LDQL: A Query Language for the Web of Linked Data

Olaf Hartig[1] and Jorge Pérez[2](✉)

[1] Hasso-Plattner-Institute for IT Systems Engineering, Potsdam, Germany
http://olafhartig.de/
[2] Department of Computer Science, Universidad de Chile, Santiago, Chile
jperez@dcc.uchile.cl

Abstract. The Web of Linked Data is composed of tons of RDF documents interlinked to each other forming a huge repository of distributed semantic data. Effectively querying this distributed data source is an important open problem in the Semantic Web area. In this paper, we propose LDQL, a declarative language to query Linked Data on the Web. One of the novelties of LDQL is that it expresses separately (i) patterns that describe the expected query result, and (ii)Web navigation paths that select the data sources to be used for computing the result. We present a formal syntax and semantics, prove equivalence rules, and study the expressiveness of the language. In particular, we show that LDQL is strictly more expressive than the query formalisms that have been proposed previously for Linked Data on the Web. The high expressiveness allows LDQL to define queries for which a complete execution is not computationally feasible over the Web. We formally study this issue and provide a syntactic sufficient condition to avoid this problem; queries satisfying this condition are ensured to have a procedure to be effectively evaluated over the Web of Linked Data.

1 Introduction

In recent years an increasing amount of structured data has been published and interlinked on the World Wide Web (WWW) in adherence to the Linked Data principles [3]. These principles are based on standard Web technologies. In particular, (i) the Hypertext Transfer Protocol (HTTP) is used to access data, (ii) HTTP-based Uniform Resource Identifiers (URIs) are used as identifiers for entities described in the data, and (iii) the Resource Description Framework (RDF) is used as data model. Then, any HTTP URI in an RDF triple presents a *data link* that enables software clients to retrieve more data by looking up the URI with an HTTP request. The adoption of these principles has lead to the creation of a globally distributed dataspace: the *Web of Linked Data*.

The emergence of the Web of Linked Data makes possible an *online execution* of declarative queries over up-to-date data from a virtually unbounded set of data sources, each of which is readily accessible without any need for implementing source-specific APIs or wrappers. This possibility has spawned

DOI: 10.1007/978-3-319-25007-6_5

research interest in approaches to query Linked Data on the WWW as if it was a single (distributed) database. For an overview on query execution techniques proposed in this context refer to [12].

The main contribution of this paper is the proposal of LDQL, a novel query language for the Web of Linked Data. The most important feature of LDQL is that it clearly separates query components for selecting query-relevant regions of the Web of Linked Data, from components for specifying the query result that has to be constructed from the data in the selected regions. The most basic construction in LDQL are tuples of the form $\langle L, Q \rangle$ where L is an expression used to select a set of relevant documents, and Q is a query intended to be executed over the data in these documents as if they were a single RDF repository. In an abstract setting one can use several formalisms to express L and Q. In our proposal, for the former part we introduce the notion of *link path expressions* that are a form of nested regular expressions (with some other important features) used to navigate the link graph of the Web. For the latter, we use standard SPARQL graph patterns. To begin evaluating these queries one needs to specify a set of seed URIs. The language also possesses features to dynamically (at query time) identify new seed URIs to evaluate portions of a query. Additionally, such queries can be combined by using conjunctions, disjunctions, and projection. We present a formal syntax and semantics for LDQL, propose some rewrite rules, and study its expressive power.

While there does not exist a standard language for expressing queries over Linked Data on the WWW, a few options have been proposed. In particular, a first strand of research focuses on extending the scope of SPARQL such that an evaluation of SPARQL queries over Linked Data has a well-defined semantics [9, 11,14,18]. A second strand of research focuses on navigational languages [7,14]. Although these languages have different motivations, a commonality of all these proposals is that, in contrast to LDQL, the definition of query-relevant regions of the Web of Linked Data and the definition of query-relevant data within the specified regions are mixed.

As our second main contribution we compare LDQL with three previously proposed formalisms for querying the Web of Linked Data: *SPARQL under reachability-based query semantics* [11], *NautiLOD* [7], and *SPARQL Property Path patterns under context-based semantics* [14]. We formally prove that LDQL is strictly more expressive than every one of these. We show that for every query Q in the previous languages, one can effectively construct an LDQL query which is equivalent to Q. Moreover, for every one of the previous languages, there exists an LDQL query that cannot be expressed in that language. These results show that LDQL presents an interesting expressive power.

The downside of the expressiveness provided by LDQL is the existence of queries for which a complete execution is not feasible in practice. To capture this issue formally, we define a notion of *Web-safeness* for LDQL queries. Then, the obvious question that arises is how to identify LDQL queries that are Web-safe. Our last technical contribution is the identification of a sufficient syntactic condition for Web-safeness.

The rest of the paper is structured as follows. Section 2 introduces a data model that provides the basis for defining the semantics of LDQL. In Section 3 we formally define the syntax and semantics of LDQL and show some simple algebraic properties. In Section 4 we compare LDQL with the three mentioned languages, and in Section 5 we focus on Web-safeness. Section 6 concludes the paper and sketches future work. Proofs of the formal results in this paper can be found in an extended version of the paper [13].

A preliminary version of some of the results in this paper have been presented in a workshop [10]. This paper is a substantial extension of [10] refining the definition of LDQL and introducing important changes to the syntax and the semantics of the language. Moreover, the comparison with previous proposals was not discussed in [10].

2 Data Model

In this section we introduce a structural data model that captures the concept of a Web of Linked Data formally. As usual [7,9,11,14,18], for the definitions and analysis in this paper, we assume that the Web is fixed during the execution of any single query.

We use the RDF data model [5] as a basis for our model of a Web of Linked Data. That is, we assume three pairwise disjoint, infinite sets $\mathcal{U}$ (URIs), $\mathcal{B}$ (blank nodes), and $\mathcal{L}$ (literals). An *RDF triple* is a tuple $\langle s, p, o \rangle \in \mathcal{T}$ with $\mathcal{T} = (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. For any RDF triple $t = \langle s, p, o \rangle$ we write $\mathrm{uris}(t)$ to denote the set of all URIs in t.

Additionally, we assume another infinite set $\mathcal{D}$ that is disjoint from $\mathcal{U}$, $\mathcal{B}$, and $\mathcal{L}$, respectively. We refer to elements in this set as *documents* and use them to represent the concept of Web documents from which Linked Data can be extracted. Hence, we assume a function, say data, that maps each document $d \in \mathcal{D}$ to a finite set of RDF triples $\mathrm{data}(d) \subseteq \mathcal{T}$ such that the data of each document uses a unique set of blank nodes.

Given these preliminaries, we are ready to define a *Web of Linked Data*.

Definition 1. A **Web of Linked Data** is a tuple $W = \langle D, adoc \rangle$ that consists of a set of documents $D \subseteq \mathcal{D}$ and a partial function $adoc : \mathcal{U} \to D$ that is surjective.

Function *adoc* of a Web of Linked Data $W = \langle D, adoc \rangle$ captures the relationship between the URIs that can be looked up in this Web and the documents that can be retrieved by such lookups. Since not every URI can be looked up, the function is partial. For any URI $u \in \mathcal{U}$ with $u \in \mathrm{dom}(adoc)$ (i.e., any URI that can be looked up in W), document $d = adoc(u)$ can be considered the authoritative source of data for u in W (hence, the name *adoc*). To accommodate for documents that are authoritative for multiple URIs, we do not require injectivity for function *adoc*. However, we require surjectivity because we conceive documents as irrelevant for a Web of Linked Data if they cannot be retrieved by any URI lookup in this Web.

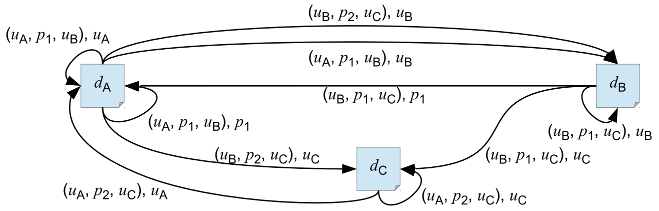

Fig. 1. The link graph $\mathcal{G}_{W_{\mathsf{ex}}}$ of our example Web of Linked Data W_{ex}.

Let $W = \langle D, adoc \rangle$ be a Web of Linked Data. W is said to be finite [11] if its set D of documents is finite. In this paper we assume that every Web of Linked Data is finite. Given documents $d, d' \in D$ and a triple $t \in \mathrm{data}(d)$, we say that a URI $u \in \mathrm{uris}(t)$ establishes a *data link* from d to d', if $adoc(u) = d'$. As a final concept, we formalize the notion of a *link graph* associated to W. This graph has documents in D as nodes, and directed edges representing data links between documents. Each edge is associated with a label that identifies both the particular RDF triple and the URI in this triple that establishes the corresponding data link. These labels shall provide the basis for defining the navigational component of our query language.

