LDQL: A Query Language for the Web of Linked Data
(Extended Version)*

Olaf Hartig$^1$ and Jorge Pérez$^2$

$^1$http://olafhartig.de/
$^2$Department of Computer Science, Universidad de 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 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

* This document is an extended version of a paper published in ISWC 2015 [13].
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 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 demonstrate 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.

As our second main contribution we compare LDQL with three previously proposed formalisms for querying the Web of Linked Data: SPARQL under reachability-based 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 demonstrate that LDQL presents an interesting expressive power.

The rest of the paper is structured as follows. Section 2 introduces a data model that provides the basis for defining the semantical properties of LDQL. Section 3 concludes the paper and sketches future work. Proofs of the formal results in this paper can be found in the Appendix.
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 \( (s, p, o) \in \mathcal{T} \) with \( \mathcal{T} = (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L}) \). For any RDF triple \( t = (s, p, o) \) we write \( \text{uris}(t) \) 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 \( \text{data} \), that maps each document \( d \in \mathcal{D} \) to a finite set of RDF triples \( \text{data}(d) \subseteq \mathcal{T} \) 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 \( W = \langle D, \text{adoc} \rangle \) that consists of a set of documents \( D \subseteq \mathcal{D} \) and a partial function \( \text{adoc}: \mathcal{U} \rightarrow D \) that is surjective.

Function \( \text{adoc} \) of a Web of Linked Data \( W = \langle D, \text{adoc} \rangle \) 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 \( u \in \text{dom}(\text{adoc}) \) (i.e., any URI that can be looked up in \( W \)), document \( d = \text{adoc}(u) \) can be considered the authoritative source of data for \( u \) in \( W \) (hence, the name \( \text{adoc} \)). To accommodate for documents that are authoritative for multiple URIs, we do not require injectivity for function \( \text{adoc} \). 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.

Let \( W = \langle D, \text{adoc} \rangle \) 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 \( t \in \text{data}(d) \), we say that a URI \( u \in \text{uris}(t) \) establishes a data link from \( d \) to \( d' \), if \( \text{adoc}(u) = d' \). 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 \( W = \langle D, \text{adoc} \rangle \), is a directed, edge-labeled multigraph, \( \mathcal{G}_W = (D, E_W) \), with set of edges \( E_W \subseteq D \times (\mathcal{T} \times \mathcal{U}) \times D \) defined as \( E_W = \{ (d_{\text{src}}, (t, u), d_{\text{tgt}}) \mid t \in \text{data}(d_{\text{src}}), u \in \text{uris}(t) \text{ and } d_{\text{tgt}} = \text{adoc}(u) \} \).

For a link graph edge \( e = (d_{\text{src}}, (t, u), d_{\text{tgt}}) \), 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 \( W_{ex} = \langle D_{ex}, \text{adoc}_{ex} \rangle \) with three documents, \( d_A \), \( d_B \), and \( d_C \) (i.e., \( D_{ex} = \{ d_A, d_B, d_C \} \)). The data in these documents are the following sets of RDF triples:

\[
\begin{align*}
\text{data}(d_A) &= \{ (u_A, p_1, u_B), (u_B, p_2, u_C) \}; \\
\text{data}(d_B) &= \{ (u_B, p_1, u_C) \}; \\
\text{data}(d_C) &= \{ (u_A, p_2, u_C) \};
\end{align*}
\]
and for function $adoc_{ex}$ we have: $adoc_{ex}(u_A) = d_A$, $adoc_{ex}(u_B) = d_B$, $adoc_{ex}(u_C) = d_C$, and $adoc_{ex}(p_1) = d_A$ (i.e., $\text{dom}(adoc_{ex}) = \{u_A, u_B, u_C, p_1\}$). This Web contains 10 data links. For instance, URI $u_A$ in the RDF triple $(u_A, p_2, u_C) \in \text{data}(d_C)$ establishes a data link to document $d_A$. Hence, the corresponding edge in the link graph of $W_{ex}$ is $\langle d_C, (\langle u_A, p_2, u_C \rangle, u_A), d_A \rangle$. Figure 1 illustrates the link graph $G_{W_{ex}}$ 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 [16,2]. 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 $(U \cup \{\_\} \times (U \cup \{\_\} + \}) \times (U \cup \text{L} \cup \{\_\} + \})$. 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 $(+, p_1, \_)$.

Intuitively, in the context of URI $u_A$, the edge with label $(\langle u_A, p_1, u_B \rangle, u_B)$ from document $d_A$ to document $d_B$, matches the link pattern $(+, p_1, \_)$. 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 $\_$.

On the other hand, the edge with label $(\langle u_A, p_1, u_B \rangle, u_A)$ from $d_A$ to $d_A$, does not match $(+, p_1, \_)$; 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 \( \langle d_A , \langle \langle u_B , p_2 , u_C \rangle , u_B \rangle , d_B \rangle \) and \( \langle d_A , \langle \langle u_B , p_2 , u_C \rangle , u_C \rangle , d_C \rangle \). The next definition formalizes this notion of matching.

**Definition 3.** A link graph edge with label \( \langle \langle x_1 , x_2 , x_3 \rangle , u \rangle \) 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 \( x_i = u_{ctx} \), 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 \( \langle d_{src} , (t, u) , d_{tgt} \rangle \in G_W \) is authoritative in a Web of Linked Data \( W = \langle D, adoc \rangle \) if \( d_{src} = adoc(u') \) 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 \( d_{ctx} = adoc(u_{ctx}) \).

### 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 \( \text{AND} \), \( \text{UNION} \) and projection, to combine them. We also introduce an operator \( \text{SEED} \) 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:

\[
q := \langle lp, P \rangle \mid (\text{SEED} \ U \ q) \mid (\text{SEED} \ ?v \ q) \mid (q \ \text{AND} \ q) \mid (q \ \text{UNION} \ q) \mid \pi_V \ q
\]

\[
lpe := \varepsilon \mid lp \mid lpe/lpe \mid lp[lpe \mid lp]^\ast \mid \{lp\} \mid \langle ?v, q \rangle
\]

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 $G_W$ 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 $G_W$, 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 $\cdot$, 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 $\langle lpe \rangle$, it checks whether $G_W$ contains a path from $d_{ctx} = adoc(u)$ 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 $\text{dataset}_W(\cdot)$ 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, $\text{dataset}_W(U)$ is an RDF dataset (as per [8,2]) that has the set of triples $\{ t \in \text{data}(adoc(u)) \mid u \in U \cap \text{dom}(adoc) \}$ as default graph. Moreover, for every URI $u \in U \cap \text{dom}(adoc)$, $\text{dataset}_W(U)$ contains the named graph $\langle u, \text{data}(adoc(u)) \rangle$.

**Example 2.** Consider the Web $W_{ex}$ in Example 1 and the set of URIs $U = \{u_A, u_C\}$. Then $\text{dataset}_{W_{ex}}(U)$ has $\{\langle u_A, p_1, u_B \rangle, \langle u_B, p_2, u_C \rangle, \langle u_A, p_2, u_C \rangle\}$ as default graph, and two named graphs, $\langle u_A, \{\langle u_A, p_1, u_B \rangle, \langle u_B, p_2, u_C \rangle\} \rangle$ and $\langle u_C, \{\langle u_A, p_2, u_C \rangle\} \rangle$.

