A Context-Based Semantics for SPARQL Property Paths over the Web

Olaf Hartig[1] and Giuseppe Pirrò[2]

[1] University of Waterloo, Canada
ohartig@uwaterloo.ca
[2] Institute for High Performance Computing and Networking, ICAR-CNR, Rende, Italy
pirro@icar.cnr.it

Abstract As of today, there exists no standard language for querying Linked Data *on the Web*, where navigation across distributed data sources is a key feature. A natural candidate seems to be SPARQL, which recently has been enhanced with navigational capabilities thanks to the introduction of *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 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. In particular, we introduce a query semantics for PPs that couples navigation at the data level with navigation on the Web graph. Given this semantics we find that for some PP-based SPARQL queries a complete evaluation on the Web is not feasible. To enable systems to identify queries that can be evaluated completely, we establish a decidable syntactic property of such queries.

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 [4] 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) [16] as standard data format. RDF adopts a graph-based data model, which can be queried upon by using the SPARQL query language [12]. When it comes to Linked Data on the WWW, the common way to provide query-based access 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. Hence, as of today, there exists no standard query language for Linked Data on the WWW, although SPARQL is clearly a candidate.

While earlier research on using SPARQL for Linked Data is limited to fragments of the first version of the language [5,13,14,25], the more recent version 1.1 introduces a

feature that is particularly interesting in the context of queries over a graph-like environment such as Linked Data on the WWW. This feature is called *property paths* (PPs) and equips SPARQL with navigational capabilities [12]. However, the standard definition of PPs is limited to single, centralized 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. To this end, we make the following main contributions:

1. We formalize a query semantics for PP-based SPARQL queries that are meant to be evaluated over Linked Data on the WWW. This semantics is *context-based*; it intertwines Web graph navigation with navigation at the level of data.
2. We study the feasibility of evaluating queries under this semantics. 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 there exist cases in which query evaluation under the context-based semantics is not feasible.
3. We provide a decidable syntactic property of queries for which an evaluation under the context-based semantics is feasible.

The remainder of the paper is organized as follows. Section 2 provides an overview on related work. Section 3 introduces the formal framework for this paper, including a data model that captures a notion of Linked Data. In Section 4 we focus on PPs, independently from other SPARQL operators. In Section 5 we broaden our view to study PP-based SPARQL graph patterns; we characterize a class of *Web-safe* patterns and prove their feasibility. Finally, in Section 6 we conclude and sketch future work.

2 Related Work

The idea of querying the WWW as a database is not new (see Florescu et al.'s survey [11]). Perhaps the most notable early works in this context are by Konopnicki and Shmueli [18], Abiteboul and Vianu [1], and Mendelzon et al. [20], all of which tackled the problem of evaluating SQL-like queries on the traditional 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.

From a graph-oriented perspective, languages for the *navigation and specification* of vertices in graphs have a long tradition (see Wood's survey [26]). In the RDF world, extensions of SPARQL such as PSPARQL [2], nSPARQL [21], and SPARQLeR [17] introduced navigational features since those were missing in the first version of SPARQL. Only recently, with the addition of *property paths* (PPs) in version 1.1 [12], 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 [3,19], and there also already exists a proposal to extend PPs with more expressive power [9]. However, the main assumption of all these navigational extensions of SPARQL is to work on a single, centralized RDF graph. Our departure point is different: *We aim at defining semantics of SPARQL queries (including property paths)*

over Linked Data on the WWW, which involves dealing with two graphs of different types; namely, an RDF graph that is distributed over documents on the WWW and the Web graph of how 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 [5,13,25] or a more expressive fragment that includes AND, OPT, UNION and FILTER [14]. The second strand focuses on navigational languages such as NautiLOD [8,10]. 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. We believe that the outcome of this research can be a starting point toward the definition of a language for querying and navigating over Linked Data on the WWW.

3 Formal Framework

This section provides a formal framework for studying semantics of PPs over Linked Data. We first recall the definition of PPs as per the SPARQL standard [12]. Thereafter, we introduce a data model that captures the notion of Linked Data on the WWW.

3.1 Preliminaries

Assume four pairwise disjoint, countably infinite sets $\mathcal{I}$ (IRIs), $\mathcal{B}$ (blank nodes), $\mathcal{L}$ (literals), and $\mathcal{V}$ (variables). An *RDF triple* (or simply *triple*) is a tuple from the set $\mathcal{T} = (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$. For any triple $t \in \mathcal{T}$ we write $\mathrm{iris}(t)$ to denote the set of IRIs in that triple. A set of triples is called an *RDF graph*.

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

$$\mathtt{path} = u \mid \mathtt{!}(u_1 \mid \ldots \mid u_n) \mid \mathtt{\hat{}path} \mid \mathtt{path/path} \mid (\mathtt{path|path}) \mid (\mathtt{path})^*$$

Note that the SPARQL standard introduces additional types of PP expressions [12]. 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. *An example of a PP pattern is* $\langle \mathsf{Tim}, (\mathsf{knows})^*/\mathsf{name}, ?n \rangle$*, which retrieves the names of persons that can be reached from* Tim *by an arbitrarily long path of* knows *relationships (which includes* Tim*). Another example are the two PP patterns* $\langle ?p, \mathsf{knows}, \mathsf{Tim} \rangle$ *and* $\langle \mathsf{Tim}, \hat{}\mathsf{knows}, ?p \rangle$*, both of which retrieve persons that know* Tim.

The (standard) query semantics of PP patterns is defined by an evaluation function that returns multisets of *solution mappings* where a solution mapping μ is a partial function

Function $\texttt{ALP1}(\gamma, \texttt{path}, G)$

Input: $\gamma \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, `path` is a PP expression, G is an RDF graph.

1: *Visited* := $\emptyset$
2: $\texttt{ALP2}(\gamma, \texttt{path}, \mathit{Visited}, G)$
3: **return** *Visited*

Function $\texttt{ALP2}(\gamma, \texttt{path}, \mathit{Visited}, G)$

Input: $\gamma \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, `path` is a PP expression, $\mathit{Visited} \subseteq (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, G is an RDF graph.

4: **if** $\gamma \notin \mathit{Visited}$ **then**
5: add γ to *Visited*
6: **for all** $\mu \in [\![\langle ?x, \texttt{path}, ?y\rangle]\!]_G$ s.t. $\mu(?x) = \gamma$ **do**
7: $\texttt{ALP2}(\mu(?y), \texttt{path}, \mathit{Visited}, G)$ // $?x, ?y \in \mathcal{V}$

Figure 1. Auxiliary functions for defining the semantics of PP expressions of the form `path`*.

$\mu : \mathcal{V} \to (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$. Given a solution mapping μ and a PP pattern P, we write $\mu[P]$ to denote the PP pattern obtained by replacing the variables in P according to μ (unbound variables must not be replaced). Two solution mappings, say μ_1 and μ_2, are *compatible*, denoted by $\mu_1 \sim \mu_2$, if $\mu_1(?v) = \mu_2(?v)$ for all variables $?v \in \big(\mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)\big)$.

We represent a *multiset* of solution mappings by a pair $M = \langle \Omega, \mathit{card}\rangle$ where Ω is the underlying set (of solution mappings) and $\mathit{card} : \Omega \to \{1, 2, ...\}$ is the corresponding *cardinality function*. By abusing notation slightly, we write $\mu \in M$ for all $\mu \in \Omega$. Furthermore, we introduce a family of special (parameterized) cardinality functions that shall simplify the definition of any multiset whose solution mappings all have a cardinality of 1. That is, for any set of solution mappings Ω, let $\mathsf{card1}^{(\Omega)} : \Omega \to \{1, 2, ...\}$ be the *constant-1 cardinality function* that is defined by $\mathsf{card1}^{(\Omega)}(\mu) = 1$ for all $\mu \in \Omega$.