Definition 2. The **link graph** of a Web of Linked Data $W = \langle D, adoc \rangle$, is a directed, edge-labeled multigraph, $\mathcal{G}_W = \langle D, E_W \rangle$, with set of edges $E_W \subseteq D \times (\mathcal{T} \times \mathcal{U}) \times D$ defined as $E_W = \big\{ \langle d_{\mathsf{src}}, (t,u), d_{\mathsf{tgt}} \rangle \mid t \in \mathrm{data}(d_{\mathsf{src}}), u \in \mathrm{uris}(t) \text{ and } d_{\mathsf{tgt}} = adoc(u) \big\}$.

For a link graph edge $e = \langle d_{\mathsf{src}}, (t,u), d_{\mathsf{tgt}} \rangle$, tuple (t,u) is the label of e. Moreover, we sometimes write $e \in \mathcal{G}_W$ to denote that e is an edge in the link graph $\mathcal{G}_W$.

Example 1. As a running example for this paper assume a simple Web of Linked Data $W_{\mathsf{ex}} = \langle D_{\mathsf{ex}}, adoc_{\mathsf{ex}} \rangle$ with three documents, d_{A}, d_{B}, and d_{C} (i.e., $D_{\mathsf{ex}} = \{d_{\mathsf{A}}, d_{\mathsf{B}}, d_{\mathsf{C}}\}$). The data in these documents are the following sets of RDF triples:

$$\begin{aligned} \mathrm{data}(d_{\mathsf{A}}) &= \{\langle u_{\mathsf{A}}, p_1, u_{\mathsf{B}} \rangle, & \mathrm{data}(d_{\mathsf{B}}) &= \{\langle u_{\mathsf{B}}, p_1, u_{\mathsf{C}} \rangle\}; \\ &\quad\ \langle u_{\mathsf{B}}, p_2, u_{\mathsf{C}} \rangle\}; & \mathrm{data}(d_{\mathsf{C}}) &= \{\langle u_{\mathsf{A}}, p_2, u_{\mathsf{C}} \rangle\}; \end{aligned}$$

and for function $adoc_{\mathsf{ex}}$ we have: $adoc_{\mathsf{ex}}(u_{\mathsf{A}}) = d_{\mathsf{A}}$, $adoc_{\mathsf{ex}}(u_{\mathsf{B}}) = d_{\mathsf{B}}$, $adoc_{\mathsf{ex}}(u_{\mathsf{C}}) = d_{\mathsf{C}}$, and $adoc_{\mathsf{ex}}(p_1) = d_{\mathsf{A}}$ (i.e., $\mathrm{dom}(adoc_{\mathsf{ex}}) = \{u_{\mathsf{A}}, u_{\mathsf{B}}, u_{\mathsf{C}}, p_1\}$). This Web contains 10 data links. For instance, URI u_{A} in the RDF triple $\langle u_{\mathsf{A}}, p_2, u_{\mathsf{C}} \rangle \in \mathrm{data}(d_{\mathsf{C}})$ establishes a data link to document d_{A}. Hence, the corresponding edge in the link graph of W_{ex} is $\big\langle d_{\mathsf{C}}, (\langle u_{\mathsf{A}}, p_2, u_{\mathsf{C}} \rangle, u_{\mathsf{A}}), d_{\mathsf{A}} \big\rangle$. Figure 1 illustrates the link graph $\mathcal{G}_{W_{\mathsf{ex}}}$ with all 10 edges.

3 Definition of LDQL

This section defines our Linked Data query language, LDQL. LDQL queries are meant to be evaluated over a Web of Linked Data and each such query is built from two types of components: *Link path expressions* (*LPEs*) for selecting query-relevant documents of the queried Web of Linked Data; and SPARQL graph patterns for specifying the query result that has to be constructed from the data in the selected documents. For this paper, we assume that the reader is familiar with the definition of SPARQL [8], including the algebraic formalization introduced in [2,16]. In particular, for SPARQL graph patterns we closely follow the formalization in [2] considering operators AND, OPT, UNION, FILTER, and GRAPH, plus the operator BIND defined in [8]. We begin this section by introducing the most basic concept of our language, the notion of link patterns. We use link patterns as the basis for navigating the link graph of a Web of Linked Data.

3.1 Link Patterns

A link pattern is a tuple in $\big(\mathcal{U} \cup \{\_, +\}\big) \times \big(\mathcal{U} \cup \{\_, +\}\big) \times \big(\mathcal{U} \cup \mathcal{L} \cup \{\_, +\}\big)$. Link patterns are used to match link graph edges in the context of a designated *context* URI. The special symbol $+$ denotes a placeholder for the context URI. The special symbol $\_$ denotes a wildcard that will drive the direction of the navigation. Before formalizing how link graph edges actually match link patterns, we show some intuition. Consider the link graph of Web W_{ex} in Example 1 (see Fig. 1), and the link pattern $\langle +, p_1, \_\rangle$. Intuitively, in the context of URI u_{A}, the edge with label $(\langle u_{\mathsf{A}}, p_1, u_{\mathsf{B}}\rangle, u_{\mathsf{B}})$ from document d_{A} to document d_{B}, matches the link pattern $\langle +, p_1, \_\rangle$. Notice that in the matching, the context URI u_{A} takes the place of symbol $+$, and u_{B} takes the place of the wildcard symbol $\_$. Notice that u_{B} also denotes the direction of the edge that matches the link pattern. On the other hand, the edge with label $(\langle u_{\mathsf{A}}, p_1, u_{\mathsf{B}}\rangle, u_{\mathsf{A}})$ from d_{A} to d_{A}, does not match $\langle +, p_1, \_\rangle$; although u_{B} can take the place of the wildcard symbol $\_$, the direction of the edge is not to u_{B}. That is, when matching an edge labeled by (t, u) we require URI u to be taking the place of a wildcard in the link pattern. When more than one wildcard symbol is used, the link pattern can be matched by edges pointing to the direction of any of the URIs taking the place of a wildcard. For instance, in the context of u_{A}, the link pattern $\langle \_, p_2, \_\rangle$ is matched by edges $\langle d_{\mathsf{A}}, (\langle u_{\mathsf{B}}, p_2, u_{\mathsf{C}}\rangle, u_{\mathsf{B}}), d_{\mathsf{B}}\rangle$ and $\langle d_{\mathsf{A}}, (\langle u_{\mathsf{B}}, p_2, u_{\mathsf{C}}\rangle, u_{\mathsf{C}}), d_{\mathsf{C}}\rangle$. The next definition formalizes this notion of matching.

Definition 3. A link graph edge with label $(\langle x_1, x_2, x_3\rangle, u)$ **matches** a link pattern $\langle y_1, y_2, y_3\rangle$ in the context of a URI u_{ctx} if the following two properties hold:

1. there exists $i \in \{1, 2, 3\}$ such that $y_i = \_$ and $x_i = u$, and
2. for every $i \in \{1, 2, 3\}$ either $y_i = +$ and $x_i = u_{\mathsf{ctx}}$, or $y_i = x_i$, or $y_i = \_$.

One of the rationales for adopting the notion of a context URI and the $+$ symbol in our definition of link patterns, is to support cases in which link graph navigation has to be focused solely on data links that are *authoritative*. A data link represented by link graph edge $\langle d_{\mathsf{src}}, (t, u), d_{\mathsf{tgt}}\rangle \in \mathcal{G}_W$ is authoritative in a Web of Linked Data $W = \langle D, adoc\rangle$ if $d_{\mathsf{src}} = adoc(u')$ for some URI $u' \in \text{uris}(t)$. Thus, if we fix a context URI u_{ctx}, a link pattern that uses the $+$ symbol allows us to follow only authoritative data links from document $d_{\mathsf{ctx}} = adoc(u_{\mathsf{ctx}})$.

3.2 LDQL Queries

The most basic construction in LDQL queries are tuples of the from $\langle L, P\rangle$ where L is an expression used to select a set of documents from the Web of Linked Data, and P is a SPARQL graph pattern to query these documents as if they were a single RDF dataset. In an abstract setting, one can use any formalism to specify L as long as L defines sets of RDF documents. In our proposal we use what we call *link path expressions* (LPEs) that are a form of nested regular expressions [17] over the alphabet of link patterns. Every link path expression begins its navigation in a context URI, traverses the Web, and returns a set of URIs; these URIs are used to construct an RDF dataset with all the documents to be retrieved by looking up the URIs. This dataset is passed to the SPARQL graph pattern to obtain the final evaluation of the whole query. Besides the basic constructions of the form $\langle L, P\rangle$, in LDQL one can also use AND, UNION and projection, to combine them. We also introduce an operator SEED that is used to dynamically change, at query time, the seed URI from which the navigation begins. The next definition formalizes the syntax of LDQL queries and LPEs.