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 $D$, an RDF graph $G$ in $D$, and a SPARQL graph pattern $P$, we denote by $[P]_D^G$ the evaluation of $P$ over $G$ in $D$ [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 $\lfloor \cdot \rfloor^S_W$, 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 $\lfloor \cdot \rfloor^u_W$ 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 $W = (D, adoc)$, denoted by $\lfloor \cdot \rfloor^S_W$, is defined recursively as follows:

$$\lfloor (\text{select } P) \rfloor^S_W = [P]_G^D$$ where $\mathcal{D} = \text{dataset}_W \left( \bigcup_{u \in S} \lfloor \text{lpe} \rfloor^u_W \right)$ with default graph $G$, 

$$\lfloor \langle \text{seed } U \text{ q} \rangle \rfloor^S_W = [q]^U_W,$$

$$\lfloor \langle \text{seed } ?v \text{ q} \rangle \rfloor^S_W = \bigcup_{u \in \mathcal{U}} \left( \lceil [q]^u_W \rceil \times \{\mu_u\} \right) \text{ where } \mu_u = \{?v \mapsto u\} \text{ for all } u \in \mathcal{U},$$

$$\lfloor (\text{q1 UNION q2}) \rfloor^S_W = \lfloor q_1 \rfloor^S_W \cup \lfloor q_2 \rfloor^S_W,$$

$$\lfloor (\text{q1 AND q2}) \rfloor^S_W = \lfloor q_1 \rfloor^S_W \times \lfloor q_2 \rfloor^S_W,$$

$$\lfloor \pi_V^q \rfloor^S_W = \{\mu | \text{there exists } \mu' \in \lfloor q \rfloor^S_W \text{ such that } \mu \text{ and } \mu' \text{ are compatible and } \text{dom}(\mu) = \text{dom}(\mu') \cap V\}.$$

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

$$\lfloor \varepsilon \rfloor^{u_{\text{ctx}}}_W = \{u_{\text{ctx}}\},$$

$$\lfloor \text{lp} \rfloor^{u_{\text{ctx}}}_W = \{u \in \mathcal{U} | \text{there exist a link graph edge } (d_{\text{src}}, (t, u), d_{\text{tgt}}) \in \mathcal{G}_W, \text{ with } d_{\text{src}} = \text{adoc}(u_{\text{ctx}}), \text{ that matches } \text{lp} \text{ in the context of } u_{\text{ctx}}\},$$

$$\lfloor \text{lpe}_1 / \text{lpe}_2 \rfloor^{u_{\text{ctx}}}_W = \{u \in \lfloor \text{lpe}_2 \rfloor^{u_{\text{ctx}}}_W | u' \in \lfloor \text{lpe}_1 \rfloor^{u_{\text{ctx}}}_W \},$$

$$\lfloor \text{lpe}_1 \times \text{lpe}_2 \rfloor^{u_{\text{ctx}}}_W = \lfloor \text{lpe}_1 \rfloor^{u_{\text{ctx}}}_W \cup \lfloor \text{lpe}_2 \rfloor^{u_{\text{ctx}}}_W,$$

$$\lfloor \text{lpe} \star \rfloor^{u_{\text{ctx}}}_W = \{u_{\text{ctx}}\} \cup \lfloor \text{lpe} \rfloor^{u_{\text{ctx}}}_W \cup \lfloor \text{lpe} / \text{lpe} \rfloor^{u_{\text{ctx}}}_W \cup \lfloor \text{lpe} / \text{lpe} / \text{lpe} \rfloor^{u_{\text{ctx}}}_W \cup \ldots,$$

$$\lfloor \text{lpe} \rfloor^{u_{\text{ctx}}}_W = \{u_{\text{ctx}} | \lfloor \text{lpe} \rfloor^{u_{\text{ctx}}}_W \neq \emptyset\},$$

$$\lfloor (\text{?v, q}) \rfloor^{u_{\text{ctx}}}_W = \{u \in \mathcal{U} | \text{there exists } \mu \in \lfloor q \rfloor^{u_{\text{ctx}}}_W \text{ such that } \mu(\text{?v}) = u\}.$$

Moreover, if $u_{\text{ctx}} \notin \text{dom}(\text{adoc})$, then $\lfloor \text{lpe} \rfloor^{u_{\text{ctx}}}_W = \emptyset$ for every LPE.

**Example 3.** Let $\text{lpe}_{ex}$ 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 $d_A = \text{adoc}_{ex}(u_A)$ and $d_C = \text{adoc}_{ex}(u_C)$. More precisely, we have $\lfloor \text{lpe}_{ex} \rfloor^{u_A}_{W_{ex}} = \{u_A, u_C\}$. 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 $S_{ex} = \{u_A\}$ and a basic LDQL query $\langle \text{lpe}_{ex}, B_{ex} \rangle$ whose LPE is $\text{lpe}_{ex}$ as introduced in Example 3 and whose SPARQL graph pattern is a basic graph pattern that contains two triple patterns, $B_{ex} = \{\langle ?x, p_1, ?y \rangle, \langle ?x, p_2, ?z \rangle\}$. Given that we have $\lfloor \text{lpe}_{ex} \rfloor^{u_A}_{W_{ex}} = \{u_A, u_C\}$ (cf. Example 3), dataset $W_{ex}(\lfloor \text{lpe}_{ex} \rfloor^{u_A}_{W_{ex}})$ has the default graph $\{\langle u_A, p_1, u_B \rangle, \langle u_B, p_2, u_C \rangle, \langle u_A, p_2, u_C \rangle\}$ (cf. Example 2). Then, according to the query semantics, the result of query $\langle \text{lpe}_{ex}, B_{ex} \rangle$ over $W_{ex}$ using seeds $S_{ex}$ consists of a single solution mapping, namely $\mu = \{?x \mapsto u_A, ?y \mapsto u_B, ?z \mapsto u_C\}$. 
Example 5. Consider an LDQL query \( q_{ex} = \langle \text{SEED} \ ?x \ (\varepsilon, \langle ?x, p_1, ?w \rangle) \rangle \) whose sub-query is a basic LDQL query that has a single triple pattern as its SPARQL graph pattern. Additionally, let \( q'_{ex} = \langle \text{lpex}, \{ (?x, p_1, ?y), \langle ?x, p_2, ?z \rangle \} \rangle \) be the basic LDQL query introduced in Example 4, and let \( q''_{ex} \) be the conjunction of these two queries; i.e., \( q''_{ex} = (q_{ex} \text{ AND } q'_{ex}) \). By Example 4 we know that \([q_{ex}]^S_{ex} = \{ \mu \} \) with \( \mu = \{ ?x \mapsto u_A, ?y \mapsto u_B, ?z \mapsto u_C \} \). Furthermore, based on the data given in Example 1, it is easy to see that \([q'_{ex}]^S_{ex} = \{ \mu_1, \mu_2 \} \) with \( \mu_1 = \{ ?x \mapsto u_A, ?w \mapsto u_B \} \) and \( \mu_2 = \{ ?x \mapsto u_B, ?y \mapsto u_C \} \). For the \( S_{ex} \)-based evaluation of \( q''_{ex} \) over \( W_{ex} \), the result sets \([q_{ex}]^S_{ex} \) and \([q'_{ex}]^S_{ex} \) have to be joined. Thus, we need to compute \( \{ \mu_1, \mu_2 \} \times \{ \mu \} \), which results in a single mapping \( \mu' = \mu_1 \cup \mu = \{ ?x \mapsto u_A, ?w \mapsto u_C, ?y \mapsto u_B, ?z \mapsto u_C \} \).

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 \([q]_V^S = [q']_V^S \) 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:

\[
(q_1 \text{ AND } (q_2 \text{ UNION } q_3)) \equiv ((q_1 \text{ AND } q_2) \text{ UNION } q_3) \quad (1) \\
\pi_V(q_1 \text{ UNION } q_2) \equiv (\pi_V q_1 \text{ UNION } \pi_V q_2) \quad (2) \\
(\text{SEED } U (q_1 \text{ UNION } q_2)) \equiv ((\text{SEED } U q_1) \text{ UNION } (\text{SEED } U q_2)) \quad (3) \\
(\text{SEED } ?v (q_1 \text{ UNION } q_2)) \equiv ((\text{SEED } ?v q_1) \text{ UNION } (\text{SEED } ?v q_2)) \quad (4)
\]

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 \( l_p \), concatenation \( / \), disjunction \( \mid \), and the test \( [ \cdot ] \), are just syntactic sugar as they can be simulated by using \( \varepsilon, \langle ?v, q \rangle \) 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 \( \varepsilon \), 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 \( \langle ?v, \langle \varepsilon, \langle \text{GRAPH} ?u (\langle ?u, p, ?v \rangle) \rangle \rangle \rangle \), and we can similarly encode all types of link patterns. To encode \( / \) we make use of \( \langle ?v, q \rangle \) and the operator \( \text{AND} \) 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:

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

Similarly, to encode \( \mid \) we make use of UNION and to encode \([ \cdot ] \) we use projection.