To define the aforementioned evaluation function we also need to introduce several SPARQL algebra operators. Let $M_1 = \langle \Omega_1, \mathit{card}_1\rangle$ and $M_2 = \langle \Omega_2, \mathit{card}_2\rangle$ be multisets of solution mappings and let $V \subseteq \mathcal{V}$ be a finite set of variables. Then:

- $M_1 \sqcup M_2 = \langle \Omega, \mathit{card}\rangle$ where $\Omega = \Omega_1 \cup \Omega_2$ and (i) $\mathit{card}(\mu) = \mathit{card}_1(\mu)$ for all solution mappings $\mu \in \Omega \setminus \Omega_2$, (ii) $\mathit{card}(\mu) = \mathit{card}_2(\mu)$ for all $\mu \in \Omega \setminus \Omega_1$, and (iii) $\mathit{card}(\mu) = \mathit{card}_1(\mu) + \mathit{card}_2(\mu)$ for all $\mu \in \Omega_1 \cap \Omega_2$.
- $M_1 \bowtie M_2 = \langle \Omega, \mathit{card}\rangle$ where $\Omega = \big\{\, \mu_1 \cup \mu_2 \,|\, (\mu_1, \mu_2) \in \Omega_1 \times \Omega_2 \text{ and } \mu_1 \sim \mu_2 \big\}$ and, for every $\mu \in \Omega$, $\mathit{card}(\mu) = \sum_{(\mu_1,\mu_2) \in \Omega_1 \times \Omega_2 \text{ s.t. } \mu = \mu_1 \cup \mu_2} \mathit{card}(\mu_1) \cdot \mathit{card}(\mu_2)$.
- $M_1 \setminus M_2 = \langle \Omega, \mathit{card}\rangle$ where $\Omega = \big\{\, \mu_1 \in \Omega_1 \,|\, \mu_1 \not\sim \mu_2 \text{ for all } \mu_2 \in \Omega_2 \big\}$ and, for every $\mu \in \Omega$, $\mathit{card}(\mu) = \mathit{card}_1(\mu)$.
- $\pi_V(M_1) = \langle \Omega, \mathit{card}\rangle$ where $\Omega = \big\{\mu \,|\, \exists \mu' \in \Omega_1 : \mu \sim \mu' \text{ and } \mathrm{dom}(\mu) = V \cap \mathrm{dom}(\mu')\big\}$ and, for every $\mu \in \Omega$, $\mathit{card}(\mu) = \sum_{\mu' \in \Omega_1 \text{ s.t. } \mu \sim \mu'} \mathit{card}_1(\mu')$.

In addition to these algebra operators, the SPARQL standard introduces auxiliary functions to define the semantics of PP patterns of the form $\langle \alpha, \texttt{path}^*, \beta\rangle$. Figure 1 provides these functions—which we call `ALP1` and `ALP2`—adapted to our formalism.[3]

We are now ready to define the standard query semantics of PP patterns.

Definition 1. *The* evaluation *of a PP pattern P over an RDF graph G, denoted by $[\![P]\!]_G$, is a multiset of solution mappings $\langle \Omega, \mathit{card}\rangle$ that is defined recursively as given in Figure 2 where $\alpha, \beta \in (\mathcal{I} \cup \mathcal{L} \cup \mathcal{V})$, $x_\mathrm{L}, x_\mathrm{R} \in (\mathcal{I} \cup \mathcal{L})$, $?v_\mathrm{L}, ?v_\mathrm{R} \in \mathcal{V}$, $u, u_1, ..., u_n \in \mathcal{I}$, $?v \in \mathcal{V}$ is a fresh variable, and $\mu_\emptyset$ denotes the empty solution mapping ($\mathrm{dom}(\mu_\emptyset) = \emptyset$).*

[3] Variable $?x$ in line 6 is necessary since PP patterns in our formalism do not have blank nodes.

$$
\begin{aligned}
[\![\langle \alpha, u, \beta\rangle]\!]_G &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = (\{\alpha,\beta\} \cap \mathcal{V}) \text{ and } \mu[\langle \alpha, u, \beta\rangle] \in G \,\},\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle \alpha, !(u_1 \mid \ldots \mid u_n), \beta\rangle]\!]_G &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = (\{\alpha,\beta\} \cap \mathcal{V}) \text{ and } \\
&\qquad \exists\, \mu[\langle \alpha, u, \beta\rangle] \in G : u \in (\mathcal{I} \setminus \{u_1, \ldots, u_n\}) \,\},\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle \alpha, {}^{\wedge}\mathtt{path}, \beta\rangle]\!]_G &= [\![\langle \beta, \mathtt{path}, \alpha\rangle]\!]_G \\
[\![\langle \alpha, \mathtt{path}_1/\mathtt{path}_2, \beta\rangle]\!]_G &= \pi_{\{\alpha,\beta\}\cap\mathcal{V}}\Big([\![\langle \alpha, \mathtt{path}_1, ?v\rangle]\!]_G \bowtie [\![\langle ?v, \mathtt{path}_2, \beta\rangle]\!]_G\Big) \\
[\![\langle \alpha, (\mathtt{path}_1 | \mathtt{path}_2), \beta\rangle]\!]_G &= [\![\langle \alpha, \mathtt{path}_1, \beta\rangle]\!]_G \sqcup [\![\langle \alpha, \mathtt{path}_2, \beta\rangle]\!]_G \\
[\![\langle x_{\mathrm{L}}, (\mathtt{path})^*, ?v_{\mathrm{R}}\rangle]\!]_G &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = \{?v_{\mathrm{R}}\} \text{ and } \mu(?v_{\mathrm{R}}) \in \mathtt{ALP1}(x_{\mathrm{L}}, \mathtt{path}, G) \,\},\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle ?v_{\mathrm{L}}, (\mathtt{path})^*, ?v_{\mathrm{R}}\rangle]\!]_G &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = \{?v_{\mathrm{L}}, ?v_{\mathrm{R}}\} \text{ and } \mu(?v_{\mathrm{L}}) \in \mathrm{terms}(G) \text{ and } \\
&\qquad \mu(?v_{\mathrm{R}}) \in \mathtt{ALP1}(\mu(?v_{\mathrm{L}}), \mathtt{path}, G) \,\},\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle ?v_{\mathrm{L}}, (\mathtt{path})^*, x_{\mathrm{R}}\rangle]\!]_G &= [\![\langle x_{\mathrm{R}}, ({}^{\wedge}\mathtt{path})^*, ?v_{\mathrm{L}}\rangle]\!]_G \\
[\![\langle x_{\mathrm{L}}, (\mathtt{path})^*, x_{\mathrm{R}}\rangle]\!]_G &= \Big\langle \begin{cases} \{\mu_\emptyset\} & \text{if } \exists\, \mu \in [\![\langle x_{\mathrm{L}}, (\mathtt{path})^*, ?v\rangle]\!]_G : \mu(?v) = x_{\mathrm{R}}, \\ \emptyset & \text{else} \end{cases},\ \mathsf{card1}^{(\Omega)} \Big\rangle
\end{aligned}
$$

Figure 2. SPARQL 1.1 W3C property paths semantics.

3.2 Data Model

The standard SPARQL evaluation function for PP patterns (cf. Section 3.1) defines the expected result of the evaluation of a pattern over a single RDF graph. Since the WWW is not an RDF graph, the standard definition is insufficient as a formal foundation for evaluating PP patterns over Linked Data on the WWW. To provide 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 proposed in our earlier work [14]. Here, a *Web of Linked Data* (*WoLD*) is a tuple $W = \langle D, data, adoc\rangle$ consisting of (i) a set D of so called *Linked Data documents* (*documents*), (ii) a mapping $data : D \to 2^{\mathcal{T}}$ that maps each document to a finite set of RDF triples (representing the data that can be obtained from the document), and (iii) a partial mapping $adoc : \mathcal{I} \to D$ that maps (some) IRIs to a document and, thus, captures a IRI-based retrieval of documents. In this paper we assume that the set of documents D in any WoLD $W = \langle D, data, adoc\rangle$ is finite, in which case we say W is *finite* (for a discussion of infiniteness refer to our earlier work [14]).