Definition 4. The syntax of LDQL is given by the following production rules in which lp is an arbitrary link pattern, $?v$ is a variable, P is a SPARQL graph pattern (as per [2]), V is a finite set of variables, and U is a finite set of URIs:

$$\begin{aligned} q &:= \langle lpe, P\rangle \mid (\textsf{SEED}\ \ U\ q) \mid (\textsf{SEED}\ \ ?v\ q) \mid (q\ \textsf{AND}\ q) \mid (q\ \textsf{UNION}\ q) \mid \pi_V q \\ lpe &:= \varepsilon \mid lp \mid lpe/lpe \mid lpe|lpe \mid lpe^* \mid [lpe] \mid \langle ?v, q\rangle \end{aligned}$$

Any expression that satisfies the production q is an **LDQL query**, any expression that satisfies the production lpe is a **link path expression** (**LPE**), and any LDQL query of the form $\langle lpe, P\rangle$ is a **basic LDQL query**.

Before going into the formal semantics of LDQL and LPEs, we give some more intuition about how these expressions are evaluated in a Web of Linked Data W. As mentioned before, the most basic expression in LDQL is of the form $\langle lpe, P\rangle$. To evaluate this expression over W we will need a set S of *seed* URIs. When evaluating $\langle lpe, P\rangle$, every one of the seed URIs in S will trigger a navigation of link graph $\mathcal{G}_W$ via the link path expression lpe starting on that seed. That is, the seed URIs are passed to lpe as *context* URIs in which the LPE should be evaluated. These evaluations of lpe will result in a set of URIs that are used to construct a dataset over which P is finally evaluated.

Regarding the navigation of link graph $\mathcal{G}_W$, the most basic form of navigation is to follow a single link graph edge that matches a link pattern lp. When a navigation via a link pattern lp is triggered from a context URI u, we proceed as follows. We first go to the authoritative document for u, that is $adoc(u)$, and try to find outgoing link graph edges that match lp in the context of u (as explained in Section 3.1). Every one of these matches defines a new context URI u' from which the navigation can continue. More complex forms of navigation are obtained by combining link patterns via classical regular expression operators such as concatenation $/$, disjunction $|$, and recursive concatenation $(\cdot)^*$.

The nesting operator $[\cdot]$ is used to test for existence of paths. When a context URI u is passed to an expression $[lpe]$, it checks whether $\mathcal{G}_W$ contains a path from $d_{\mathsf{ctx}} = adoc(u)$ that matches lpe. If such a path exists, the navigation can continue from the same context URI u. The most involved form of navigation is by using the expression $\langle ?v, q\rangle$ with q an LDQL query. To evaluate this expression from context URI u one first has to pass u as a seed URI for q and recursively evaluate q from that seed. This evaluation generates a set of solution mappings, and for every one of these mappings its value on variable $?v$ is used as the new context URI from which the navigation continues. Finally, note that our notion of LPEs does not provide an operator for navigating paths in their inverse direction. The reason for omitting such an operator is that traversing arbitrary data links backwards is impossible on the WWW.

To formally define the semantics of LDQL we need to introduce some terminology. We first define a function $\mathrm{dataset}_W(\cdot)$ that from a set of URIs constructs an RDF dataset with all the documents pointed to by those URIs in W. Formally, given a Web of Linked Data $W = \langle D, adoc\rangle$ and a set U of URIs, $\mathrm{dataset}_W(U)$ is an RDF dataset (as per [2,8]) that has the set of triples $\{t \in \mathrm{data}(adoc(u)) \mid u \in U \cap \mathrm{dom}(adoc)\}$ as default graph. Moreover, for every URI $u \in U \cap \mathrm{dom}(adoc)$, $\mathrm{dataset}_W(U)$ contains the named graph $\langle u, \mathrm{data}(adoc(u))\rangle$.

Example 2. Consider the Web W_{ex} in Example 1 and the set of URIs $U = \{u_{\mathsf{A}}, u_{\mathsf{C}}\}$. Then $\mathrm{dataset}_{W_{\mathsf{ex}}}(U)$ has $\{\langle u_{\mathsf{A}}, p_1, u_{\mathsf{B}}\rangle, \langle u_{\mathsf{B}}, p_2, u_{\mathsf{C}}\rangle, \langle u_{\mathsf{A}}, p_2, u_{\mathsf{C}}\rangle\}$ as default graph, and two named graphs, $\langle u_{\mathsf{A}}, \{\langle u_{\mathsf{A}}, p_1, u_{\mathsf{B}}\rangle, \langle u_{\mathsf{B}}, p_2, u_{\mathsf{C}}\rangle\}\rangle$ and $\langle u_{\mathsf{C}}, \{\langle u_{\mathsf{A}}, p_2, u_{\mathsf{C}}\rangle\}\rangle$.

In the formalization of the semantics of LDQL, we use the standard join operator $\bowtie$ over sets of solution mappings [8,16]. We also make use of the semantics of SPARQL graph patterns over datasets as defined in [2]. In particular, given an RDF dataset $\mathfrak{D}$, an RDF graph G in $\mathfrak{D}$, and a SPARQL graph pattern P, we denote by $[\![P]\!]^{\mathfrak{D}}_G$ the evaluation of P over G in $\mathfrak{D}$ [2, Definition 13.3].

We are now ready to formally define the semantics of LDQL and LPEs. Given a Web of Linked Data W and a set S of URIs, we formalize the evaluation of LDQL queries over W from the seed URIs S, as a function $[\![\cdot]\!]^S_W$ that given an LDQL query, produces a set of solution mappings. Similarly, the evaluation of LPEs over W from a context URI u, is formalized as a function $[\![\cdot]\!]^u_W$ that given an LPE, produces a set of URIs.

Definition 5. Given a finite set $S \subseteq \mathcal{U}$, the **S-based evaluation** of LDQL queries over a Web of Linked Data $W = \langle D, adoc\rangle$, denoted by $[\![\cdot]\!]^S_W$, is defined recursively as follows:

$$\begin{aligned}
[\![\langle lpe, P\rangle]\!]^S_W &= [\![P]\!]^{\mathfrak{D}}_G \quad \text{where } \mathfrak{D} = \text{dataset}_W\left(\textstyle\bigcup_{u\in S}[\![lpe]\!]^u_W\right) \text{ with default graph } G,\\
[\![(\mathsf{SEED}\ \ U\ q)]\!]^S_W &= [\![q]\!]^U_W,\\
[\![(\mathsf{SEED}\ \ ?v\ q)]\!]^S_W &= \textstyle\bigcup_{u\in\mathcal{U}}\big([\![q]\!]^{\{u\}}_W \bowtie \{\mu_u\}\big) \quad \text{where } \mu_u = \{?v \mapsto u\} \text{ for all } u \in \mathcal{U},\\
[\![(q_1\ \mathsf{UNION}\ q_2)]\!]^S_W &= [\![q_1]\!]^S_W \cup [\![q_2]\!]^S_W,\\
[\![(q_1\ \mathsf{AND}\ q_2)]\!]^S_W &= [\![q_1]\!]^S_W \bowtie [\![q_2]\!]^S_W,\\
[\![\,\pi_V q\,]\!]^S_W &= \{\mu \mid \text{there exists } \mu' \in [\![q]\!]^S_W \text{ such that } \mu \text{ and } \mu' \text{ are}\\
&\qquad\qquad \text{compatible and } \mathrm{dom}(\mu) = \mathrm{dom}(\mu') \cap V\}.
\end{aligned}$$