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

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\(^3\), 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 \( \alpha \) and \( \beta \) 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 \( u \in \text{dom(adoc)} \) returns the RDF triples in \( \text{data(adoc(u))} \) that have \( u \) in the subject position. Formally, for every URI \( u \in \text{dom(adoc)} \) we have \( C^W(u) = \{ \langle s, p, o \rangle \in \text{data(adoc(u))} \mid s = u \} \). To simplify the exposition, the authors extended the definition of \( C^W(\cdot) \) to also handle URIs not in \( \text{dom(adoc)} \), and literals and blank nodes. For any such RDF term \( a \) they define \( C^W(a) \) as the empty set.

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 \( \left[ \cdot \right]_{W}^{\text{ext}} \), 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 \( \mu \) and an RDF term \( \alpha \), we

\(^3\) In [14] the reverse path construction \( ^\text{rev}pe \) 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.
use the notation $\mu[\alpha]$ with the following meaning: $\mu[\alpha] = \mu(\alpha)$ if $\alpha \in \text{dom}(\mu)$, and $\mu[\alpha] = \alpha$ in the other case. Similarly, $\mu[(s, p, o)] = (\mu[s], \mu[p], \mu[o])$.

$\boxed{[(\alpha, p, \beta)]^\text{ctx}_W = \{ \mu | \text{dom}(\mu) = \{\alpha, \beta\} \cap V \text{ and } \mu[(\alpha, p, \beta)] \in C_W(\mu[\alpha]) \}$

$\boxed{\text{dom}(\mu) = \{\alpha, \beta\} \cap V \text{ and exists } p \text{ s.t. }}$

$\boxed{\mu[(\alpha, p, \beta)] \in C_W(\mu[\alpha]) \text{ and } p \notin \{u_1, ..., u_k\} \}$

$\boxed{[(\alpha, pe_1/pe_2, \beta)]^\text{ctx}_W = \pi_{\{\alpha, \beta\} \cap V}([(\alpha, pe_1, ?v)]^\text{ctx}_W \times [(?v, pe_2, \beta)]^\text{ctx}_W)}$

$\boxed{([(\alpha, pe_1, \beta)]^\text{ctx}_W \cup [(\alpha, pe_2, \beta)]^\text{ctx}_W)}$

$\boxed{([(\alpha, pe, \beta)]^\text{ctx}_W \cup [(\alpha, pe/pe, \beta)]^\text{ctx}_W \cup [(\alpha, pe/pe/pe, \beta)]^\text{ctx}_W)}$

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 $q = \{\text{SEED } U \langle \langle +, p, _- \rangle, (?x, ?y) \rangle\}$ with $U = \{u\}$ cannot be expressed by PP patterns under context-based semantics because this semantics is “blind” to triples that are not authoritative. For instance, in a Web $W = \{(d, d'), \text{adoc}\}$ with $\text{data}(d) = \{\langle u, p, u \rangle\}$, $\text{data}(d') = \{\langle u', p, u \rangle, \langle u, u, u \rangle\}$, $\text{adoc}(u) = d$ and $\text{adoc}(u') = d'$, 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 $d' = \text{adoc}(u') \neq \text{adoc}(u)$. 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 $\alpha, \beta \in U \cup L \cup V$. Then, for every PP-pattern $\langle \alpha, pe, \beta \rangle$, there exists an LDQL query $q$ such that $\boxed{\{\{\alpha, pe, \beta\}\}^\text{ctx}_W = \{\{q\}^\emptyset_W \}}$ 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 $\text{dom}(\text{adoc})$, 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 $\pi_{\{?x, ?y\}}(Q_{pe_1}(?x, ?y) \land Q_{pe_2}(?x, ?y))$, and for $\langle ?x, \langle u_1 \cdot \cdot \cdot u_k \rangle, ?y \rangle$ the translation is $(\text{SEED } ?x \langle \varepsilon, \langle ?x, ?p, ?y \rangle \text{ FILTER } ?p \neq u_1 \land \cdot \cdot \cdot \land ?p \neq u_k \rangle)$. To handle $pe^*$ we need to use the construction $(\langle q, q \rangle)^\emptyset$ of LPEs, plus $\langle \cdot \rangle^*$.}

### 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 $\rho \in U$ and $P$ is a SPARQL graph pattern).

$$ne := p \mid p^* \mid \langle \_ \rangle \mid ne/ne \mid ne\text{ne} \mid ne^* \mid ne\{(\text{ASK } P)\}$$
In terms of our data model\footnote{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 $\text{dom}(\text{adoc})$. Hence, to properly capture the semantics of NautiLOD in terms of our data model we have to introduce conditions of the form “$u' \in \text{dom}(\text{adoc})$.”}, the semantics of NautiLOD expressions over a Web of Linked Data $W = \langle D, \text{adoc} \rangle$ from URI $u \in \text{dom}(\text{adoc})$ is defined recursively as follows.

$$\{ p \}^W = \{ u' \mid \langle u, p, u' \rangle \in \text{data}(\text{adoc}(u)) \}$$
$$\{ p' \}^W = \{ u' \mid \langle u', p, u' \rangle \in \text{data}(\text{adoc}(u)) \}$$
$$\{ \bot \}^W = \{ u' \mid \langle u, p, u' \rangle \in \text{data}(\text{adoc}(u)) \text{ for some } p \in U \}$$
$$\{ \text{ne}_1/\text{ne}_2 \}^W = \{ u'' \mid u'' \in \{ \text{ne}_2 \}^W_u \text{ for some } u' \in \{ \text{ne}_1 \}^W_u \text{ with } u' \in \text{dom}(\text{adoc}) \}$$
$$\{ \text{ne}_1 \text{ne}_2 \}^W = \{ \text{ne}_1 \}^W_u \cup \{ \text{ne}_2 \}^W_u$$
$$\{ \text{ne}^* \}^W = \{ u \cup \{ \text{ne} \}^W_u \cup \{ \text{ne}/\text{ne} \}^W_u \cup \{ \text{ne}/\text{ne}/\text{ne} \}^W_u \cup \cdots \}$$
$$\{ \text{ne}(\text{ASK } P) \}^W = \{ u' \mid u' \in \{ \text{ne} \}^W_u, u' \in \text{dom}(\text{adoc}) \text{ and } \{ P \}^W_{\text{data}(\text{adoc}(u'))} \neq \emptyset \}$$

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 $\text{ne}$ are equivalent if for every Web of Linked Data $W = \langle D, \text{adoc} \rangle$ and URIs $u, u'$ with $u \in \text{dom}(\text{adoc})$ it holds that $u' \in \{ \text{ne} \}^W_u$ if and only if $\{ ?x \rightarrow u' \} \in \{ q(?x) \}^W_{\{ u \}}$.

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

**Proof (Sketch).** The proof begins with a simple translation that replaces every $p \in U$ in a NautiLOD expression by a link pattern $\langle +, p, \bot \rangle$. For instance, the expression $p_1/p_2$ is translated into $\langle +, p_1, \bot \rangle/\langle +, p_2, \bot \rangle$. To translate $\bot$ and $\{ \text{ASK } P \}$ we use $\langle ?v, q \rangle$. The complete translation poses several other complications (as described in the appendix). 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 $\text{UNION}$ 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 \( c : T \times U \times P \to \{ \text{true}, \text{false} \} \) where \( P \) 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 \( d \in D \) is \((c, S, P)\)-reachable in a Web of Linked Data \( W = \langle D, adoc \rangle \) if any of the following two conditions hold:

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

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 \( c_{\text{All}} \), \( c_{\text{None}} \), and \( c_{\text{Match}} \) [11], where \( c_{\text{All}} \) selects all data links, and \( c_{\text{None}} \) ignores all data links; i.e., \( c_{\text{All}}(t, u, P) = \text{true} \) and \( c_{\text{None}}(t, u, P) = \text{false} \) for all triples \( (t, u, P) \in T \times U \times P \). In contrast to such an all-or-nothing strategy, criterion \( c_{\text{Match}} \) returns \text{true} for every data link whose triple matches a triple pattern of the given graph pattern; formally, \( c_{\text{Match}}(t, u, P) = \text{true} \) if and only if there exists some solution mapping \( \mu \) 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 \( [P]_W^{(S,c)} \), 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 \( c_{\text{All}} \), \( c_{\text{None}} \), and \( c_{\text{Match}} \). 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 \( c \in \{ c_{\text{All}}, c_{\text{None}}, c_{\text{Match}} \} \). For every SPARQL graph pattern \( P \) there exists an LDQL query \( q \) such that \( [P]_W^{(S,c)} = [q]_W^S \) for every Web \( W \) and \( S \subseteq U \).