A few other concepts are needed for the subsequent discussion. For any two documents $d, d' \in D$ in a WoLD $W = \langle D, data, adoc\rangle$, document d has a *data link* to d' if the data of d mentions an IRI $u \in \mathcal{I}$ (i.e., there exists a triple $\langle s, p, o\rangle \in data(d)$ with $u \in \{s, p, o\}$) that can be used to retrieve d' (i.e., $adoc(u) = d'$). Such data links establish the *link graph* of the WoLD W, that is, a directed graph $\langle D, E\rangle$ in which the edges E are all pairs $\langle d, d'\rangle \in D \times D$ for which d has a data link to d'. Note that this graph, as well as the tuple $\langle D, data, adoc\rangle$ typically are not available directly to systems that aim to compute queries over the Web captured by W. For instance, the complete domain of the partial mapping $adoc$ (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). Also note that the link graph of a WoLD is a different type of graph than the RDF "graph" whose triples are distributed over the documents in the WoLD.

4 Web-aware Query Semantics for Property Paths

We are now ready to introduce our framework, which does not deal with syntactic aspects of PPs but aims at defining query semantics that provide a formal foundation for using PP patterns as queries over a WoLD (and, thus, over Linked Data on the WWW).

4.1 Full-Web Query Semantics

As a first approach we may assume a full-Web query semantics that is based on the standard evaluation function (as introduced in Section 3.1) and defines an expected query result for any PP pattern in terms of *all data* on the queried WoLD. Formally:

Definition 2. *Let P be a PP pattern, let $W = \langle D, data, adoc \rangle$ be a WoLD, and let G^* be an RDF graph such that $G^* = \bigcup_{d \in D} data(d)$, then the* evaluation *of P over W under* full-Web semantics*, denoted by $[\![P]\!]^{\mathtt{fw}}_W$, is defined by $[\![P]\!]^{\mathtt{fw}}_W = [\![P]\!]_{G^*}$.*

We emphasize that the full-Web query semantics is mostly of theoretical interest. In practice, that is, for a WoLD W that represents the "real" WWW (as it runs on the Internet), there cannot exist a system that guarantees to compute the given evaluation function $[\![\cdot]\!]^{\mathtt{fw}}_{\cdot}$ over W using an algorithm that both terminates and returns complete query results. In earlier work, we showed such a limitation for evaluating other types of SPARQL graph patterns—including triple patterns—under a corresponding full-Web query semantics defined for these patterns [14]. This result readily carries over to the full-Web query semantics for PP patterns because any PP pattern $P = \langle \alpha, \mathtt{path}, \beta \rangle$ with PP expression $\mathtt{path}$ being an IRI $u \in \mathcal{I}$ is, in fact, a triple pattern $\langle \alpha, u, \beta \rangle$. Informally, we explain this negative result by the fact that the three structures D, $data$, and $adoc$ that capture the queried Web formally, are not available in practice. Consequently, to enumerate the set of all triples on the Web (i.e., the RDF graph G^* in Definition 2), a query execution system would have to enumerate all documents (the set D); given that such a system has limited access to mapping $adoc$ (in particular, $\mathrm{dom}(adoc)$—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-scheme) IRI. Since these are infinitely many [7], the enumeration process cannot terminate.

4.2 Context-Based Query Semantics

Given the limited practical applicability of full-Web query semantics for PPs, we propose an alternative query semantics that interprets PP patterns as a language for navigation over Linked Data on the Web (i.e., along the lines of earlier navigational languages for Linked Data such as NautiLOD [8]). We refer to this semantics as *context-based*.

The main idea behind 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 WoLD. As a basis for formalizing 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 RDF triples that have the given IRI as their subject (such

a set of triples resembles Harth and Speiser's notion of subject authoritative triples [13]). Formally, for any WoLD $W = \langle D, data, adoc \rangle$, the context selector of W is a function $\mathrm{C}^W \colon \mathcal{I} \cup \mathcal{B} \cup \mathcal{L} \cup \mathcal{V} \to 2^{\mathcal{T}}$ that, for each $\gamma \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L} \cup \mathcal{V})$, is defined as follows:[4]

$$\mathrm{C}^W(\gamma) = \begin{cases} \{\langle s,p,o\rangle \in data\big(adoc(\gamma)\big) \,\big|\, \gamma = s\} & \text{if } \gamma \in \mathcal{I} \text{ and } \gamma \in \mathrm{dom}(adoc), \\ \emptyset & \text{otherwise.} \end{cases}$$

Informally, we explain how a context selector restricts the scope of PP patterns over a WoLD as follows. Suppose a sequence of triples $\langle s_1, p_1, o_1\rangle, \ldots, \langle s_k, p_k, o_k\rangle$ presents a path that already matches a sub-expression of a given PP expression. Under the previously defined full-Web query semantics (cf. Section 4.1), the next triple for such a path can be searched for in an arbitrary document in the queried WoLD W. By contrast, under the context-based query semantics, the next triple has to be searched for only in $\mathrm{C}^W(o_k)$. Given these preliminaries, we now define context-based semantics:

Definition 3. *Let P be a PP pattern and let $W = \langle D, data, adoc\rangle$ be a WoLD. The* evaluation *of P over W under* context-based semantics*, denoted by $[\![P]\!]_W^{ctx}$, returns a multiset of solution mappings $\langle \Omega, card\rangle$ defined recursively as given in Figure 3, where $u, .., u_n \in \mathcal{I}$; $x_\mathrm{L}, x_\mathrm{R} \in (\mathcal{I} \cup \mathcal{L})$; $?v_\mathrm{L}, ?v_\mathrm{R} \in \mathcal{V}$; $\mu_\emptyset$ is the empty solution mapping (i.e., $\mathrm{dom}(\mu_\emptyset) = \emptyset$); function* `ALPW1` *is given in Figure 4; and $?v \in \mathcal{V}$ is a fresh variable.*

There are three points worth mentioning w.r.t. Definition 3: First, note how the context selector restricts the data that has to be searched to find matching triples (e.g., consider the first line in Figure 3). Second, we emphasize that context-based query semantics is defined such that it resembles the standard semantics of PP patterns as close as possible (cf. Section 3.1). Therefore, for the part of our definition that covers PP patterns of the form $\langle \alpha, \mathtt{path}^*, \beta\rangle$, we also use auxiliary functions—`ALPW1` and `ALPW2` (cf. Figure 4). These functions evaluate the sub-expression `path` recursively over the queried WoLD (instead of using a fixed RDF graph as done in the standard semantics in Figure 1). Third, the two base cases with a variable in the subject position (i.e., the third and the sixth line in Figure 3) require an enumeration of all IRIs. Such a requirement is necessary to preserve consistency with the standard semantics, as well as to preserve commutativity of operators that can be defined on top of PP patterns (such as the 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 2. *Consider the PP pattern $P_{\mathsf{E2}} = \langle ?v, \mathsf{knows}, \mathsf{Tim}\rangle$, which asks for the IRIs of people that know Tim. 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 includes the triple $\langle u', \mathsf{knows}, \mathsf{Tim}\rangle$. While, for any WoLD that is finite, there exists only a finite number of such IRIs, determining these IRIs and guaranteeing completeness requires to enumerate the infinite set of all IRIs and to check each of them (unless one knows the complete—and finite—subset of*

[4] To simplify the following formalization of context-based semantics, context selectors are defined not only over IRIs, but also over blank nodes, literals, and variables.