Now for the semantics of LPEs, given a context URI $u_{\mathsf{ctx}} \in \mathrm{dom}(adoc)$, the u_{ctx}**-based evaluation** of LPEs over W, denoted by $[\![\cdot]\!]^{u_{\mathsf{ctx}}}_W$, is defined recursively as follows:

$$\begin{aligned}
[\![\,\varepsilon\,]\!]^{u_{\mathsf{ctx}}}_W &= \{u_{\mathsf{ctx}}\},\\
[\![lp]\!]^{u_{\mathsf{ctx}}}_W &= \{u \in \mathcal{U} \mid \text{there exist a link graph edge } \langle d_{\mathsf{src}}, (t,u), d_{\mathsf{tgt}}\rangle \in \mathcal{G}_W, \text{ with}\\
&\qquad\qquad d_{\mathsf{src}} = adoc(u_{\mathsf{ctx}}), \text{ that matches } lp \text{ in the context of } u_{\mathsf{ctx}}\},\\
[\![lpe_1/lpe_2]\!]^{u_{\mathsf{ctx}}}_W &= \{u \in [\![lpe_2]\!]^{u'}_W \mid u' \in [\![lpe_1]\!]^{u_{\mathsf{ctx}}}_W\},\\
[\![lpe_1|lpe_2]\!]^{u_{\mathsf{ctx}}}_W &= [\![lpe_1]\!]^{u_{\mathsf{ctx}}}_W \cup [\![lpe_2]\!]^{u_{\mathsf{ctx}}}_W,\\
[\![lpe^*]\!]^{u_{\mathsf{ctx}}}_W &= \{u_{\mathsf{ctx}}\} \cup [\![lpe]\!]^{u_{\mathsf{ctx}}}_W \cup [\![lpe/lpe]\!]^{u_{\mathsf{ctx}}}_W \cup [\![lpe/lpe/lpe]\!]^{u_{\mathsf{ctx}}}_W \cup \dots,\\
[\![\,[lpe]\,]\!]^{u_{\mathsf{ctx}}}_W &= \{u_{\mathsf{ctx}} \mid [\![lpe]\!]^{u_{\mathsf{ctx}}}_W \neq \emptyset\},\\
[\![\,\langle ?v, q\rangle\,]\!]^{u_{\mathsf{ctx}}}_W &= \{u \in \mathcal{U} \mid \text{there exists } \mu \in [\![q]\!]^{\{u_{\mathsf{ctx}}\}}_W \text{ such that } \mu(?v) = u\}.
\end{aligned}$$

Moreover, if $u_{\mathsf{ctx}} \notin \mathrm{dom}(adoc)$, then $[\![lpe]\!]^{u_{\mathsf{ctx}}}_W = \emptyset$ for every LPE.

Example 3. Let lpe_{ex} be the LPE $\langle\_, p_1, \_\rangle^*/[\langle\_, p_2, \_\rangle]$. This LPE selects documents that can be reached via arbitrarily long paths of data links with predicate p_1 and, additionally, have some outgoing data link with predicate p_2. For our example Web W_{ex} and context URI u_{A}, the LPE selects documents $d_{\mathsf{A}} = adoc_{\mathsf{ex}}(u_{\mathsf{A}})$ and $d_{\mathsf{C}} = adoc_{\mathsf{ex}}(u_{\mathsf{C}})$. More precisely, we have $[\![lpe_{\mathsf{ex}}]\!]^{u_{\mathsf{A}}}_{W_{\mathsf{ex}}} = \{u_{\mathsf{A}}, u_{\mathsf{C}}\}$. Note that document d_{B} can also be reached via a p_1–path, but it does not pass the p_2–related test.

Example 4. Consider a set of URIs $S_{\mathsf{ex}} = \{u_{\mathsf{A}}\}$ and a basic LDQL query $\langle lpe_{\mathsf{ex}}, B_{\mathsf{ex}}\rangle$ whose LPE is lpe_{ex} as introduced in Example 3 and whose SPARQL graph pattern is a basic graph pattern that contains two triple patterns, $B_{\mathsf{ex}} = \{\langle ?x, p_1, ?y\rangle, \langle ?x, p_2, ?z\rangle\}$. Given that we have $[\![lpe_{\mathsf{ex}}]\!]^{u_{\mathsf{A}}}_{W_{\mathsf{ex}}} = \{u_{\mathsf{A}}, u_{\mathsf{C}}\}$ (cf. Example 3), $\text{dataset}_{W_{\mathsf{ex}}}([\![lpe_{\mathsf{ex}}]\!]^{u_{\mathsf{A}}}_{W_{\mathsf{ex}}})$ has the default graph $\{\langle u_{\mathsf{A}}, p_1, u_{\mathsf{B}}\rangle, \langle u_{\mathsf{B}}, p_2, u_{\mathsf{C}}\rangle, \langle u_{\mathsf{A}}, p_2, u_{\mathsf{C}}\rangle\}$ (cf. Example 2). Then, according to the query semantics, the result of query $\langle lpe_{\mathsf{ex}}, B_{\mathsf{ex}}\rangle$ over W_{ex} using seeds S_{ex} consists of a single solution mapping, namely $\mu = \{?x \mapsto u_{\mathsf{A}}, ?y \mapsto u_{\mathsf{B}}, ?z \mapsto u_{\mathsf{C}}\}$.

Example 5. Consider an LDQL query $q_{\mathsf{ex}} = \big(\mathsf{SEED}\ \ ?x\ \langle\varepsilon, \langle ?x, p_1, ?w\rangle\rangle\big)$ whose subquery is a basic LDQL query that has a single triple pattern as its SPARQL graph pattern. Additionally, let $q'_{\mathsf{ex}} = \big\langle lpe_{\mathsf{ex}}, \{\langle ?x, p_1, ?y\rangle, \langle ?x, p_2, ?z\rangle\}\big\rangle$ be the basic LDQL query introduced in Example 4, and let q''_{ex} be the conjunction of these two queries; i.e., $q''_{\mathsf{ex}} = (q_{\mathsf{ex}}\ \mathsf{AND}\ q'_{\mathsf{ex}})$. By Example 4 we know that

$[\![q'_{\mathsf{ex}}]\!]^{S_{\mathsf{ex}}}_{W_{\mathsf{ex}}} = \{\mu\}$ with $\mu = \{?x \mapsto u_{\mathsf{A}}, ?y \mapsto u_{\mathsf{B}}, ?z \mapsto u_{\mathsf{C}}\}$. Furthermore, based on the data given in Example 1, it is easy to see that $[\![q_{\mathsf{ex}}]\!]^{S_{\mathsf{ex}}}_{W_{\mathsf{ex}}} = \{\mu_1, \mu_2\}$ with $\mu_1 = \{?x \mapsto u_{\mathsf{A}}, ?w \mapsto u_{\mathsf{B}}\}$ and $\mu_2 = \{?x \mapsto u_{\mathsf{B}}, ?w \mapsto u_{\mathsf{C}}\}$. For the S_{ex}-based evaluation of q''_{ex} over W_{ex}, the result sets $[\![q_{\mathsf{ex}}]\!]^{S_{\mathsf{ex}}}_{W_{\mathsf{ex}}}$ and $[\![q'_{\mathsf{ex}}]\!]^{S_{\mathsf{ex}}}_{W_{\mathsf{ex}}}$ have to be joined. Thus, we need to compute $\{\mu_1, \mu_2\} \bowtie \{\mu\}$, which results in a single mapping $\mu' = \mu_1 \cup \mu = \{?x \mapsto u_{\mathsf{A}}, ?w \mapsto u_{\mathsf{C}}, ?y \mapsto u_{\mathsf{B}}, ?z \mapsto u_{\mathsf{C}}\}$.

3.3 Algebraic Properties of LDQL Queries

As a basis for the discussion in the next sections, we show some simple algebraic properties. We say that LDQL queries q and q' are semantically equivalent, denoted by $q \equiv q'$, if $[\![q]\!]^S_W = [\![q']\!]^S_W$ holds for every Web of Linked Data W and every finite set $S \subseteq \mathcal{U}$.

Lemma 1. *The operators* AND *and* UNION *are associative and commutative.*

Lemma 2. *Let q_1, q_2, q_3 be LDQL queries, the following semantic equivalences hold:*

$$(q_1 \text{ AND } (q_2 \text{ UNION } q_3)) \equiv ((q_1 \text{ AND } q_2) \text{ UNION } (q_1 \text{ AND } q_3)) \tag{1}$$

$$\pi_V(q_1 \text{ UNION } q_2) \equiv (\pi_V q_1 \text{ UNION } \pi_V q_2) \tag{2}$$

$$(\text{SEED } U \ (q_1 \text{ UNION } q_2)) \equiv ((\text{SEED } U \ q_1) \text{ UNION } (\text{SEED } U \ q_2)) \tag{3}$$

$$(\text{SEED } ?v \ (q_1 \text{ UNION } q_2)) \equiv ((\text{SEED } ?v \ q_1) \text{ UNION } (\text{SEED } ?v \ q_2)) \tag{4}$$

Lemma 1 allows us to write sequences of either AND or UNION without parentheses. Our next result shows the power of the construction $\langle ?v, q\rangle$. In particular, it shows the somehow surprising finding that link patterns *lp*, concatenation /, disjunction |, and the test [·], are just *syntactic sugar* as they can be simulated by using ε, $\langle ?v, q\rangle$ and $(\cdot)^*$.