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

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

### 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 \(\text{adoc}\) (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 \(W = \langle D, \text{adoc}\rangle\) 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 \(\text{dom} (\text{adoc})\).

**Example 6.** Recall our example queries \(q_{ex}\), \(q'_{ex}\), and \(q''_{ex}\) (cf. Example 5). For query \(q_{ex} = (\text{SEED} \cap x \cap \langle \langle x, p_1, ?z \rangle \rangle)\), 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 \(\text{dom}(\text{adoc})\) of the queried Web \(W = \langle D, \text{adoc}\rangle\), 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 \(q''_{ex} = (\text{SEED} \cap x \cap \langle \langle x, p_1, ?z \rangle \rangle)\) is Web-safe, and so is \(q'_{ex} = (\text{adoc}_x, B_{ex})\). Given \(u\) as seed URI, a possible execution algorithm for \(q'_ex\) may first compute \([\text{adoc}_x]_{adoc}^u\) by traversing the queried Web \(W\) based on \(\text{adoc}_x\). Thereafter, the algorithm retrieves documents by looking up all URIs \(u \in [\text{adoc}_x]_W^u\) (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 \((\text{SEED} \cap x \cap q)\) 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 \((\pi \cdot q)\), \((\pi \cdot q)\), \((\pi \cdot q)\), and \((\pi \cdot q)\).
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 \( \langle \delta, q' \rangle \) and LDQL query \( q' \) is Web-safe, or (ii) it is of any form other than \( \langle \delta, q' \rangle \) and all its subexpressions (if any) are Web-safe LPEs;
2. Query \( q \) is of the form \( \pi_V q' \) or \( \langle \text{SEED} \ U q' \rangle \), and subquery \( q' \) is Web-safe; or
3. Query \( q \) is of the form \( q_1 \cup \ldots \cup q_n \) and each \( q_i \) \( (1 \leq i \leq n) \) is Web-safe.

It remains to discuss LDQL queries of the form \( \langle q_1 \text{ AND} \ldots \text{ AND} q_m \rangle \). 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 \( \langle \text{SEED} \ ?v \ q \rangle \) 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 \( q_1 \cup q_2 \), 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 \( \langle \text{SEED} \ U q' \rangle \), then \( \text{sbvars}(q) = \text{sbvars}(q') \).
6. If \( q \) is of the form \( \langle \text{SEED} \ ?v q' \rangle \), 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 \( \mu \in [q]_W^\mu \), 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 \( \langle q_1 \text{ AND} q_2 \text{ AND} \ldots \text{ AND} q_m \rangle \) is Web-safe if there exists a total order \( \prec \) over the set of subqueries \( \{q_1, q_2, \ldots, q_m\} \) such that for each subquery \( q_i \), \( (1 \leq i \leq m) \), it holds that either (i) \( q_i \) is Web-safe or (ii) \( q_i \) is of the form \( \langle \text{SEED} \ ?v q \rangle \) where \( q \) is Web-safe and \( ?v \in \bigcup_{i \neq j} \text{sbvars}(q_j) \).

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 \( q_1 \ldots q_m \) 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 \((\text{SEED} ?v q)\). For instance, the theorem cannot be applied to show that an LDQL query of the form \(((q_1 \text{ AND } (q_2 \text{ UNION } (\text{SEED} ?x q_3))))\) 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 \(((q_1 \text{ AND } q_2) \text{ UNION } (q_1 \text{ AND } (\text{SEED} ?x q_3))))\) 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 \((q_1 \text{ AND } \ldots \text{ AND } q_m)\) 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 \(\pi_V q, (\text{SEED } U q)\) or \((\text{SEED} ?v q)\) such that subquery \(q\) is in UNION-free normal form. An LDQL query is in **UNION normal form** if it is of the form \((q_1 \text{ UNION } \ldots \text{ UNION } q_n)\) 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 \(q_{eq} = (q_1 \text{ UNION } \ldots \text{ UNION } q_n)\) 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., \(q_i = (q_i^1 \text{ AND } \ldots \text{ AND } q_i^{m_i})\). The test is to find an order for their subqueries \(q_i^1, \ldots, q_i^{m_i}\) 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 \(q_{eq}\) 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.

Acknowledgements 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. At http://www.w3.org/DesignIssues/LinkedData.html (2006)
4. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and Optimization of the SPARQL 1.1 Federation Extension. In: Proc. 8th Extended Semantic Web Conf. (2011)
5. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation (Feb 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 (Jun 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 (Mar 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
11. Hartig, O.: SPARQL for a Web of Linked Data: Semantics and Computability. In: Proc. 9th Extended Semantic Web Conf. (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. In: Proc. 14th Int. Semantic Web Conf. (2015)
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. ACM Transactions on Database Systems 34 (2009)
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)
A Proofs

A.1 Proof of Lemma 1

We formalize the claims in Lemma 1 as follows: Let \( q_1, q_2, \) and \( q_3 \) be LDQL queries, the following semantic equivalences hold:

\[
(q_1 \text{ AND } q_2) \equiv (q_2 \text{ AND } q_1) \quad (5)
\]
\[
(q_1 \text{ UNION } q_2) \equiv (q_2 \text{ UNION } q_1) \quad (6)
\]
\[
(q_1 \text{ AND } (q_2 \text{ AND } q_3)) \equiv ((q_1 \text{ AND } q_2) \text{ AND } q_3) \quad (7)
\]
\[
(q_1 \text{ UNION } (q_2 \text{ UNION } q_3)) \equiv ((q_1 \text{ UNION } q_2) \text{ UNION } q_3) \quad (8)
\]

Since the definition of LDQL operators AND and UNION is equivalent to their SPARQL counterparts, these semantic equivalences follow from corresponding equivalences for SPARQL graph patterns as shown by Pérez et al. [16, Lemma 2.5].

A.2 Proof of Lemma 2

The equivalences follow directly from the definition of every operator.

A.3 Proof of Proposition 1

The proof is based on a recursive translation of link path expressions beginning with link patterns. Let \( (y_1, y_2, y_3) \) be a link pattern. We construct an LPE \( \text{trans}_L((y_1, y_2, y_3)) \) as follows. Assume that \( y_1 = \_ \), then we construct the LDQL query

\[
q_1 = \langle \varepsilon, (\text{GRAPH}?u (\text{?out}, Y_2, Y_3)) \rangle
\]

where (i) if \( y_2 = + \) then \( Y_2 = ?u \), (ii) if \( y_2 \in \mathcal{U} \) then \( Y_2 = y_2 \) and (iii) if \( y_2 = \_ \) then \( Y_2 = ?y_2 \). And similarly, if \( y_3 = + \) then \( Y_3 = ?u \), (ii) if \( y_3 \in \mathcal{U} \) then \( Y_3 = y_3 \) and (iii) if \( y_3 = \_ \) then \( Y_3 = ?y_3 \). If \( ?y_2 = \_ \) then we can construct query \( q_2 = \langle \varepsilon, (\text{GRAPH}?u (Y_1, \text{?out}, Y_3)) \rangle \), and if \( ?y_3 = \_ \) query \( q_3 = \langle \varepsilon, (\text{GRAPH}?u (Y_1, Y_2, \text{?out})) \rangle \), following a similar process as for \( q_1 \). Now consider the query \( q \) which is the UNION of the above queries for every \( y_i = \_ \). Then LPE \( \text{trans}_L((y_1, y_2, y_3)) \) is constructed as

\[
\text{trans}_L((y_1, y_2, y_3)) = \langle \text{?out}, q \rangle.
\]