$$
\begin{aligned}
[\![\langle u_{\mathrm{L}}, p, \beta\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = (\{\beta\} \cap \mathcal{V}) \text{ and } \mu[\langle u_{\mathrm{L}}, p, \beta\rangle] \in \mathrm{C}^W(u_{\mathrm{L}}) \,\}\,,\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle l_{\mathrm{L}}, p, \beta\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \emptyset, \mathsf{card1}^{(\emptyset)} \Big\rangle \\
[\![\langle ?v_{\mathrm{L}}, p, \beta\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = (\{?v_{\mathrm{L}}, \beta\} \cap \mathcal{V}) \text{ and } \\
&\qquad \mu[\langle ?v_{\mathrm{L}}, p, \beta\rangle] \in \bigcup_{u \in \mathcal{I}} \mathrm{C}^W(u) \,\}\,,\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle u_{\mathrm{L}}, !(u_1 \mid \cdots \mid u_n), \beta\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = (\{\beta\} \cap \mathcal{V}) \text{ and } \\
&\qquad \exists\, \mu[\langle u_{\mathrm{L}}, p, \beta\rangle] \in \mathrm{C}^W(u_{\mathrm{L}}) : p \notin \{u_1, \ldots, u_n\} \,\}\,,\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle l_{\mathrm{L}}, !(u_1 \mid \cdots \mid u_n), \beta\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \emptyset, \mathsf{card1}^{(\emptyset)} \Big\rangle \\
[\![\langle ?v_{\mathrm{L}}, !(u_1 \mid \cdots \mid u_n), \beta\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = (\{?v_{\mathrm{L}}, \beta\} \cap \mathcal{V}) \text{ and } \\
&\qquad \exists\, \mu[\langle ?v_{\mathrm{L}}, p, \beta\rangle] \in \bigcup_{u \in \mathcal{I}} \mathrm{C}^W(u) : p \notin \{u_1, \ldots, u_n\} \,\}\,,\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle \alpha, {}^{\wedge}\mathtt{path}, \beta\rangle]\!]_W^{\mathtt{ctx}} &= [\![\langle \beta, \mathtt{path}, \alpha\rangle]\!]_W^{\mathtt{ctx}} \\
[\![\langle \alpha, \mathtt{path}_1/\mathtt{path}_2, \beta\rangle]\!]_W^{\mathtt{ctx}} &= \pi_{\{\alpha,\beta\}\cap\mathcal{V}}\Big([\![\langle \alpha, \mathtt{path}_1, ?v\rangle]\!]_W^{\mathtt{ctx}} \bowtie [\![\langle ?v, \mathtt{path}_2, \beta\rangle]\!]_W^{\mathtt{ctx}}\Big) \\
[\![\langle \alpha, \mathtt{path}_1 \mid \mathtt{path}_2, \beta\rangle]\!]_W^{\mathtt{ctx}} &= [\![\langle \alpha, \mathtt{path}_1, \beta\rangle]\!]_W^{\mathtt{ctx}} \sqcup [\![\langle \alpha, \mathtt{path}_2, \beta\rangle]\!]_W^{\mathtt{ctx}} \\
[\![\langle x_{\mathrm{L}}, (\mathtt{path})^*, ?v_{\mathrm{R}}\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = \{?v_{\mathrm{R}}\} \text{ and } \mu(?v_{\mathrm{R}}) \in \mathtt{ALPW1}(x_{\mathrm{L}}, \mathtt{path}, W)\} \,\}\,,\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle ?v_{\mathrm{L}}, (\mathtt{path})^*, ?v_{\mathrm{R}}\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \{\, \mu \mid \mathrm{dom}(\mu) = \{?v_{\mathrm{L}}, ?v_{\mathrm{R}}\} \text{ and } \mu(?v_{\mathrm{L}}) \in \mathrm{terms}(W) \text{ and } \\
&\qquad \mu(?v_{\mathrm{R}}) \in \mathtt{ALWP1}(\mu(?v_{\mathrm{L}}), \mathtt{path}, W) \,\}\,,\ \mathsf{card1}^{(\Omega)} \Big\rangle \\
[\![\langle ?v_{\mathrm{L}}, (\mathtt{path})^*, x_{\mathrm{R}}\rangle]\!]_W^{\mathtt{ctx}} &= [\![\langle x_{\mathrm{R}}, ({}^{\wedge}\mathtt{path})^*, ?v_{\mathrm{L}}\rangle]\!]_W^{\mathtt{ctx}} \\
[\![\langle x_{\mathrm{L}}, (\mathtt{path})^*, x_{\mathrm{R}}\rangle]\!]_W^{\mathtt{ctx}} &= \Big\langle \begin{cases} \{\mu_\emptyset\} & \text{if } \exists\, \mu \in [\![\langle x_{\mathrm{L}}, (\mathtt{path})^*, ?v\rangle]\!]_W^{\mathtt{ctx}} : \mu(?v) = x_{\mathrm{R}}, \\ \emptyset & \text{else} \end{cases} ,\ \mathsf{card1}^{(\Omega)} \Big\rangle
\end{aligned}
$$

Figure 3. Context-based query semantics for SPARQL property paths over the Web.

all IRIs that can be used to retrieve some document, which, due to the infiniteness of possible HTTP 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 not a sufficient condition in general. For instance, the PP pattern $\langle \mathsf{Tim}, {}^{\wedge}\mathsf{knows}, ?v\rangle$ has the same issue as the pattern in Example 2 (in fact, both patterns are semantically equivalent under context-based semantics). A question that arises is whether there exists a property of PP patterns that can be used to distinguish between patterns that do not have this issue (i.e., evaluating them over any WoLD is feasible) and those that do. We shall discuss this question for the more general case of PP-based SPARQL queries.

5 SPARQL with Property Paths on the Web

After considering PP patterns in separation, 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. We define the resulting PP-based SPARQL queries, discuss the fea-

Function $\texttt{ALPW1}(\gamma, \texttt{path}, W)$

Input: $\gamma \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, $\texttt{path}$ is a PP expression, W is a WoLD.

1: *Visited* := $\emptyset$
2: $\texttt{ALPW2}(\gamma, \texttt{path}, \mathit{Visited}, W)$
3: **return** *Visited*

Function $\texttt{ALPW2}(\gamma, \texttt{path}, \mathit{Visited}, W)$

Input: $\gamma \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, $\texttt{path}$ is a PP expression, $\mathit{Visited} \subseteq (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, W is a WoLD.

4: **if** $\gamma \notin \mathit{Visited}$ **then**
5: add γ to *Visited*
6: **for all** $\mu \in [\![\langle ?x, \texttt{path}, ?y\rangle]\!]_W^{\texttt{ctx}}$ s.t. $\mu(?x) = \gamma$ **do**
7: $\texttt{ALPW2}(\mu(?y), \texttt{path}, \mathit{Visited}, W)$ // $?x, ?y \in \mathcal{V}$

Figure 4. Auxiliary functions used for defining context-based query semantics.

sibility of evaluating these queries over the Web, and introduce a syntactic property to identify queries for which an evaluation under context-based semantics is feasible.

5.1 Definition

By using the algebraic syntax of SPARQL [22], we define a *graph pattern* recursively as follows: (i) Any PP pattern $\langle \alpha, \texttt{path}, \beta \rangle$ is a graph pattern; and (ii) if P_1 and P_2 are graph patterns, then $(P_1 \text{ AND } P_2)$, $(P_1 \text{ UNION } P_2)$, and $(P_1 \text{ OPT } P_2)$ are graph patterns.[5] For any graph pattern P, we write $\texttt{V}(P)$ to denote the set of *all variables* in P.