Proposition 1. *For every LDQL query q, there exists an LDQL query q' s.t. $q \equiv q'$ and every LPE in q' consists only of the symbol ε, the construction $\langle ?v, q\rangle$, and operator $(\cdot)^*$.*

Proof (Sketch). The proof is based on a recursive translation of link path expressions beginning with link patterns. For instance, a link pattern of the form $\langle +, p, \_\rangle$ is encoded by $\langle ?v, \langle \varepsilon, (\text{GRAPH } ?u \ (?u, p, ?v))\rangle\rangle$, and we can similarly encode all types of link patterns. To encode / we make use of $\langle ?v, q\rangle$ and the operator AND inside q as follows. Consider an LPE $r = r_1/r_2$. It can be shown that r is equivalent to $\langle ?v, q\rangle$ where q is:

$$\big(\ \langle r_1, (\text{GRAPH } ?x \ \{ \ \})\rangle \ \text{ AND } \ \big(\text{SEED } ?x \ \langle r_2, (\text{GRAPH } ?v \ \{ \ \})\rangle\big) \ \big).$$

Similarly, to encode | we make use of UNION and to encode [·] we use projection.

Although not strictly necessary, we decided to keep link patterns and operators /, |, and [·] since they represent a natural and intuitive way of expressing navigation paths.

4 Comparison with Previous Linked Data Query Formalisms

In this section, we compare LDQL with alternative formalisms to query Linked Data on the WWW. There are some general query languages for the WWW (proposed before the advent of Linked Data) that are related to our proposal; in particular, WebSQL [15], which is similar in spirit to LDQL but different in the features that the languages posses. Two main novelties of LDQL compared with WebSQL are the possibility to dynamically select seed URIs at query time, and the traversal of links according to properties of the queried documents that can be defined in the same LDQL query. Neither of these are expressible in WebSQL. While a complete formal comparison between LDQL and WebSQL is certainly very interesting, we leave it for future work and, instead, focus on three more recent proposals of query formalisms for the Web of Linked Data [7,11,14]. We formally show that LDQL is strictly more expressive than every one of them.

4.1 Comparison with Property Paths Under Context-Based Query Semantics

Property paths (PPs for short) were introduced in SPARQL 1.1 as a way of adding navigational power to the language [8]. PPs are a form of regular expressions that are evaluated over a single (local) RDF graph; a PP expression is used to retrieve pairs $\langle a, b\rangle$ of nodes in the graph such that there is a path from a to b whose sequence of edge labels belongs (as a string) to the regular language defined by the expression. The syntax of PP expressions is given by the following grammar[1], where $p, u_1, u_2, \ldots, u_k$ are URIs.

$$pe := p \mid\ !(u_1|u_2|\cdots|u_k) \mid pe/pe \mid pe|pe \mid pe^*$$

A PP-pattern is defined as a tuple of the form $\langle \alpha, pe, \beta\rangle$ where pe is a PP expression, and α and β are in $\mathcal{U} \cup \mathcal{L} \cup \mathcal{V}$.

In [14] the authors adapted the semantics of PP-patterns so that they can be used to query the Web of Linked Data. The proposed query semantics is called *context-based semantics* [14]. To define this semantics, the authors first introduce the notion of a *context selector* for a Web of Linked Data W. This context selector is a function $C^W(\cdot)$ that given a URI $u \in \mathrm{dom}(adoc)$ returns the RDF triples in $\mathrm{data}(adoc(u))$ that have u in the subject position. Formally, for every URI $u \in \mathrm{dom}(adoc)$ we have $C^W(u) = \{\langle s, p, o\rangle \in \mathrm{data}(adoc(u)) \mid s = u\}$. To simplify the exposition, the authors extended the definition of $C^W(\cdot)$ to also handle URIs not in $\mathrm{dom}(adoc)$, and literals and blank nodes. For any such RDF term a they define $C^W(a)$ as the empty set.

[1] In [14] the reverse path construction $\hat{}pe$ is also considered. We do not consider it here as the form of navigation of these reverse paths does not represent a traversal of the link graph.

The context-based semantics for PPs over the Web of Linked Data in [14] is a bag semantics that follows closely the semantics for PPs defined in the normative semantics of SPARQL 1.1 [8]. Hence, both semantics use a procedure, the *ArbitraryLengthPath* procedure [8], to define the semantics of the $(\cdot)^*$ operator. It was shown in [1] that for sets semantics, the normative semantics of PPs can be defined by using standard techniques for regular expressions. To make the comparison with LDQL, in this paper we adapt the context-based semantics for PPs presented in [14] by following the techniques in [1], and consider only sets of mappings. To this end, we define a function $[\![\cdot]\!]^{\mathsf{ctxt}}_W$, that given a PP-pattern, returns its evaluation under context-based semantics over the Web of Linked Data W. In the definition, for a solution mapping μ and an RDF term α, we use the notation $\mu[\alpha]$ with the following meaning: $\mu[\alpha] = \mu(\alpha)$ if $\alpha \in \mathrm{dom}(\mu)$, and $\mu[\alpha] = \alpha$ in the other case. Similarly, $\mu[\langle s, p, o\rangle] = \langle \mu[s], \mu[p], \mu[o]\rangle$.

$$\begin{aligned}
[\![(\alpha, p, \beta)]\!]^{\mathsf{ctxt}}_W &= \{\mu \mid \mathrm{dom}(\mu) = \{\alpha, \beta\} \cap \mathcal{V} \text{ and } \mu[\langle \alpha, p, \beta\rangle] \in C^W(\mu[\alpha])\}\\
[\![(\alpha, !(u_1|\cdots|u_k), \beta)]\!]^{\mathsf{ctxt}}_W &= \{\mu \mid \mathrm{dom}(\mu) = \{\alpha, \beta\} \cap \mathcal{V} \text{ and exists } p \text{ s.t.}\\
&\qquad\qquad \mu[\langle \alpha, p, \beta\rangle] \in C^W(\mu[\alpha]) \text{ and } p \notin \{u_1, \ldots, u_k\}\}\\
[\![(\alpha, pe_1/pe_2, \beta)]\!]^{\mathsf{ctxt}}_W &= \pi_{\{\alpha,\beta\}\cap\mathcal{V}}\big([\![(\alpha, pe_1, ?v)]\!]^{\mathsf{ctxt}}_W \bowtie [\![(?v, pe_2, \beta)]\!]^{\mathsf{ctxt}}_W\big)\\
[\![(\alpha, pe_1|pe_2, \beta)]\!]^{\mathsf{ctxt}}_W &= [\![(\alpha, pe_1, \beta)]\!]^{\mathsf{ctxt}}_W \cup [\![(\alpha, pe_2, \beta)]\!]^{\mathsf{ctxt}}_W\\
[\![(\alpha, pe^*, \beta)]\!]^{\mathsf{ctxt}}_W &= \{\mu \mid \mathrm{dom}(\mu) = \{\alpha, \beta\} \cap \mathcal{V}, \mu[\alpha] = \mu[\beta] \text{ and } \mu[\alpha] \in \mathit{terms}(W)\} \cup\\
&\qquad [\![(\alpha, pe, \beta)]\!]^{\mathsf{ctxt}}_W \cup [\![(\alpha, pe/pe, \beta)]\!]^{\mathsf{ctxt}}_W \cup [\![(\alpha, pe/pe/pe, \beta)]\!]^{\mathsf{ctxt}}_W \cup \cdots
\end{aligned}$$

A *PP-based SPARQL query* [14] is an expression formed by combining PP-patterns using the standard SPARQL operators AND, UNION, OPT, FILTER and so on, following the standard semantics for these operators [2]. Our next results show that LDQL is strictly more expressive than PP-based SPARQL queries under context-based semantics.

Theorem 1. *There exists an LDQL query that cannot be expressed as a PP-based SPARQL query under context-based semantics.*

Proof (Sketch). One can show that LDQL query $q = \big(\mathsf{SEED}\ U\ \langle\langle +, p, \_\rangle, (?x, ?x, ?x)\rangle\big)$ with $U = \{u\}$ cannot be expressed by PPs under context-based semantics because this semantics is "blind" to triples that are not authoritative. For instance, in a Web $W = \langle \{d, d'\}, \mathit{adoc}\rangle$ with $\mathrm{data}(d) = \{\langle u, p, u'\rangle\}$, $\mathrm{data}(d') = \{\langle u', p, u\rangle, \langle u, u, u\rangle\}$, $\mathit{adoc}(u) = d$ and $\mathit{adoc}(u') = d'$, the evaluation of q is the solution mapping $\{?x \mapsto u\}$. Notice that the only authoritative triple in d' is $\langle u', p, u\rangle$ as $d' = \mathit{adoc}(u') \neq \mathit{adoc}(u)$. Hence, one can prove that PP-based SPARQL queries under context-based semantics cannot access triple $\langle u, u, u\rangle$ in d', and thus, will never have $\{?x \mapsto u\}$ as solution.