It is not difficult to prove that \( \text{trans}_L((y_1, y_2, y_3)) \) is constructed as follows:

- For the case of LPE \( r = r_1/r_2 \), we have that \( \text{trans}_L(r) = \langle \varepsilon, q \rangle \) where \( q \) is:
  \[
  \{ (\text{trans}_L(r_1), (\text{GRAPH}?x \{ \})) \text{ AND } (\text{SEED}?x (\text{trans}_L(r_2), (\text{GRAPH}?v \{ \})) \} \).
  \]
- For the case of LPE \( r = r_1|r_2 \), we have that \( \text{trans}_L(r) = \langle \varepsilon, q \rangle \) where \( q \) is:
  \[
  \{ (\text{trans}_L(r_1), (\text{GRAPH}?v \{ \})) \text{ UNION } (\text{trans}_L(r_2), (\text{GRAPH}?v \{ \})) \} \).
For the case of LPE \( r = [r_1] \), we have that \( \text{trans}_L(r) = \langle ?v, q \rangle \) where \( q \) is:

\[
\langle \varepsilon, (\text{GRAPH} \ ?v \ { }) \rangle \text{ AND } \pi_{\{?v\}}(\text{SEED} \ ?v \ (\text{trans}_L(r_1), (\text{GRAPH} \ ?x \ { })))
\]

The general proof proceeds by induction. We next prove that \( \llbracket \text{trans}_L(r_1 | r_2) \rrbracket^u_W = \llbracket r_1 | r_2 \rrbracket^u_W \). The proof for the other cases are similar. Thus assume that \( u' \in \llbracket r_1 | r_2 \rrbracket^u_W \), then we know that \( u' \in \llbracket r_1 \rrbracket^u_W \cup \llbracket r_2 \rrbracket^u_W \). If \( u' \in \llbracket r_1 \rrbracket^u_W \) then by induction hypothesis we know that \( u' \in \llbracket \text{trans}_L(r_1) \rrbracket^u_W \). Now notice that

\[
\llbracket (\text{trans}_L(r_1), (\text{GRAPH} \ ?v \ { })) \rrbracket^u_W = \llbracket (\text{GRAPH} \ ?v \ { }) \rrbracket^P_W
\]

Where \( D = \text{dataset}_W((\llbracket \text{trans}_L(r_1) \rrbracket^u_W). \) Thus given that \( u' \in \llbracket \text{trans}_L(r_1) \rrbracket^u_W \) we know that \( D \) has a dataset \langle u', \text{data}(\text{adoc}(u')) \rangle, which implies that \{?v \rightarrow u'\} is a solution for \( (\text{GRAPH} \ ?v \ { }) \rrbracket^P \), and thus \{?v \rightarrow u'\} \in \llbracket (\text{trans}_L(r_1), (\text{GRAPH} ?v \ { })) \rrbracket^u_W \). From this it is straightforward to conclude that \( u' \in \llbracket \text{trans}_L(r_1 | r_2) \rrbracket^u_W \). The other direction is similar.

### A.4 Proof of Theorem 1

Consider the LDQL \( Q \) query given by

\[
(\text{SEED} \ u \ (\langle +, p, \_ \rangle, (?x, ?x, ?x)))
\]

with \( u, p \in \mathcal{U} \). Now assume that there exists a property path pattern \( P \) and a set of URIs \( S \) such that

\[
\llbracket P \rrbracket^s_W = \llbracket Q \rrbracket^s_W
\]

for every Web of Linked Data \( W \). Let \( u' \in \mathcal{U} \). Consider now \( W_1 \) having only two documents \( d_1 = \{p, u, u'\} \) and \( d_2 = \{a, a, a\} \) such that \( \text{adoc}(u) = d_1 \) and \( \text{adoc}(u') = d_2 \). Moreover, consider \( W_2 \) having also two documents \( d_1 = \{u, p, u'\} \) and \( d_3 = \{b, b, b\} \) such that \( \text{adoc}(u) = d_1 \) and \( \text{adoc}(u') = d_3 \). First notice that for every \( S \) we have that

\[
\llbracket Q \rrbracket^s_{W_1} = \{?x \rightarrow a\} \neq \llbracket Q \rrbracket^s_{W_2} = \{?x \rightarrow b\}
\]

Notice that \( C^{W_1}(u) = C^{W_2}(u) = \{p, u, u'\} \) and \( C^{W_1}(u') = C^{W_2}(u') = \emptyset \). In general, we have that for every term \( v \neq u \) it holds that \( C^{W_1}(v) = C^{W_2}(v) = \emptyset \). This essentially shows that the context selectors \( C^{W_1} \) and \( C^{W_2} \) are equivalent. Given that the semantics of property paths is based on context selectors it is easy to prove that for every PP-based SPARQL query \( R \) we have that \( \llbracket R \rrbracket^s_{W_1} = \llbracket R \rrbracket^s_{W_2} \). This can be done by induction in the construction of PP-based SPARQL queries. For example, the evaluation of a base PP-pattern of the form \( (v, p, \beta) \), with \( v \in \mathcal{U} \) and \( \beta \in \mathcal{U} \cup \mathcal{V} \) over \( W_1 \) is given by

\[
\llbracket (v, p, \beta) \rrbracket^s_{W_1} = \{\mu \mid \text{dom}(\mu) = \{\beta\} \cap \mathcal{V} \text{ and } \mu(\{v, p, \beta\}) \in C^{W_1}(v)\}
\]

which is equal to \( \llbracket (v, p, \beta) \rrbracket^s_{W_2} \) since \( C^{W_1}(v) = C^{W_2}(v) \). All the other cases for the construction of property paths are equivalent. Moreover, since for the case of property path patterns the evaluation is the same over \( W_1 \) and over \( W_2 \), we have that for a general
PP-based query using operator \texttt{AND}, \texttt{UNION}, \texttt{OPT} and so on, the evaluation is also the same. Thus we have that

$$[P]_{W_1}^{\text{ctx}} = [P]_{W_2}^{\text{ctx}}$$

but also that

$$[Q]_{W_1}^S \neq [Q]_{W_2}^S$$

which contradicts the fact that $[P]_{W}^{\text{ctx}} = [Q]_{W}^S$ for every Web of Linked Data $W$.

### A.5 Proof of Theorem 2

We associate to every property-path expression $r$, an LDQL query $Q_r(\?x, \?y)$ with $\?x$ and $\?y$ as free variables. The definition of $Q_r(\?x, \?y)$ is by induction in the construction of property-path expressions. In the construction, all the variables mentioned, besides $\?x$ and $\?y$, are considered as fresh variables.

- If $r \in \mathcal{U}$ then $Q_r(\?x, \?y) = (\text{SEED } \langle \varepsilon, (\?x, r, \?y) \rangle)$.
- If $r = !(u_1 | \cdots | u_k)$ with $u_i \in \mathcal{U}$ then $Q_r(\?x, \?y)$ is defined as
  $$(\text{SEED } \langle \varepsilon, ((\?x, ?p, ?y) \text{ FILTER } (\?p \neq u_1 \land \cdots \land \?p \neq u_k) \rangle) \rangle.$$
- If $r = r_1 / r_2$ then $Q_r(\?x, \?y)$ is defined as
  $$\pi_{\{?x, ?y\}}(Q_{r_1}(\?x, ?z) \text{ AND } Q_{r_2}(\?z, ?y)).$$
- If $r = r_1 \mid r_2$ then $Q_r(\?x, \?y)$ is defined as
  $$(Q_{r_1}(\?x, ?y) \text{ UNION } Q_{r_2}(\?x, ?y)).$$
- If $r = r_1^*$ then $Q_r(\?x, \?y)$ is defined as follows. First consider the LDQL query
  $$Q_{e}(\?x, \?y) = \pi_{\{\?x, \?y\}}(\text{SEED } \langle \varepsilon, P \rangle)$$
  where $P$ is the following pattern
  $$P = ((\?x, ?p, ?o) \text{ AND } (\?y, ?p, ?o) \text{ FILTER } (\?x = ?y)) \text{ UNION } \ldots$$

Now consider the LDQL query $Q_s(\?v)$ defined as

$$Q_s(\?v) = \langle \varepsilon, (\text{GRAPH } ?u \{ \}) \rangle \text{ AND } Q_{r_1}(\?u, ?v).$$

Then, query $Q_r(\?x, \?y)$ is defined by

$$Q_{e}(\?x, \?y) \text{ UNION } ((\text{SEED } \langle \varepsilon, (Q_s(\?v))^* \rangle, (\text{GRAPH } z \{ \})) \rangle \text{ AND } Q_{r_1}(\?z, ?y)$$
We prove now that for every property path pattern \(?x, r, ?y\) we have that
\[
[(?x, r, ?y)]^\text{ctst}_W = [Q_r(?x, ?y)]^0_W.
\]

The proof is by induction in the construction of \(Q_r(?x, ?y)\). We proceed by cases.