By using PP patterns as the basic building block of graph patterns, we can readily carry over our context-based semantics to graph patterns: For any graph pattern P and any WoLD W, the *evaluation* of P over W under context-based semantics is a multiset of solution mappings, denoted by $[\![P]\!]_W^{\texttt{ctx}}$, that is defined recursively as follows:[6]

- If P is a PP pattern, then $[\![P]\!]_W^{\texttt{ctx}}$ is defined in Definition 3.
- If P is $(P_1 \text{ AND } P_2)$, then $[\![P]\!]_W^{\texttt{ctx}} = [\![P_1]\!]_W^{\texttt{ctx}} \bowtie [\![P_2]\!]_W^{\texttt{ctx}}$.
- If P is $(P_1 \text{ UNION } P_2)$, then $[\![P]\!]_W^{\texttt{ctx}} = [\![P_1]\!]_W^{\texttt{ctx}} \sqcup [\![P_2]\!]_W^{\texttt{ctx}}$.
- If P is $(P_1 \text{ OPT } P_2)$, then $[\![P]\!]_W^{\texttt{ctx}} = \big([\![P_1]\!]_W^{\texttt{ctx}} \bowtie [\![P_2]\!]_W^{\texttt{ctx}}\big) \sqcup \big([\![P_1]\!]_W^{\texttt{ctx}} \setminus [\![P_2]\!]_W^{\texttt{ctx}}\big)$.

5.2 Discussion

Given a query semantics for evaluating PP-based graph patterns over a WoLD, we now discuss the feasibility of such evaluation. To this end, we introduce the notion of *Web-safeness* of graph patterns. Informally, graph patterns are Web-safe if evaluating them completely under context-based semantics is possible. Formally:

Definition 4. *A graph pattern P is* Web-safe *if there exists an algorithm that, for any finite WoLD $W = \langle D, data, adoc \rangle$, computes $[\![P]\!]_W^{\texttt{ctx}}$ by looking up only a finite number of IRIs without assuming direct access to the sets D and* $\mathrm{dom}(adoc)$.

[5] For this paper we leave out other types of SPARQL graph patterns such as filters. Adding them is an exercise that would not have any significant implication on the following discussion.

[6] Note that the definition uses the algebra operators introduced in Section 3.1.

Example 3. *Consider graph pattern* $P_{\mathsf{E3}} = \big(\langle \mathsf{Bob}, \mathsf{knows}, ?v\rangle \ \mathsf{AND} \ \langle ?v, \mathsf{knows}, \mathsf{Tim}\rangle\big)$. *The right sub-pattern* $P_{\mathsf{E2}} = \langle ?v, \mathsf{knows}, \mathsf{Tim}\rangle$ *is* not *Web-safe because evaluating it completely over the WWW is not feasible under context-based semantics (cf. Example 2). However, the larger pattern* P_{E3} *is Web-safe; it can be evaluated completely under context-based semantics. For instance, a possible algorithm may first evaluate the left sub-pattern, which is feasible because it requires the lookup of a single IRI only (the IRI* Bob*). Thereafter, the evaluation of the right sub-pattern* P_{E2} *can be reduced to looking up a finite number of IRIs only, namely the IRIs bound to variable* $?v$ *in solution mappings obtained for the left sub-pattern. Although any other IRI* u^* *might also be used to discover matching triples for* P_{E2}*, 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 4.2). Therefore, the solution mappings resulting from such matching 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* AND *in pattern* P_{E3}.

The example illustrates that some graph patterns are Web-safe even if some of their sub-patterns are not. Consequently, we are interested in a *decidable* property that enables to identify Web-safe patterns, 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 $P_S = \big(\mathsf{SERVICE}\ ?v\ P\big)$ are allowed [6]. Here, 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 tackle this problem, Buil-Aranda et al. introduce a notion of *strong 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 set of *strongly bound variables* in a graph pattern P, denoted by $\mathtt{SBV}(P)$, is defined recursively as follows:

- If P is a PP pattern, then $\mathtt{SBV}(P) = \mathtt{V}(P)$ (recall that $\mathtt{V}(P)$ are all variables in P).
- If P is of the form $(P_1 \ \mathsf{AND}\ P_2)$, then $\mathtt{SBV}(P) = \mathtt{SBV}(P_1) \cup \mathtt{SBV}(P_2)$.
- If P is of the form $(P_1 \ \mathsf{UNION}\ P_2)$, then $\mathtt{SBV}(P) = \mathtt{SBV}(P_1) \cap \mathtt{SBV}(P_2)$.
- If P is of the form $(P_1 \ \mathsf{OPT}\ P_2)$, then $\mathtt{SBV}(P) = \mathtt{SBV}(P_1)$.

The idea behind the notion of strongly bound variables has already been used in earlier work (e.g., *"certain variables"* [23], *"output variables"* [24]), and it is tempting to adopt it for our problem. However, we note that one cannot identify Web-safe graph patterns by using strong boundedness in a manner similar to its use in Buil-Aranda et al.'s work alone. For instance, consider graph pattern P_{E3} from Example 3. We know that (i) P_{E3} is Web-safe and that (ii) $\mathtt{V}(P_{\mathsf{E3}}) = \{?v\}$ and also $\mathtt{SBV}(P_{\mathsf{E3}}) = \{?v\}$. Then, one might hypothesize that for every graph pattern P, if $\mathtt{SBV}(P) = \mathtt{V}(P)$, then P is Web-safe. However, the PP pattern $P_{\mathsf{E2}} = \langle ?v, \mathrm{knows}, \mathrm{Tim}\rangle$ disproves such a hypothesis because, even if $\mathtt{SBV}(P_{\mathsf{E2}}) = \mathtt{V}(P_{\mathsf{E2}})$, pattern P_{E2} is not Web-safe (cf. Example 2).

We conjecture the following reason why strong boundedness cannot be used directly for our problem. For complex patterns (i.e., patterns that are not PP patterns), the sets of strongly bound variables of all sub-patterns are defined *independent* from each other, whereas the algorithm outlined in Example 3 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 Web-bounded variables*, the definition of which, for complex graph patterns, is based on specific relationships between sub-patterns. This notion shall turn out to be suitable for our case.

Definition 5. *The* conditionally Web-bounded variables *of a graph pattern P w.r.t. a set of variables X is the subset $\mathit{CBV}(P \mid X) \subseteq \mathit{V}(P)$ that is defined recursively as follows:*