Theorem 2. *Let $\alpha, \beta \in \mathcal{U} \cup \mathcal{L} \cup \mathcal{V}$. Then, for every PP-pattern $\langle \alpha, pe, \beta\rangle$, there exists an LDQL query q such that $[\![\langle \alpha, pe, \beta\rangle]\!]^{\mathsf{ctxt}}_W = [\![q]\!]^{\emptyset}_W$ for every Web of Linked Data W.*

Proof (Sketch). In the proof we provide a translation scheme from PPs to LDQL. One major complication is that PPs can retrieve literals and, in general, values that are not in $\mathrm{dom}(adoc)$, which are difficult to handle by LPEs. For every PP-pattern $\langle ?x, pe, ?y\rangle$ we construct an LDQL query $Q_{pe}(?x, ?y)$. For example, for $\langle ?x, pe_1/pe_2, ?y\rangle$, our query is $\pi_{\{?x,?y\}}\big(Q_{pe_1}(?x, ?z) \text{ AND } Q_{pe_2}(?z, ?y)\big)$, and for $\langle ?x, !(u_1|\cdots|u_k), ?y\rangle$ the translation is $\big(\text{SEED } ?x \ \big\langle \varepsilon, \big((?x, ?p, ?y) \text{ FILTER } (?p \neq u_1 \wedge \cdots \wedge ?p \neq u_k)\big)\big\rangle\big)$. To handle pe^* we need to use the construction $\langle ?v, q\rangle$ of LPEs, plus $(\cdot)^*$.

4.2 Comparison with NautiLOD

NautiLOD is a navigation language to traverse Linked Data on the WWW and to perform actions (such as sending emails) during the traversal [7]. We compare LDQL with NautiLOD without action rules. The syntax of NautiLOD expressions (without actions) is given by the following grammar (where $p \in \mathcal{U}$ and P is a SPARQL graph pattern).

$$ne := p \mid p\hat{\ } \mid \langle \_ \rangle \mid ne/ne \mid ne|ne \mid ne^* \mid ne[(\text{ASK } P)]$$

In terms of our data model[2], the semantics of NautiLOD expressions over a Web of Linked Data $W = \langle D, adoc\rangle$ from URI $u \in \mathrm{dom}(adoc)$ is defined recursively as follows.

$$\begin{aligned}
[\![p]\!]^u_W &= \{u' \mid \langle u, p, u'\rangle \in \mathrm{data}(adoc(u))\}\\
[\![p\hat{\ }]\!]^u_W &= \{u' \mid \langle u', p, u\rangle \in \mathrm{data}(adoc(u))\}\\
[\![\langle \_ \rangle]\!]^u_W &= \{u' \mid \langle u, p, u'\rangle \in \mathrm{data}(adoc(u)) \text{ for some } p \in \mathcal{U}\}\\
[\![ne_1/ne_2]\!]^u_W &= \{u'' \mid u'' \in [\![ne_2]\!]^{u'}_W \text{ for some } u' \in [\![ne_1]\!]^u_W \text{ with } u' \in \mathrm{dom}(adoc)\}\\
[\![ne_1 | ne_2]\!]^u_W &= [\![ne_1]\!]^u_W \cup [\![ne_2]\!]^u_W\\
[\![ne^*]\!]^u_W &= \{u\} \cup [\![ne]\!]^u_W \cup [\![ne/ne]\!]^u_W \cup [\![ne/ne/ne]\!]^u_W \cup \cdots\\
[\![ne[(\text{ASK } P)]]\!]^u_W &= \{u' \mid u' \in [\![ne]\!]^u_W,\ u' \in \mathrm{dom}(adoc) \text{ and } [\![P]\!]_{\mathrm{data}(adoc(u'))} \neq \emptyset\}
\end{aligned}$$

We next show that for every NautiLOD expression there exists an equivalent LDQL query. Notice that the evaluation of a NautiLOD expression is a set of URIs, whereas the evaluation of an LDQL query is a set of mappings. Thus, to formally state our result we compare NautiLOD with LDQL queries that have a single *free variable*. Let $q(?x)$ be an LDQL query with $?x$ as free variable. We say that $q(?x)$ and a NautiLOD expression ne are equivalent if for every Web of Linked Data $W = \langle D, adoc\rangle$ and URIs u, u' with $u \in \mathrm{dom}(adoc)$ it holds that $u' \in [\![ne]\!]^u_W$ if and only if $\{?x \mapsto u'\} \in [\![q(?x)]\!]^{\{u\}}_W$.

Theorem 3. *For every NautiLOD expression ne, there exists an LDQL query $q(?x)$, with $?x$ a free variable, that is equivalent to ne.*

[2] In [7], all URIs have an assigned set of RDF triples (which may be empty). In our data model one can have URIs not in $\mathrm{dom}(adoc)$. Hence, to properly capture the semantics of NautiLOD in terms of our data model we have to introduce conditions of the form "$u' \in \mathrm{dom}(adoc)$."

Proof (Sketch). The proof begins with a simple translation that replaces every $p \in \mathcal{U}$ in a NautiLOD expression by a link pattern $\langle +, p, \_ \rangle$. For instance, the expression p_1/p_2^* is translated into $\langle +, p_1, \_ \rangle / \langle +, p_2, \_ \rangle^*$. To translate $\langle \_ \rangle$ and $[(\textsf{ASK}\ P)]$ we use $\langle ?v, q \rangle$. The complete translation poses several other complications (as described in the extended version [13]). In particular, the last step of NautiLOD expressions must be translated by using a SPARQL pattern and not an LPE. For this we use the following property. Given a regular expression r that does not generate the empty word, one can always write r as $r_1/a_1 | \cdots | r_k/a_k$ where the a_i's are base symbols of the alphabet. Thus, we can translate r by using LPEs to translate the r_i's as outlined above; next, translate the a_i's by using a method similar to the proof of Theorem 2, and finally use UNION for $|$.

Along the same lines of Theorem 1 one can prove the following result.

Theorem 4. *There exists an LDQL query $q(?x)$ that cannot be expressed in NautiLOD.*

4.3 Comparison with SPARQL Under Reachability-Based Query Semantics

In [11] the author introduces a family of reachability-based query semantics based on which SPARQL graph patterns can be used as a query language for Linked Data on the WWW. Similar to how the scope of the SPARQL part of a basic LDQL query is restricted to particular documents, reachability-based semantics restrict the scope of SPARQL queries to documents that can be reached by traversing a well-defined set of data links. To specify what data links belong to such a set, the notion of a *reachability criterion* is used; that is, a function $c\colon \mathcal{T} \times \mathcal{U} \times \mathcal{P} \to \{\text{true}, \text{false}\}$ where $\mathcal{P}$ denotes the set of all SPARQL graph patterns. Then, given such a reachability criterion c, a finite set S of URIs and a SPARQL graph pattern P, a document $d \in \mathcal{D}$ is *(c, S, P)-reachable* in a Web of Linked Data $W = \langle D, adoc \rangle$ if any of the following two conditions holds:

1. There exists a URI $u \in S$ such that $adoc(u) = d$; or
2. there exists a link graph edge $\langle d_{\mathsf{src}}, (t, u), d_{\mathsf{tgt}} \rangle \in \mathcal{G}_W$ such that (i) d_{src} is (c, S, P)-reachable in W, (ii) $c(t, u, P) = \text{true}$, and (iii) $d_{\mathsf{tgt}} = d$.

Notice how the second condition restricts the notion of reachability by ignoring data links that do not satisfy the given reachability criterion c. Concrete examples of reachability criteria are c_{All}, c_{None}, and c_{Match} [11], where c_{All} selects all data links, and c_{None} ignores all data links; i.e., $c_{\mathsf{All}}(t, u, P) = \text{true}$ and $c_{\mathsf{None}}(t, u, P) = \text{false}$ for all tuples $\langle t, u, P \rangle \in \mathcal{T} \times \mathcal{U} \times \mathcal{P}$. In contrast to such an all-or-nothing strategy, criterion c_{Match} returns true for every data link whose triple matches a triple pattern of the given graph pattern; formally, $c_{\mathsf{Match}}(t, u, P) = \text{true}$ if and only if there exists some solution mapping μ such that $\mu[tp] = t$ for an arbitrary triple pattern tp that is contained in P.

Given the notion of a reachability criterion, it is possible to define a family of (reachability-based) query semantics for SPARQL. To this end, let c be a reachability criterion, let S be a finite set of URIs, and let P be a SPARQL graph pattern. Then, for any Web of Linked Data $W = \langle D, adoc \rangle$, the *$S$-based evaluation* of P over W *under c-semantics*, denoted by $[\![P]\!]_W^{\mathsf{R}(c,S)}$, is the set of solution mappings $[\![P]\!]_G$ where G is the RDF graph that consists of all triples from all documents that are (c, S, P)-reachable in W.