- Assume that \(r \in \mathcal{U}\). Then \(\mu \in [Q_r(?x, ?y)]^0_W\) if and only if
  \[
  \mu \in [[\langle \text{SEED} \; ?x \; \langle \varepsilon, (?x, r, ?y) \rangle \rangle \; \text{FILTER} \; (?p \neq u_1 \land \cdots \land ?p \neq u_k)]^0_W.
  \]
  Notice that this occurs if and only if there exists a mapping \(\mu'\) and a URI \(u\) such that \(\mu' \in [[\langle \varepsilon, (?x, r, ?y) \rangle \rangle^u_W\), \(\mu'\) is compatible with the mapping \(\{ ?x \rightarrow u \}\), and \(\mu = \mu' \cup \{ ?x \rightarrow u \}\). Now, given that \([\varepsilon]^u_W = \{ u \}\), we have that \(\mu' \in [[\langle \varepsilon, (?x, r, ?y) \rangle \rangle^u_W\) if and only if \(\mu' \in [[(?x, r, ?y)]^D_W\) with \(D\) the data set with \(\text{data}(\text{adoc}(u))\) as default graph. With all this we have that \(\mu \in [[(?x, r, ?y)]^0_W\) if and only if \(\text{dom}(\mu) = \{ ?x, ?y \}, \mu(?x) \in \text{dom}(\text{adoc}), \text{and} \ (\mu(?x), r, \mu(?y)) \in \text{data}(\text{adoc}(\mu(?x)))\), which is exactly the property
  \[
  \mu((?x, r, ?y)) \in C_W^W(\mu(?x)).
  \]
  This last property holds if and only if \(\mu \in [[(?x, r, ?y)]^\text{ctst}_W\).

- For the case in which \(r = !(u_1 \mid \cdots \mid u_k)\) with \(u_i \in \mathcal{U}\), the proof is similar. We have that
  \[
  \mu \in [[\langle \text{SEED} \; ?x \; \langle \varepsilon, (?x, ?p, ?y) \rangle \; \text{FILTER} \; (?p \neq u_1 \land \cdots \land ?p \neq u_k)\rangle]_W^0
  \]
  if and only if \(\mu\) is in the evaluation of
  \[
  ([?x, ?p, ?y] \; \text{FILTER} \; (?p \neq u_1 \land \cdots \land ?p \neq u_k))
  \]
  over the graph \(\text{data}(\mu(?x))\). This happens if and only if
  \[
  \mu((?x, ?p, ?y)) \in C_W^W(\mu(?x))\text{for }p \notin \{ u_1, \ldots, u_k \},
  \]
  which is exactly the property
  \[
  \mu \in [[(?x, !(u_1 \mid \cdots \mid u_k), ?y)]^\text{ctst}_W.
  \]

- For the cases \(r = r_1/r_2\), \(r = r_1 | r_2\), the semantics of the corresponding LDQL query exactly matches the semantics of the property path expression. Just notice that the semantics of \(\text{AND}\) is that of the join, and the semantics of \(\text{UNION}\) is that of the set union.

- For the case of \(r = r_1^+\) we have that \(\mu \in [[(?x, r_1^+, ?y)]^\text{ctst}_W\) if and only if \(\text{dom}(\mu) = \{ ?x, ?y \}\) and (i) \(\mu(?x) = \mu(?y)\) and \(\mu(?x), \mu(?y) \in \text{terms}(W)\), or (ii) \(\mu \in [[(?x, r_1^+, ?y)]^\text{ctst}_W\) for some \(k > 0\). For the case (i) it is easy to see that \(\mu \in [[Q_r(?x, ?y)]^0_W\). Just notice that if \(\mu(?x)\) is in \(\text{terms}(W)\) then there exists a URI \(u \in \text{dom}(\text{adoc})\) and a triple \(t\) in \(\text{data}(\text{adoc}(u))\) such that \(\mu(?x)\) appears in \(t\). If \(\mu(?x)\) appears in the subject position, then we know that \(\mu\) is compatible with a mapping in
Then we know that there exists apply the same argument as in the base case. Now assume that this is because we will show that if \( \mu \in \{ \text{SEED} \} \) then it is easy to see that by induction hypothesis on the construction of property paths, we have that \( \mu \in [Q_e(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})]_W^0 \) and thus \( \mu \in [Q_e(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})]_W^0 \). If \( \mu(\texttt{?}\texttt{x}) \) appears in the predicate or object position, the proof is similar. For the case (ii) we will show that

\[
\mu \in [Q_e(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})]_W^0 \cup \\
\bigcup_{i=0}^{k-1} [\pi(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})]((\texttt{SEED} \texttt{?}\texttt{x} ((\texttt{?}\texttt{v}, Q_e(\texttt{?}\texttt{v})))^1, (\texttt{GRAPH} ?z \{ \}))) \text{ AND } Q_r(\texttt{?}\texttt{z}, \texttt{?}\texttt{y})]_W^0 \quad (9)
\]

We will use an inductive argument. Assume \( k = 1 \), then \( \mu \in [\{ \texttt{?}\texttt{x}, \texttt{r}\texttt{1}, \texttt{?}\texttt{y} \}]_W^{\text{const}} \). By the induction hypothesis on the construction of property paths, we have that \( \mu \in [Q_r(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})]_W^0 \). Now, if \( \mu(\texttt{?}\texttt{x}) \notin \text{dom(adoc)} \), then we have that \( \mu(\texttt{?}\texttt{y}) = \mu(\texttt{?}\texttt{y}) \) and thus \( \mu \in [Q_e(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})]_W^0 \). If \( \mu(\texttt{?}\texttt{x}) \in \text{dom(adoc)} \), then it is easy to see that

\[
\mu \in \pi(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})([\{ \texttt{?}\texttt{x}, \texttt{r}\texttt{1}, \texttt{?}\texttt{y} \}]_W^{\text{const}} \times [\{ \texttt{?}\texttt{z}, \texttt{r}\texttt{1}, \texttt{?}\texttt{y} \}]_W^{\text{const}}).
\]

By induction hypothesis we have that

\[
\mu \in \pi(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})([Q_e(\texttt{?}\texttt{x}, \texttt{?}\texttt{z})]_W^0 \cup \\
\bigcup_{i=0}^{k-1} [\pi(\texttt{?}\texttt{x}, \texttt{?}\texttt{z})]((\texttt{SEED} \texttt{?}\texttt{x} ((\texttt{?}\texttt{v}, Q_e(\texttt{?}\texttt{v})))^1, (\texttt{GRAPH} ?u \{ \}))) \text{ AND } Q_r(\texttt{?}\texttt{u}, \texttt{?}\texttt{z})]_W^0)]_W^0 \times [Q_r(\texttt{?}\texttt{z}, \texttt{?}\texttt{y})]_W^0).
\]