If P is:	*then $\mathit{CBV}(P \mid X)$ is:*
1) $\langle \alpha, u, \beta \rangle$ *or* $\langle \alpha, !(u_1 \mid \ldots \mid u_n), \beta \rangle$ *such that* $\alpha \in (\mathcal{I} \cup \mathcal{L})$ *or* $\alpha \in X$	$\mathit{V}(P)$
2) $\langle \alpha, u, \beta \rangle$ *or* $\langle \alpha, !(u_1 \mid \ldots \mid u_n), \beta \rangle$ *such that* $\alpha \notin (\mathcal{I} \cup \mathcal{L})$ *and* $\alpha \notin X$	$\emptyset$
3) $\langle \alpha, (\mathtt{path})^*, \beta \rangle$ *s.t.* $\alpha \in \mathcal{V}$ *and* $\beta \notin \mathcal{V}$	$\mathit{CBV}\big(\langle \beta, (\hat{}\mathtt{path})^*, \alpha \rangle \mid X\big)$
4) $\langle \alpha, (\mathtt{path})^*, \beta \rangle$ *s.t. (i)* $\alpha \notin \mathcal{V}$ *or* $\beta \in \mathcal{V}$*, and (ii) for any two variables* $?x, ?y \in \mathcal{V}$ *it holds that* $\mathit{CBV}\big(\langle ?x, \mathtt{path}, ?y \rangle \mid \{?x\}\big) = \{?x, ?y\}$	$\mathit{CBV}\big(\langle \alpha, \mathtt{path}, \beta \rangle \mid X\big)$
5) $\langle \alpha, (\mathtt{path})^*, \beta \rangle$ *such that none of the above*	$\emptyset$
6) $\langle \alpha, \hat{}\mathtt{path}, \beta \rangle$ *with* $P' = \langle \beta, \mathtt{path}, \alpha \rangle$	$\mathit{CBV}(P' \mid X)$
7) $\langle \alpha, (\mathtt{path}_1 \vert \mathtt{path}_2), \beta \rangle$ *with* $P' = \big(\langle \alpha, \mathtt{path}_1, \beta \rangle$ UNION $\langle \alpha, \mathtt{path}_2, \beta \rangle\big)$	$\mathit{CBV}(P' \mid X)$
8) $\langle \alpha, \mathtt{path}_1 / \mathtt{path}_2, \beta \rangle$ *s.t., for any* $?v \in \mathcal{V} \setminus \big(X \cup \{\alpha, \beta\}\big)$, $?v \in \mathit{CBV}(P' \mid X)$ *where* $P' = \big(\langle \alpha, \mathtt{path}_1, ?v \rangle$ AND $\langle ?v, \mathtt{path}_2, \beta \rangle\big)$	$\mathit{CBV}(P' \mid X) \setminus \{?v\}$
9) $\langle \alpha, \mathtt{path}_1 / \mathtt{path}_2, \beta \rangle$ *such that none of the above*	$\emptyset$
10) $(P_1$ AND $P_2)$ *s.t.* $\mathit{CBV}(P_1 \mid X) = \mathit{V}(P_1)$ *and* $\mathit{CBV}(P_2 \mid X) = \mathit{V}(P_2)$	$\mathit{V}(P)$
11) $(P_1$ AND $P_2)$ *s.t.* $\mathit{CBV}(P_1 \mid X) = \mathit{V}(P_1)$ *and* $\mathit{CBV}(P_2 \mid X \cup \mathit{SBV}(P_1)) = \mathit{V}(P_2)$	$\mathit{V}(P)$
12) $(P_1$ AND $P_2)$ *s.t.* $\mathit{CBV}(P_2 \mid X) = \mathit{V}(P_2)$ *and* $\mathit{CBV}(P_1 \mid X \cup \mathit{SBV}(P_2)) = \mathit{V}(P_1)$	$\mathit{V}(P)$
13) $(P_1$ AND $P_2)$ *such that none of the above*	$\emptyset$
14) $(P_1$ UNION $P_2)$	$\mathit{CBV}(P_1 \mid X) \cap \mathit{CBV}(P_2 \mid X)$
15) $(P_1$ OPT $P_2)$ *s.t.* $\mathit{CBV}(P_1 \mid X) = \mathit{V}(P_1)$ *and* $\mathit{CBV}(P_2 \mid X) = \mathit{V}(P_2)$	$\mathit{V}(P)$
16) $(P_1$ OPT $P_2)$ *s.t.* $\mathit{CBV}(P_1 \mid X) = \mathit{V}(P_1)$ *and* $\mathit{CBV}(P_2 \mid X \cup \mathit{SBV}(P_1)) = \mathit{V}(P_2)$	$\mathit{V}(P)$
17) $(P_1$ OPT $P_2)$ *such that none of the above*	$\emptyset$

Example 4. *For the PP pattern $P_{\mathsf{E2}} = \langle ?v, \mathsf{knows}, \mathsf{Tim} \rangle$—which is* not *Web-safe (as discussed in Example 2)—if we use the set $\{?v\}$ as condition, then, by line 1 in Definition 5, it holds that $\mathit{CBV}\big(P_{\mathsf{E2}} \,\big|\, \{?v\}\big) = \{?v\}$. However, if we use the empty set instead, we obtain $\mathit{CBV}(P_{\mathsf{E2}} \mid \emptyset) = \emptyset$ (cf. line 2 in Definition 5).*

While for the non-Web-safe pattern P_{E2} we thus observe $\mathit{CBV}(P_{\mathsf{E2}} \mid \emptyset) \neq \mathit{V}(P_{\mathsf{E2}})$, for graph pattern $P_{\mathsf{E3}} = \big(\langle \mathsf{Bob}, \mathsf{knows}, ?v \rangle$ AND $\langle ?v, \mathsf{knows}, \mathsf{Tim} \rangle\big)$—which is Web-safe (cf. Example 3)—we have $\mathit{CBV}(P_{\mathsf{E3}} \mid \emptyset) = \mathit{V}(P_{\mathsf{E3}})$. The fact that $\mathit{CBV}(P_{\mathsf{E3}} \mid \emptyset) = \{?v\}$ follows from (i) $\mathit{CBV}\big(\langle \mathsf{Bob}, \mathsf{knows}, ?v \rangle \,\big|\, \emptyset\big) = \{?v\}$, (ii) $\mathit{SBV}(\langle \mathsf{Bob}, \mathsf{knows}, ?v \rangle) = \{?v\}$, (iii) $\mathit{CBV}\big(\langle ?v, \mathsf{knows}, \mathsf{Tim} \rangle \,\big|\, \{?v\}\big) = \{?v\}$, and (iv) line 11 in Definition 5.

The example seems to suggest that, if *all* variables of a graph pattern are conditionally Web-bounded w.r.t. the empty set of variables, then the graph pattern is Web-safe. The following result verifies this hypothesis.

Theorem 1. *A graph pattern P is Web-safe if $\mathit{CBV}(P \mid \emptyset) = \mathit{V}(P)$.*

Note 1. *Due to the recursive nature of Definition 5, the condition $\mathit{CBV}(P \mid \emptyset) = \mathit{V}(P)$ (as used in Theorem 1) is decidable for any graph pattern P.*

We prove Theorem 1 based on 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 WoLD W that takes a solution mapping μ as input and returns only the solutions for P over W that are compatible with μ.

Definition 6. *Let P be a graph pattern, let W be a WoLD, and let $\langle \Omega, card \rangle = [\![P]\!]_W^{\mathtt{ctx}}$. Given a solution mapping μ, the* μ-restricted evaluation *of P over W under* context-based semantics*, denoted by $[\![P \,|\, \mu]\!]_W^{\mathtt{ctx}}$, is the multiset of solution mappings $\langle \Omega', card' \rangle$ with $\Omega' = \{\mu' \in \Omega \,|\, \mu' \sim \mu\}$ and $card'(\mu') = card(\mu')$ for all $\mu' \in \Omega'$.*

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

Lemma 1. *Let P be a graph pattern and let μ_{in} be a solution mapping. If it holds that $\mathtt{CBV}(P \,|\, \mathrm{dom}(\mu_{\mathsf{in}})) = \mathtt{V}(P)$, there exists an algorithm that, for any finite WoLD W, computes $[\![P \,|\, \mu_{\mathsf{in}}]\!]_W^{\mathtt{ctx}}$ by looking up a finite number of IRIs only.*

Before providing the proof of the lemma (and of Theorem 1), we point out two important properties of Definition 6. First, it is easily seen that, for any graph pattern P and WoLD W, $[\![P \,|\, \mu_\emptyset]\!]_W^{\mathtt{ctx}} = [\![P]\!]_W^{\mathtt{ctx}}$, where $\mu_\emptyset$ is the empty solution mapping (i.e., $\mathrm{dom}(\mu_\emptyset) = \emptyset$). Consequently, given an algorithm, say A, that has the properties of the algorithm described by Lemma 1, a trivial algorithm that can be used to prove Theorem 1 may simply call algorithm A with the empty solution mapping and return the result of this call (we shall elaborate more on this approach in the proof of Theorem 1 below). Second, for any PP pattern $\langle \alpha, \mathtt{path}, \beta \rangle$ and WoLD W, if α is a variable and $\mathtt{path}$ is a base PP expression (i.e., one of the first two cases in the grammar in Section 3.1), then $[\![P \,|\, \mu]\!]_W^{\mathtt{ctx}}$ is empty for every solution mapping μ that binds (variable) α to a literal or a blank node. Formally, we show the latter as follows.