While there exist an infinite number of possible reachability criteria, in this paper we focus on c_{All}, c_{None}, and c_{Match}. The following two results show that LDQL is strictly more expressive than SPARQL graph patterns under any of these three query semantics.

Theorem 5. *Let $c \in \{c_{\mathsf{All}}, c_{\mathsf{None}}, c_{\mathsf{Match}}\}$. For every SPARQL graph pattern P there exists an LDQL query q such that $[\![P]\!]_W^{\mathsf{R}(c,S)} = [\![q]\!]_W^S$ for every Web W and $S \subseteq \mathcal{U}$.*

Proof (Sketch). We only sketch the case of c_{All}-semantics. In this case, one can prove that the LPE $lpe^{c_{\mathsf{All}}} = \langle \_, \_, \_ \rangle^*$ simulates the reachability criterion c_{All}, and, thus, $[\![P]\!]_W^{\mathsf{R}(c_{\mathsf{All}},S)} = [\![\langle lpe^{c_{\mathsf{All}}}, P \rangle]\!]_W^S$. One can also find LPEs to simulate c_{None} and c_{Match}.

Theorem 6. *Let $c \in \{c_{\mathsf{All}}, c_{\mathsf{None}}, c_{\mathsf{Match}}\}$. There exists an LDQL query q for which there does not exist a SPARQL pattern P such that $[\![P]\!]_W^{\mathsf{R}(c,S)} = [\![q]\!]_W^S$ for every W and $S \subseteq \mathcal{U}$.*

5 Web-Safeness of LDQL Queries

In this section we study the "Web-safeness" of LDQL queries, where, informally, we call a query *Web-safe* if a complete execution of the query over the WWW is possible in practice (which is not the case for all LDQL queries as we shall see). To provide a more formal definition of this notion of Web-safeness we make the following observations. While the mathematical structures introduced by our data model capture the notion of Linked Data on the WWW formally (and, thus, allow us to provide a formal semantics for LDQL queries), in practice, these structures are not available completely for the WWW. For instance, given that an infinite number of strings can be used as HTTP URIs [6], we cannot assume complete information about which URIs are in the domain of the partial function *adoc* (i.e., can be looked up to retrieve some document) and which are not; in fact, disclosing this information would require a process that systematically tries to look up every possible HTTP URI and, thus, would never terminate. Therefore, it is also impossible to guarantee the discovery of every document in the set D (without looking up an infinite number of URIs). Consequently, any query whose execution requires a complete enumeration of this set is not feasible in practice. Based on these observations, we define *Web-safeness* of LDQL queries as follows.

Definition 6. An LDQL query q is **Web-safe** if there exists an algorithm that, for any finite Web of Linked Data $W = \langle D, adoc \rangle$ and any finite set S of URIs, computes $[\![q]\!]_W^S$ by looking up only a finite number of URIs without assuming an a priori availability of any information about the sets D and $\mathrm{dom}(adoc)$.

Example 6. Recall our example queries q_{ex}, q'_{ex}, and q''_{ex} (cf. Example 5). For query $q_{\mathsf{ex}} = \big(\mathsf{SEED}\ ?x\ \langle \varepsilon, \langle ?x, p_1, ?z\rangle\rangle\big)$, any URI $u \in \mathcal{U}$ may be used to obtain a nonempty subset of the query result as long as a lookup of u retrieves a document whose data includes RDF triples that match $\langle u, p_1, ?z\rangle$. Therefore, without access to D or $\mathrm{dom}(adoc)$ of the queried Web $W = \langle D, adoc\rangle$, the completeness of the computed query result can be guaranteed only by checking each of the infinitely many possible HTTP URIs. Hence, query q_{ex} is *not* Web-safe. In contrast, although it contains q_{ex} as a subquery, query $q''_{\mathsf{ex}} = (q_{\mathsf{ex}}\ \mathsf{AND}\ q'_{\mathsf{ex}})$ is Web-safe, and so is $q'_{\mathsf{ex}} = \langle lpe_{\mathsf{ex}}, B_{\mathsf{ex}}\rangle$. Given u_{A} as seed URI, a possible execution algorithm for q'_{ex} may first compute $[\![lpe_{\mathsf{ex}}]\!]_W^{u_{\mathsf{A}}}$ by traversing the queried Web W based on lpe_{ex}. Thereafter, the algorithm retrieves documents by looking up all URIs $u \in [\![lpe_{\mathsf{ex}}]\!]_W^{u_{\mathsf{A}}}$ (or simply keeps these documents after the traversal); and, finally, the algorithm evaluates pattern B_{ex} over the union of the RDF data in the retrieved documents. If W is finite (i.e., contains a finite number of documents), the traversal process requires a finite number of URI lookups only, and so does the retrieval of documents in the second step; the final step does not look up any URI. To see that q''_{ex} is also Web-safe we note that after executing subquery q'_{ex} (e.g., by using the algorithm as outlined before), the execution of the other (non-Web-safe) subquery q_{ex} can be reduced to a finite number of URI lookups, namely the URIs bound to variable $?x$ in solution mappings obtained for subquery q'_{ex}. Although any other URI may also be used to obtain solution mappings for q_{ex}, such solution mappings cannot be joined with any of the solution mappings for q'_{ex} and, thus, are irrelevant for the result of q''_{ex}.

The example illustrates that there exists an LDQL query that is not Web-safe. In fact, it is not difficult to see that the argument for the non-Web-safeness of query q_{ex} as made in the example can be applied to any LDQL query of the form $(\mathsf{SEED}\ ?x\ q)$ where subquery q is a (satisfiable) basic LDQL query; that is, none of these queries is Web-safe. However, the example also shows that more complex queries that contain such non-Web-safe subqueries may still be Web-safe. Therefore, we now show properties to identify LDQL queries that are Web-safe even if some of their subqueries are not. We begin with queries of the forms $\langle lpe, P\rangle$, $\pi_V q$, $(\mathsf{SEED}\ \ U\ q)$, and $(q_1\ \mathsf{UNION}\ \ldots\ \mathsf{UNION}\ q_n)$.

Proposition 2. *An LDQL query q is Web-safe if any of the following properties holds:*

1. *Query q is of the form $\langle lpe, P\rangle$ and lpe is Web-safe, where we call an LPE Web-safe if either (i) it is of the form $\langle ?v, q'\rangle$ and LDQL query q' is Web-safe, or (ii) it is of any form other than $\langle ?v, q'\rangle$ and all its subexpressions (if any) are Web-safe LPEs;*
2. *Query q is of the form $\pi_V q'$ or $(\mathsf{SEED}\ \ U\ q')$, and subquery q' is Web-safe; or*
3. *Query q is of the form $(q_1\ \mathsf{UNION}\ \ldots\ \mathsf{UNION}\ q_n)$ and each q_i $(1 \leq i \leq n)$ is Web-safe.*

It remains to discuss LDQL queries of the form $(q_1 \text{ AND } \dots \text{ AND } q_m)$. Our discussion of query q''_{ex} in Example 6 suggests that such queries can be shown to be Web-safe if all non-Web-safe subqueries are of the form $(\text{SEED } ?v\ q)$ and it is possible to execute these subqueries by using variable bindings obtained from other subqueries. A necessary condition for this execution strategy is that the variable in question (i.e., $?v$) is guaranteed to be bound in every possible solution mapping obtained from the other subqueries.

To allow for an automated verification of this condition we adopt Buil-Aranda et al.'s notion of strongly bound variables [4]. To this end, for any SPARQL graph pattern P, let $\text{sbvars}(P)$ denote the set of strongly bound variables in P as defined by Buil-Aranda et al. [4]. For the sake of space, we do not repeat the definition here. However, we emphasize that $\text{sbvars}(P)$ can be constructed recursively, and each variable in $\text{sbvars}(P)$ is guaranteed to be bound in every possible solution for P [4, Proposition 1]. To carry over these properties to LDQL queries, we use the notion of strongly bound variables in SPARQL patterns to define the following notion of strongly bound variables in LDQL queries; thereafter, in Lemma 3, we show the desired boundedness guarantee.

Definition 7. The set of **strongly bound variables** in an LDQL query q, denoted by $\text{sbvars}(q)$, is defined recursively as follows:

1. If q is of the form $\langle lpe, P\rangle$, then $\text{sbvars}(q) = \text{sbvars}(P)$.
2. If q is of the form $(q_1 \text{ AND } q_2)$, then $\text{sbvars}(q) = \text{sbvars}(q_1) \cup \text{sbvars}(q_2)$.
3. If q is of the form $(q_1 \text{ UNION } q_2)$, then $\text{sbvars}(q) = \text{sbvars}(q_1) \cap \text{sbvars}(q_2)$.
4. If q is of the form $\pi_V q'$, then $\text{sbvars}(q) = \text{sbvars}(q') \cap V$.
5. If q is of the form $(\text{SEED } U\ q')$, then $\text{sbvars}(q) = \text{sbvars}(q')$.
6. If q is of the form $(\text{SEED } ?v\ q')$, then $\text{sbvars}(q) = \text{sbvars}(q') \cup \{?v\}$.