Then we know that there exists \( j \) such that \( 0 \leq j \leq k - 1 \) such that

\[
\mu \in \pi(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})([Q_e(\texttt{?}\texttt{x}, \texttt{?}\texttt{z})]_W^0 \cup \\
[\pi(\texttt{?}\texttt{x}, \texttt{?}\texttt{z})]((\texttt{SEED} \texttt{?}\texttt{x} ((\texttt{?}\texttt{v}, Q_e(\texttt{?}\texttt{v})))^1, (\texttt{GRAPH} ?u \{ \}))) \text{ AND } Q_r(\texttt{?}\texttt{u}, \texttt{?}\texttt{z})]_W^0)]_W^0 \times [Q_r(\texttt{?}\texttt{z}, \texttt{?}\texttt{y})]_W^0).
\]

If \( \mu \in \pi(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})([Q_e(\texttt{?}\texttt{x}, \texttt{?}\texttt{z})]_W^0 \times [Q_r(\texttt{?}\texttt{z}, \texttt{?}\texttt{y})]_W^0) = [Q_r(\texttt{?}\texttt{x}, \texttt{?}\texttt{y})]_W^0 \) we can apply the same argument as in the base case. Now assume that
\[ \mu \in \pi(\{x, y\}) \]
\[ \pi(\{x, y\}) \left( (\text{SEED} \, ?x \, (?v, Q_s(?v)) \land (\text{GRAPH} \, ?u \{ \}) \land Q_{r_1}(?u, ?z)) \right) W \]
\[ \times [Q_{r_1}(?z, ?y)] W \].

Then we know that there exists a mapping \( \mu' \) and \( \mu'' \) such that
\[ \mu' \in \pi(\{x, y\}) \left( (\text{SEED} \, ?x \, (?v, Q_s(?v)) \land (\text{GRAPH} \, ?u \{ \}) \land Q_{r_1}(?u, ?z)) \right) W \]
and
\[ \mu'' \in [Q_{r_1}(?z, ?y)] W \]
\( \mu \) equals \( \mu' \cup \mu'' \) restricted to variables \( ?x, ?y \). Notice that \( \mu'' \) is compatible with \( \mu' \) thus, we have that \( \mu'(?z) = \mu''(?z) \). Now, if \( \mu''(?z) \notin \text{dom(adoc)} \), since \( \mu'' \in [Q_{r_1}(?z, ?y)] W \), then necessarily \( \mu''(?z) = \mu''(?y) \), and given that \( \mu' \) is compatible with \( \mu'' \) we obtain that \( \mu'(?z) = \mu''(?y) \). All this implies that
\[ \mu \in \pi(\{x, y\}) \left( (\text{SEED} \, ?x \, (?v, Q_s(?v)) \land (\text{GRAPH} \, ?u \{ \}) \land Q_{r_1}(?u, ?z)) \right) W \]
and thus (9) holds. Assume now that \( \mu''(?z) \in \text{dom(adoc)} \). We will prove that
\[ \mu' \in \left( (\text{SEED} \, ?x \, (?v, Q_s(?v)) \land (\text{GRAPH} \, ?u \{ \}) \land Q_{r_1}(?u, ?z)) \right) W \]
We know that
\[ \mu' \in \pi(\{x, y\}) \left( (\text{SEED} \, ?x \, (?v, Q_s(?v)) \land (\text{GRAPH} \, ?u \{ \}) \land Q_{r_1}(?u, ?z)) \right) W \]
Thus \( \mu' \) equals \( \mu_1 \cup \mu_2 \) (restricted to variables \( ?x, ?z \)) where
\[ \mu_1 \in \left( (\text{SEED} \, ?x \, (?v, Q_s(?v)) \land (\text{GRAPH} \, ?u \{ \}) \right) W \]
and
\[ \mu_2 \in [Q_{r_1}(?u, ?z)] W \]
Thus, regarding \( \mu_1 \) we know that there exists a sequence of URIs, \( u_1, u_2, \ldots, u_i \) such that \( \mu_1(?x) = u_1, \mu_1(?u) = u_i \) and \( u_{i+1} \in ([?v, Q_s(?v)] W \). Now, recall that the definition of \( Q_s(?v) \) is
\[ Q_s(?v) = (\langle \varepsilon, (\text{GRAPH} \, ?f \{ \}) \rangle \land Q_{r_1}(?f, ?v)) \].

Then essentially what we have is that
\[ (?f \rightarrow u_i, ?v \rightarrow u_{i+1}) \in [Q_{r_1}(?f, ?v)] W \].
Moreover, since $\mu_1$ and $\mu_2$ are compatible, we know that $\mu_1(\exists u) = \mu_2(\exists u) = u_j$ and since $\mu_2 \in [Q_{\gamma_1}(\exists u, ?z)]^W$ we know that

$$\{ ?f \rightarrow u_j, ?v \rightarrow \mu_2(?z) \} \in [Q_{\gamma_1}(\exists f, ?v)]^W.$$ 

Finally, given that we are assuming that $\mu''(\exists z) = \mu_2(?z)$ is in $\text{dom}(adoc)$ we have that

$$\{ ?x \rightarrow \mu_1(?x), ?z \rightarrow \mu_2(?z) \} \in [[\text{SEED} ?x ((?v, Q_\gamma(?v)))^j, (\text{GRAPH} ?z \{ \}))]]^W,$$

which is what we wanted to prove. Thus we have that

$$\mu' \in [[\text{SEED} ?x ((?v, Q_\gamma(?v)))^j, (\text{GRAPH} ?z \{ \}))]]^W,$$

and also that

$$\mu'' \in [Q_{\gamma_1}(\exists z, ?y)]^W$$

and given that $\mu$ equals $\mu' \cup \mu''$ restricted to variables $?x, ?y$, we have that

$$\mu \in \left[ \prod_{?x, ?y} \left( [[\text{SEED} ?x ((?v, Q_\gamma(?v)))^j, (\text{GRAPH} ?z \{ \}))]) \text{ AND } Q_{\gamma_1}(\exists z, ?y) \right] \right]^W$$

and since $j + 1 \leq k$ we obtain

$$\mu \in \left[ \prod_{?x, ?y} \left( [[\text{SEED} ?x ((?v, Q_\gamma(?v)))^j, (\text{GRAPH} ?z \{ \}))]) \text{ AND } Q_{\gamma_1}(\exists z, ?y) \right] \right]^W$$

and since $j + 1 \leq k$ we obtain

$$\mu \in \left[ \prod_{?x, ?y} \left( [[\text{SEED} ?x ((?v, Q_\gamma(?v)))^j, (\text{GRAPH} ?z \{ \}))]) \text{ AND } Q_{\gamma_1}(\exists z, ?y) \right] \right]^W$$

If one assumes that $\mu \in [[Q_{\gamma}(?x, ?y)]]^W$, then by an argument on exactly the same lines of the argument above, one can show that $\mu \in [[?x, ?, ?y]]^W$.

We have shown how to construct an equivalent LDQL query for every property path pattern with two variables. If the triple does not have two variables, we need a slightly different construction, in particular for the case in which $(\cdot)^{\star}$ is used. We now show the details of the construction but leave the complete proof as an exercise (it can be completed using the arguments of the previous part of this proof).