Lemma 2. *Let P be a PP pattern of the form $\langle ?v, u, \beta \rangle$ or $\langle ?v, !(u_1 \mid \cdots \mid u_n), \beta \rangle$ with $?v \in \mathcal{V}$ and $u, u_1, \ldots, u_n \in \mathcal{I}$, and let μ be a solution mapping. If $?v \in \mathrm{dom}(\mu)$ and $\mu(?v) \in (\mathcal{B} \cup \mathcal{L})$, then, for any WoLD W, $[\![P \,|\, \mu]\!]_W^{\mathtt{ctx}}$ is the empty multiset.*

Proof (Lemma 2). Recall that, for any IRI u and any WoLD W, context $\mathrm{C}^W(u)$ contains only triples that have IRI u as their subject. As a consequence, for any WoLD W, every solution mapping $\mu' \in [\![P]\!]_W^{\mathtt{ctx}}$ binds variable $?v$ to some IRI (and never to a literal or blank node); i.e., $\mu'(?v) \in \mathcal{I}$. Therefore, if $?v \in \mathrm{dom}(\mu)$ and $\mu(?v) \in (\mathcal{B} \cup \mathcal{L})$, then μ cannot be compatible with any $\mu' \in [\![P]\!]_W^{\mathtt{ctx}}$ and, thus, $[\![P \,|\, \mu]\!]_W^{\mathtt{ctx}}$ is empty. □

We use Lemma 2 to prove Lemma 1 as follows.

Proof idea (Lemma 1). We prove the lemma by induction on the possible structure of graph pattern P. For the proof, we provide Algorithm 1 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). Due to space limitations, in this paper we only present a fragment of the algorithm and highlight essential properties thereof. The given fragment covers the base case (lines 1-11) and one pivotal case of the induction step, namely, graph patterns of the form $(P_1 \ \mathsf{AND} \ P_2)$ (lines 57-72). The complete version of the algorithm and the full proof can be found in an extended version of this paper [15].

For the base case, Algorithm 1 looks up at most one IRI (cf. lines 2-5). The crux of showing that the returned result is sound and complete is Lemma 2 and the fact that the only possible *context* in which a triple $\langle s, p, o \rangle$ with $s \in \mathcal{I}$ can be found is $\mathrm{C}^W(s)$.

For PP patterns of the form $(P_1 \ \mathsf{AND} \ P_2)$ consider lines 57-72. By using Definition 5, we show $\mathtt{CBV}(P_i \,|\, \mathrm{dom}(\mu_{\mathsf{in}})) = \mathtt{V}(P_i)$ and $\mathtt{CBV}(P_j \,|\, \mathrm{dom}(\mu_{\mathsf{in}}) \cup \mathrm{dom}(\mu)) = \mathtt{V}(P_j)$

Algorithm 1 *EvalCtxBased*(P, μ_{in}), which computes $[\![P \,|\, \mu_{\mathsf{in}}]\!]^{\mathtt{ctx}}_W$.

1: **if** P is of the form $\langle \alpha, u, \beta \rangle$ **or** P is of the form $\langle \alpha, !(u_1 \mid \cdots \mid u_n), \beta \rangle$ **then**
2: **if** $\alpha \in \mathcal{I}$ **then** $u' := \alpha$
3: **else if** $\alpha \in \mathcal{V}$ **and** $\alpha \in \mathrm{dom}(\mu_{\mathsf{in}})$ **and** $\mu_{\mathsf{in}}(\alpha) \in \mathcal{I}$ **then** $u' := \mu_{\mathsf{in}}(\alpha)$
4: **else** $u' := \mathtt{null}$
5: **if** u' is an IRI and looking it up results in retrieving a document, say d **then**
6: $G :=$ the set of triples in d (use a fresh set of blank node identifiers when parsing d)
7: $G' := \{\langle s, p, o \rangle \in G \,|\, s = u'\}$
8: $\langle \Omega, \mathit{card} \rangle := [\![P]\!]_{G'}$ ($[\![P]\!]_{G'}$ can be computed by using any algorithm that implements the standard SPARQL evaluation function)
9: **return** a new multiset $\langle \Omega', \mathit{card}' \rangle$ with $\Omega' = \{\mu' \in \Omega \,|\, \mu' \sim \mu_{\mathsf{in}}\}$ and $\mathit{card}'(\mu') = \mathit{card}(\mu')$ for all $\mu' \in \Omega'$
10: **else**
11: **return** a new empty multiset $\langle \Omega, \mathit{card} \rangle$ with $\Omega = \emptyset$ and $\mathrm{dom}(\mathit{card}) = \emptyset$
...
57: **else if** P is of the form $(P_1 \ \mathsf{AND} \ P_2)$ **then**
58: **if** $\mathtt{CBV}\big(P_1 \,|\, \mathrm{dom}(\mu_{\mathsf{in}})\big) = \mathtt{V}(P_1)$ **then** $i := 1$; $j := 2$ **else** $i := 2$; $j := 1$
59: Create a new empty multiset $M = \langle \Omega, \mathit{card} \rangle$ with $\Omega = \emptyset$ and $\mathrm{dom}(\mathit{card}) = \emptyset$
60: $\langle \Omega^{P_i}, \mathit{card}^{P_i} \rangle :=$ *EvalCtxBased*(P_i, μ_{in})
61: **for all** $\mu \in \Omega^{P_i}$ **do**
62: $\langle \Omega^{\mu}, \mathit{card}^{\mu} \rangle :=$ *EvalCtxBased*$(P_j, \mu_{\mathsf{in}} \cup \mu)$
63: **for all** $\mu' \in \Omega^{\mu}$ **do**
64: $\mu^* := \mu \cup \mu'$
65: $k := \mathit{card}^{P_i}(\mu) \cdot \mathit{card}^{\mu}(\mu')$
66: **if** $\mu^* \in \Omega$ **then**
67: $\mathit{old} := \mathit{card}(\mu^*)$
68: Adjust card such that $\mathit{card}(\mu^*) = k + \mathit{old}$
69: **else**
70: Adjust card such that $\mathit{card}(\mu^*) = k$
71: Add μ^* to Ω
72: **return** M

for all $\mu \in \Omega^{P_i}$. Therefore, by induction, all recursive calls (lines 60 and 62) look up a finite number of IRIs and return correct results; i.e., $\langle \Omega^{P_i}, \mathit{card}^{P_i} \rangle = [\![P_i \,|\, \mu_{\mathsf{in}}]\!]^{\mathtt{ctx}}_W$ and $\langle \Omega^{\mu}, \mathit{card}^{\mu} \rangle = [\![P_j \,|\, \mu_{\mathsf{in}} \cup \mu]\!]^{\mathtt{ctx}}_W$ for all $\mu \in \Omega^{P_i}$. Then, since each $\mu \in \Omega^{P_i}$ is compatible with all $\mu' \in \Omega^{\mu}$ and all processed solution mappings are compatible with μ_{in}, it is easily verified that the computed result is $[\![(P_1 \ \mathsf{AND} \ P_2) \,|\, \mu_{\mathsf{in}}]\!]^{\mathtt{ctx}}_W$. □

We are now ready to prove Theorem 1, for which we use Lemma 1, or more precisely the algorithm that we introduce in the proof of the lemma.