Lemma 3. *Let q be an LDQL query. For every finite set S of URIs, every Web of Linked Data W, and every solution mapping $\mu \in [\![q]\!]^S_W$, it holds that* $\text{sbvars}(q) \subseteq \text{dom}(\mu)$.

We are now ready to show the following result.

Theorem 7. *An LDQL query of the form $(q_1 \text{ AND } q_2 \text{ AND } \dots \text{ AND } q_m)$ is Web-safe if there exists a total order $\prec$ over the set of subqueries $\{q_1, q_2, \dots, q_m\}$ such that for each subquery q_i ($1 \le i \le m$), it holds that either (i) q_i is Web-safe or (ii) q_i is of the form $(\text{SEED } ?v\ q)$ where q is Web-safe and $?v \in \bigcup_{q_j \prec q_i} \text{sbvars}(q_j)$.*

Proof (Sketch). We prove Theorem 7 based on an iterative algorithm that generalizes the execution of query q''_{ex} as outlined in Example 6. That is, the algorithm executes the subqueries $q_1 \dots q_m$ sequentially in the order $\prec$ such that each iteration executes one of the subqueries by using the solution mappings computed during the previous iteration.

With the results in this section we have all ingredients to devise a procedure to show Web-safeness for a large number of queries (including queries that are arbitrarily nested). However, as a potential limitation of such a procedure we note that Theorem 7 can be applied only in cases in which all non-Web-safe subqueries are of the form $(\textsf{SEED}\ ?v\ q)$. For instance, the theorem cannot be applied to show that an LDQL query of the form $\big(q_1\ \textsf{AND}\ (q_2\ \textsf{UNION}\ (\textsf{SEED}\ ?x\ q_3))\big)$ is Web-safe if $?x \in \text{sbvars}(q_1)$ and q_1, q_2 and q_3 are Web-safe. On the other hand, for the semantically equivalent query $\big((q_1\ \textsf{AND}\ q_2)\ \textsf{UNION}\ (q_1\ \textsf{AND}\ (\textsf{SEED}\ ?x\ q_3))\big)$ we can show Web-safeness based on Theorem 7 (and Proposition 2). Fortunately, we may leverage the following fact to improve the effectiveness of applying Theorem 7 in the procedure that we aim to devise.

Fact 1. If an LDQL query q is Web-safe, then so is any LDQL query q' with $q' \equiv q$.

As a consequence of Fact 1, we may use the equivalences in Lemma 2 to rewrite a given query into an equivalent query that is more suitable for testing Web-safeness based on our results. To this end, we introduce specific normal forms for LDQL queries:

Definition 8. An LDQL query is in **union-free normal form** if it is of the form $(q_1\ \textsf{AND}\ \dots\ \textsf{AND}\ q_m)$ with $m \geq 1$ and each q_i $(1 \leq i \leq m)$ is either (i) a basic LDQL query or (ii) of the form $\pi_V q$, $(\textsf{SEED}\ U\ q)$ or $(\textsf{SEED}\ ?v\ q)$ such that subquery q is in UNION-free normal form. An LDQL query is in **union normal form** if it is of the form $(q_1\ \textsf{UNION}\ \dots\ \textsf{UNION}\ q_n)$ with $n \geq 1$ and each q_i $(1 \leq i \leq n)$ is in UNION-free normal form.

The following result is an immediate consequence of Lemma 2.

Corollary 1. *Every LDQL query is equivalent to an LDQL query in* UNION *normal form.*

In conjunction with Fact 1, Corollary 1 allows us to focus on LDQL queries in UNION normal form without losing generality. We are now ready to specify our procedure that applies the results in this paper to test a given LDQL query q for Web-safeness: First, by using the equivalences in Lemma 2, the query has to be rewritten into a semantically equivalent LDQL query $q_{\textsf{nf}} = (q_1\ \textsf{UNION}\ \dots\ \textsf{UNION}\ q_n)$ that is in UNION normal form. Next, the following test has to be repeated for every subquery q_i $(1 \leq i \leq n)$; recall that each of these subqueries is in UNION-free normal form; i.e., $q_i = (q_1^i\ \textsf{AND}\ \dots\ \textsf{AND}\ q_{m_i}^i)$. The test is to find an order for their subqueries $q_1^i, \dots, q_{m_i}^i$ that satisfies the conditions in Theorem 7. Every top-level subquery q_i $(1 \leq i \leq n)$ for which such an order exists, is Web-safe (cf. Theorem 7). If all top-level subqueries are identified to be Web-safe by this test, then $q_{\textsf{nf}}$ is Web-safe (cf. Proposition 2), and so is q (cf. Fact 1).

The given conditions are sufficient to show Web-safeness of LDQL. It remains open whether there exists a (decidable) sufficient *and* necessary condition for Web-safeness.

6 Concluding Remarks and Future Work

LDQL, the query language that we introduce in this paper, allows users to express queries over Linked Data on the WWW. We defined LDQL such that navigational features for selecting the query-relevant documents on the Web are separate from patterns that are meant to be evaluated over the data in the selected documents. This separation distinguishes LDQL from other approaches to express queries over Linked Data.

We focused on expressiveness, by comparing LDQL with previous formalisms, and on the notion of Web-safeness. Several topics remain open for future work. One of them is the complexity of query evaluation. A classical complexity analysis is easy to perform if we assume that all the data and documents are available as if they were in a centralized repository, and that they can be processed via a RAM machine model. We conjecture that under this model, the data complexity of evaluating LDQL will be polynomial. Nevertheless, a more interesting complexity analysis should consider a model that captures the inherent way of accessing the Web of Linked Data via HTTP requests, the overhead of data communication and transfer, the distribution of data and documents, etc. A more practical direction for future research on LDQL is the development of approaches to actually implement LDQL queries efficiently.

Acknowledgments. Pérez is supported by the Millennium Nucleus Center for Semantic Web Research, Grant NC120004, and Fondecyt grant 1140790.

References

1. Arenas, M., Conca, S., Pérez, J.: Counting beyond a yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In: WWW 2012, pp. 629–638 (2012)
2. Arenas, M., Gutierrez, C., Pérez, J.: On the semantics of SPARQL. In: Semantic Web Information Management - A Model-Based Perspective, chap. 13, pp. 281–307. Springer (2009)
3. Berners-Lee, T.: Linked Data (2006). http://www.w3.org/DesignIssues/LinkedData.html
4. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and optimization of the SPARQL 1.1 federation extension. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 1–15. Springer, Heidelberg (2011)
5. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, February 2014
6. Fielding, R., Gettys, J., Mogul, J.C., Frystyk, H., Masinter, L., Leach, P.J., Berners-Lee, T.: Hypertext Transfer Protocol - HTTP/1.1, June 1999
7. Fionda, V., Pirrò, G., Gutierrez, C.: NautiLOD: A Formal Language for the Web of Data Graph. ACM Transactions on the Web **9**(1), 5:1–5:43 (2015)
8. Harris, S., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 Query Language. W3C Recommendation, March 2013

9. Harth, A., Speiser, S.: On completeness classes for query evaluation on linked data. In: Proc. 26th AAAI Conf. (2012)
10. Hartig, O.: LDQL: a language for linked data queries. In: AMW 2015 (2015)
11. Hartig, O.: SPARQL for a web of linked data: semantics and computability. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 8–23. Springer, Heidelberg (2012)
12. Hartig, O.: An Overview on Execution Strategies for Linked Data Queries. Datenbank-Spektrum **13**(2) (2013)
13. Hartig, O., Pérez, J.: LDQL: A Query Language for the Web of Linked Data (Extended Version). CoRR abs/1507.04614 (2015). http://arxiv.org/abs/1507.04614
14. Hartig, O., Pirrò, G.: A context-based semantics for SPARQL property paths over the web. In: Proc. 12th Extended Semantic Web Conf. (2015)
15. Mendelzon, A.O., Mihaila, G.A., Milo T.: Querying the world wide web. In: PDIS (1996)
16. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006)
17. Pérez, J., Arenas, M., Gutierrez, C.: nSPARQL: A Navigational Language for RDF. J. Web Sem. **8**(4), 255–270 (2010)
18. Umbrich, J., Hogan, A., Polleres, A., Decker, S.: Link Traversal Querying for a Diverse Web of Data. Semantic Web Journal (2014)