogeneous, with mostly bidirectional links, then the query results under both semantics are more likely to coincide. Our second experiment presents an extreme case of such a scenario.

The downside of potentially larger query results that may be expected under `CPPMatch`-semantics is a greater number of dereferencing operations, which implies longer execution times and more network traffic generated. Our experiments provide remarkable evidence that this problem is not negligible. That is, for every query in our experiments the difference w.r.t. the corresponding number of dereferencing operations under the context-based semantics is substantial (up to two orders of magnitude). The fact that we made this observation in both experiments also shows that a greater number of dereferencing operations under `CPPMatch`-semantics is not a peculiarity of traversing an either more homogeneous or more heterogeneous link structure.

The significantly smaller number of dereferencing operations may be seen as a crucial advantage of the context-based semantics over the `CPPMatch`-semantics. The flip side of course is that users of systems that implement the context-based semantics may see query results with less solutions. Hence, choosing among the two semantics is a question of whether a user is willing to accept the price of possibly having to retrieve many more documents (and, thus, longer execution times) for the chance of seeing a greater number of solutions.

8. Concluding Remarks

This paper studies the problem of extending the scope of the Property Paths feature in SPARQL to query Linked Data that is distributed on the WWW. We have investigated reachability-based query semantics, which decouple navigation from querying. Additionally, we have proposed a different interpretation for PPs over the Web via the context-based query semantics. An interesting finding regarding this latter semantics is that there exist queries whose evaluation over the WWW is not possible in practice. We studied this aspect using a notion of Web-safeness and introduced a decidable syntactic property for identifying
queries that are Web-safe under the context-based semantics. Moreover, we have presented an experimental evaluation that compares the two semantics on different datasets showing that the context-based semantics incurs in a lower number of dereferencing operations that will have an impact on the running time.

We believe that the presented work provides valuable input to a wider discussion about defining how the SPARQL language can be used for accessing Linked Data on the WWW. There are several directions for future research including an investigation of the relationships between navigational queries and SPARQL federation, as well as an exploration of techniques based on which query execution systems may implement efficiently the machinery developed in this paper.

Acknowledgements

We thank the ESWC reviewers and the SWJ reviewers for their valuable feedback. Olaf Hartig’s work has been funded by the German Government, Federal Ministry of Education and Research under the project number 03WKCI4D. Giuseppe Pirrò’s work has been funded by the Cyber Security Technological District financed by the Italian MIUR.

References


[31] J. Pérez, M. Arenas, and C. Gutierrez. nSPARQL: A naviga-


sanne, Switzerland, March 23-25, 2010, Proceedings, ACM In-


[36] T. T. Tang, D. Hawking, N. Craswell, and K. Griffiths. Focused crawling for both topical relevance and quality of medical in-
formation. In O. Herzog, H. Schek, N. Fuhr, A. Chowdhury, and W. Teiken, editors, Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Manage-
ment, Bremen, Germany, October 31 - November 5, 2005,


Appendix A: Queries used in the Evaluation

This appendix provides the queries used in our experiment. These queries use the following prefixes:

PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/>

8.1. Distance 1 Query for the Experiment on D1

SELECT ?end WHERE {
  <http://nunolopes.org/foaf.rdf/#me> foaf:knows ?end
}

8.2. Distance 2 Query for the Experiment on D1

SELECT ?end WHERE {
  <http://nunolopes.org/foaf.rdf/#me> foaf:knows/foaf:knows ?end
}

8.3. Distance 3 Query for the Experiment on D1

SELECT ?end WHERE {
}

8.4. Distance 4 Query for the Experiment on D1

SELECT ?end WHERE {
}

8.5. Distance 5 Query for the Experiment on D1

SELECT ?end WHERE {
}

8.6. Distance 6 Query for the Experiment on D1

SELECT ?end WHERE {
}

8.7. Distance 1 Query for the Experiment on D2

SELECT ?end WHERE {
  <http://dbpedia.org/resource/Veno_Taufer> dbo:influencedBy ?end
}

8.8. Distance 2 Query for the Experiment on D2

SELECT ?end WHERE {
  <http://dbpedia.org/resource/Veno_Taufer> dbo:influencedBy/dbo:influencedBy ?end
}
8.9. Distance 3 Query for the Experiment on D2

```
SELECT ?end WHERE {
  <http://dbpedia.org/resource/Veno_Taufer>
    dbo:influencedBy/dbo:influencedBy/dbo:influencedBy ?end
}
```

8.10. Distance 4 Query for the Experiment on D2

```
SELECT ?end WHERE {
  <http://dbpedia.org/resource/Veno_Taufer>
    dbo:influencedBy/dbo:influencedBy/dbo:influencedBy/dbo:influencedBy ?end
}
```

8.11. Distance 5 Query for the Experiment on D2

```
SELECT ?end WHERE {
  <http://dbpedia.org/resource/Veno_Taufer>
    dbo:influencedBy/dbo:influencedBy/dbo:influencedBy/dbo:influencedBy/dbo:influencedBy ?end
}
```

8.12. Distance 6 Query for the Experiment on D2

```
SELECT ?end WHERE {
  <http://dbpedia.org/resource/Veno_Taufer>
    dbo:influencedBy/dbo:influencedBy/dbo:influencedBy/
      dbo:influencedBy/dbo:influencedBy/dbo:influencedBy ?end
}
```