Proof (Theorem 1). Let P be a graph pattern s.t. $\mathtt{CBV}(P \,|\, \emptyset) = \mathtt{V}(P)$. Then, given the empty solution mapping $\mu_\emptyset$ with $\mathrm{dom}(\mu_\emptyset) = \emptyset$, we have $\mathtt{CBV}\big(P \,\big|\, \mathrm{dom}(\mu_\emptyset)\big) = \mathtt{V}(P)$. Therefore, by our proof of Lemma 1 we know that, for any finite WoLD W, Algorithm 1 computes $[\![P \,|\, \mu_\emptyset]\!]^{\mathtt{ctx}}_W$ by looking up a finite number of IRIs. We also know that the empty solution mapping is compatible with any solution mapping. Consequently, by Definition 6, $[\![P \,|\, \mu_\emptyset]\!]^{\mathtt{ctx}}_W = [\![P]\!]^{\mathtt{ctx}}_W$ for any WoLD W. Hence, by passing the empty solu-

tion mapping to it, Algorithm 1 can be used to compute $[\![P]\!]_W^{\mathtt{ctx}}$ for any finite WoLD W, and during this computation the algorithm looks up a finite number of IRIs only. □

While the condition in Theorem 1 is sufficient to identify Web-safe graph patterns, the question that remains is whether it is a necessary condition (in which case it could be used to decide Web-safeness of *all* graph patterns). Unfortunately, the answer is no.

Example 5. *Consider the graph pattern $P = (P_1 \text{ UNION } P_2)$ with $P_1 = \langle u_1, p_1, ?x\rangle$ and $P_2 = \langle u_2, p_2, ?y\rangle$. We note that $\mathtt{CBV}(P_1 \mid \emptyset) = \{?x\}$ and $\mathtt{CBV}(P_2 \mid \emptyset) = \{?y\}$, and, thus, $\mathtt{CBV}(P \mid \emptyset) = \emptyset$. Hence, the pattern does not satisfy the condition in Theorem 1. Nonetheless, it is easy to see that there exists a (sound and complete) algorithm that, for any WoLD W, computes $[\![P]\!]_W^{\mathtt{ctx}}$ by looking up a finite number of IRIs only. For instance, such an algorithm, say A, may first use two other algorithms that compute $[\![P_1]\!]_W^{\mathtt{ctx}}$ and $[\![P_2]\!]_W^{\mathtt{ctx}}$ by looking up a finite number of IRIs, respectively. Such algorithms exist by Theorem 1, because $\mathtt{CBV}(P_1 \mid \emptyset) = \mathtt{V}(P_1)$ and $\mathtt{CBV}(P_2 \mid \emptyset) = \mathtt{V}(P_2)$. Finally, algorithm A can generate the (sound and complete) query result $[\![P]\!]_W^{\mathtt{ctx}}$ by computing the multiset union $[\![P_1]\!]_W^{\mathtt{ctx}} \sqcup [\![P_2]\!]_W^{\mathtt{ctx}}$, which requires no additional IRI lookups.*

Remark 1. *The example illustrates that "only if" cannot be shown in Theorem 1. It remains an open question whether there exists an alternative condition for Web-safeness that is both sufficient and necessary (and decidable).*

6 Concluding Remarks and Future Work

This paper studies the problem of extending the scope of SPARQL property paths to query Linked Data that is distributed on the WWW. We have proposed a context-based query semantics and analyzed its peculiarities. Our perhaps most interesting finding is that there exist queries whose evaluation over the WWW is not feasible. We studied this aspect and introduced a decidable syntactic property for identifying feasible queries.

We believe that the presented work provides valuable input to a wider discussion about defining a language for accessing Linked Data on the WWW. In this context, there are several directions for future research such as the following three. First, studying a more expressive navigational core for property paths over the Web; e.g., along the lines of other navigational languages such as nSPARQL [21] or NautiLOD [8]. Second, investigating relationships between navigational queries and SPARQL federation. Third, while the aim of this paper was to introduce a formal foundation for answering SPARQL queries with PPs over Linked Data on the WWW, an investigation of how systems may implement efficiently the machinery developed in this paper is certainly interesting.

References

1. Abiteboul, S., Vianu, V.: Queries and Computation on the Web. Theor. Comput. Sci. 239(2), 231–255 (2000)
2. Alkhateeb, F., Baget, J.F., Euzenat, J.: Extending SPARQL with Regular Expression Patterns (for querying RDF). J. Web Sem. 7(2), 57–73 (2009)

3. Arenas, M., Conca, S., Pérez, J.: Counting Beyond a Yottabyte, or how SPARQL 1.1 Property Paths will Prevent Adoption of the Standard. In: Proceedings of the 21st International Conference on World Wide Web (2012)
4. Berners-Lee, T.: Design issues: Linked Data. Online (Jul 2006)
5. Bouquet, P., Ghidini, C., Serafini, L.: Querying The Web Of Data: A Formal Approach. In: Proceedings of the 4th Asian Semantic Web Conference (2009)
6. Buil-Aranda, C., Arenas, M., Corcho, O., Polleres, A.: Federating Queries in SPARQL1.1: Syntax, Semantics and Evaluation. Journal on Web Semantics 18(1), 1–17 (2013)
7. Fielding, R., Gettys, J., Mogul, J.C., Frystyk, H., Masinter, L., Leach, P.J., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Jun 1999)
8. Fionda, V., Gutierrez, C., Pirrò, G.: Semantic Navigation on the Web of Data: Specification of Routes, Web Fragments and Actions. In: Proceedings of the 21st International Conference on the World Wide Web (2012)
9. Fionda, V., Pirrò, G., Consens, M.: Extended Property Paths: Writing More SPARQL Queries in a Succinct Way. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI) (2015)
10. Fionda, V., Pirrò, G., Gutierrez, C.: NautiLOD: A Formal Language for the Web of Data Graph. ACM Trans. Web 9(1) (Jan 2015)
11. Florescu, D., Levy, A., Mendelzon, A.: Database Techniques for the World-Wide Web: A Survey. SIGMOD Rec. 27, 59–74 (1998)
12. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. W3C Reccomendation (2013)
13. Harth, A., Speiser, S.: On Completeness Classes for Query Evaluation on Linked Data. In: Proceedings of the 26th AAAI Conference (2012)
14. Hartig, O.: SPARQL for a Web of Linked Data: Semantics and Computability. In: Proceedings of the 9th Extended Semantic Web Conference (2012)
15. Hartig, O., Pirrò, G.: A Context-Based Semantics for SPARQL Property Paths over the Web (Extended Version). CoRR abs/1503.04831 (2015), http://arxiv.org/abs/1503.04831
16. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax (2006)
17. Kochut, K.J., Janik, M.: SPARQLeR: Extended SPARQL for Semantic Association Discovery. In: The Semantic Web: Research and Applications. pp. 145–159. Springer (2007)
18. Konopnicki, D., Shmueli, O.: Information Gathering in the World-Wide Web: The W3QL Query Language and the W3QS System. ACM Transactions on Database Systems 23(4), 369–410 (Dec 1998)
19. Loseman, K., Martens, W.: The Complexity of Evaluating Path Expressions in SPARQL. In: Proceedings of the 31st ACM Symposium on Principles of Database Systems (2012)
20. Mendelzon, A.O., Mihaila, G.A., Milo, T.: Querying the World Wide Web. In: 1 (ed.) Int. J. on Digital Libraries. vol. 1, pp. 54–97 (1997)
21. Pérez, J., Arenas, M., Gutierrez, C.: nSPARQL: A Navigational Language for RDF. Journal on Web Semantics 8(4), 255–270 (2010)
22. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. ACM Transactions on Database Systems (TODS) 34(3) (2009)
23. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL Query Optimization. In: Proceedings of the 13th International Conference on Database Theory (2010)
24. Toman, D., Weddell, G.E.: Fundamentals of Physical Design and Query Compilation. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2011)
25. Umbrich, J., Hogan, A., Polleres, A., Decker, S.: Link Traversal Querying for a diverse Web of Data. Semantic Web Journal (2014)
26. Wood, P.T.: Query Languages for Graph Databases. SIGMOD Rec. 41(1) (2012)