Consider a property path pattern $(\alpha, r, \beta)$ where $\alpha$ is a URI or variable, and $\beta$ is a URI, variable or literal. Then for the cases $r = p \in \mathcal{U}$, $r = \{u_1, \ldots, u_k\}$, $r = r_1/r_2$, $r = r_1\cdot r_2$, we construct a query as $Q_r(\alpha, \beta)$ where $Q_r(\alpha, \beta)$ is query $Q_r(?x, ?y)$ where all occurrences of $?x$ has been replaced by $\alpha$ and all occurrences of $?y$ has been replaced by $\beta$. For the case of $r = r_1^*$ we need to do a slightly different construction. For a pattern $(u, r, ?y)$ we construct a query $P_r(\exists y)$ as

$$\langle \varepsilon, \text{BIND}(u \text{ AS } ?y) \rangle \text{ UNION } ((\text{SEED} \{ u \}) ((?v, Q_\gamma(?v))^*, (\text{GRAPH} ?z \{ \})) \text{ AND } Q_{\gamma_1}(\exists z, ?y))$$

For a pattern $(?x, r, v)$ we construct a query $S_r^v(?x)$ as

$$\langle \varepsilon, \text{BIND}(u \text{ AS } ?x) \rangle \text{ UNION } ((\text{SEED} ?x ((?v, Q_\gamma(?v))^*, (\text{GRAPH} ?z \{ \})) \text{ AND } T(?z))$$
where $T(\text{?z})$ is either $Q_{r_1}(\text{?z}, v)$ or $S^{v}_{r_1}(\text{?z})$ depending on the form of $r_1$. Finally, for a pattern $(u, r, v)$ we construct a query $\hat{U}_r$ as

$$\langle \varepsilon, (\text{BIND}(u \text{ AS } \text{?x}) \text{ AND } \text{BIND}(v \text{ AS } \text{?y}) \text{ FILTER } (?x = ?y)) \text{ UNION } ((\text{SEED } \{ u \} ((?v, Q_{{v}()}^{*}, (\text{GRAPH } \text{?z } \{ \} ))) \text{ AND } T(\text{?z})))$$

where $T(\text{?z})$ is either $Q_{r_1}(\text{?z}, v)$ or $S^{v}_{r_1}(\text{?z})$ depending on the form of $r_1$.

Finally, consider a property path pattern $(\ell, r, \beta)$, where $\ell$ is a literal. Then for the cases $r = p \in \mathcal{U}$, $r = !(u_1 | \cdots | u_k)$ we should translate it into an unsatisfiable query. One way of obtaining that query is, for example, with an expression

$$\langle \varepsilon, (\text{BIND}(\ell \text{ AS } \text{?x}) \text{ AND } \text{BIND}(\ell \text{ AS } \text{?y}) \text{ FILTER } (?x \neq ?y))$$

For the cases $r = r_1/r_2$ and $r = r_1|r_2$ we follow the same construction as if $\ell$ were a URI but with the last base case. For the case of $r = r^*_1$, if $\beta$ is a variable $y$ we consider the following query

$$\langle \varepsilon, \text{BIND}(\ell \text{ AS } \text{?y})\rangle.$$ and if $\beta$ is a URI or literal the query

$$\langle \varepsilon, (\text{BIND}(\ell \text{ AS } ?x) \text{ AND } \text{BIND}(\beta \text{ AS } ?y) \text{ FILTER } (?x = ?y))$$

The correctness of this translation can be proved along the same lines as for the case of property path pattern $(?x, r, ?y)$.

### A.6 Proof of Theorem 3

We proceed by induction showing how to translate every possible NautiLOD query. The translation works in two parts. We first define the following function $\text{trans}_N(\cdot)$ that given a NautiLOD query, produces an LPE.

$$\text{trans}_N(p) = \langle +, p, \_ \rangle$$
$$\text{trans}_N(p^*) = \langle _, p, + \rangle$$
$$\text{trans}_N((\_)) = \langle ?x, \varepsilon, (\text{GRAPH ?u (?u, ?p, ?x)))\rangle$$
$$\text{trans}_N(n_1/n_2) = \text{trans}_N(n_1)/\text{trans}_N(n_2)$$
$$\text{trans}_N(n_1|n_2) = \text{trans}_N(n_1)|\text{trans}_N(n_2)$$
$$\text{trans}_N(n^*) = \text{trans}_N(n)^*$$
$$\text{trans}_N([\text{ASK } P]) = \text{trans}_N(n)/[\langle ?x, \varepsilon, (\text{GRAPH ?x } P)\rangle]$$

Before presenting the complete translations, we prove the following result. Let $n$ be a NautiLOD expression, then for every Web of Linked Data and URIs $u, v \in \text{dom}(\text{adoc})$ we have that

$$v \in [n]^u_W \text{ if and only if } v \in [\text{trans}_N(n)]^u_W.$$ The proof is by induction in the construction of the NautiLOD expression.
- for the case of $p \in \mathcal{U}$ we have that

$$[p]^u W = \{ u' \mid (u, p, u') \in \text{data}(\text{adoc}(u)) \}$$

notice that $v \in \text{dom}(\text{adoc})$ and $v \in [p]^u W$, if and only if there is a link from document $\text{adoc}(u)$ to document $\text{adoc}(v)$ that matches $(+, p, _)$. This happens, if and only if $v \in [(+, p, _)]^u W$, which is what we wanted to prove.

- the case for $p'$ is similar but using $(+, p, _)$. 

- the case for $(\_)$, Just notice that $v \in [[(\_)]^n W]$ if and only if there exists a $p \in \mathcal{U}$ such that $(u, p, v) \in \text{data}(\text{adoc}(u))$. On the other hand we have that $v \in [\pi_{\text{trans}}(\text{graph} ? u (?u, ?p, ?x)))]^u W$ if and only if $v \in [\pi_{\text{trans}}(\text{graph} ? u (?u, ?p, ?x))]^u W$. This happens if and only if there exists $p$ such that $(u, p, v) \in \text{data}(\text{adoc}(u))$. This proves the desired property.

- the case of an expression $n_1/n_2$, we have that $v \in \text{dom}(\text{adoc})$ is in $[n_1/n_2]^u W$ if and only if, there exists $v' \in \text{dom}(\text{adoc})$ such that $v' \in [n_1]^u W$ and $v \in [n_2]^u W$. The we can apply or induction hypothesis and we have that $v \in [n_1/n_2]^u W$ if and only if $v' \in [\text{trans}_N(n_1)]^u W$ and $v \in [\text{trans}_N(n_2)]^u W$, and thus $v \in [\text{trans}_N(n_1/n_2)]^u W$.

- cases $n_1n_2$ and $n^*$ are direct from the definition of NautilOD and LDQL.

- for the case of expression $n[(\text{ASK} P)]$ we have that $v \in [n[(\text{ASK} P)]]^u W$ if and only if $v \in [n]^u W$, $v \in \text{dom}(\text{adoc})$ and $[P]_{\text{data}(\text{adoc}(v))} \neq \emptyset$. On the other hand, we have that $v \in [\text{trans}_N(n[(\text{ASK} P)])]^u W$ if and only if

$$v \in [\pi_{\text{trans}}(\text{graph} ? x P)]^u W.$$ 

This happens if and only if there exists a $v'$ such that $v' \in [\text{trans}_N(n)]^u W$ and $v \in [[(?x, \epsilon, \text{graph} ? x P)]^u W]$. From the last property and the semantics of $[\_]$ in LDQL, we have that $v = v'$ and that $[(?x, \epsilon, \text{graph} ? x P)]^u W \neq \emptyset$. The last holds if and only if $[\pi_{\text{trans}}(\text{graph} ? x P)]^u W \neq \emptyset$, with $W$ the RDF dataset \{\text{data}(\text{adoc}(v)), \{v, \text{data}(\text{adoc}(v))\}\}. Thus we have that $v \in [\text{trans}_N(n[(\text{ASK} P)])]^u W$ if and only if $v \in [\text{trans}_N(n)]^u W$ and $[P]_{\text{data}(\text{adoc}(u))} \neq \emptyset$. Applying our induction hypothesis we have $v \in [n]^u W$ and $[P]_{\text{data}(\text{adoc}(u))} \neq \emptyset$, which is exactly what we needed to prove.

Notice that the hypothesis that $v \in \text{dom}(\text{adoc})$ was fundamental to prove the previous result. Nevertheless, the output of a NautilOD query can be a URI not in $\text{dom}(\text{adoc})$ or even a literal, so we need to do a different translation in general. Thus, we use now $\text{trans}_N(\cdot)$ to translate a general NautilOD query. Given a NautilOD expression $n$ we have two cases. Assume first that $n$, as a regular expression, does not produce the empty string $\epsilon$. Then, by using regular language results, we know that we can write an equivalent expression $n'$ of the form

$$n_1/e_1 \mid \cdots \mid n_k/e_k \mid m_1[(\text{ASK} P_1)] \mid \cdots \mid m_t[(\text{ASK} P_t)]$$


where every $n_i$ and $m_j$ is a NautiLOD query, and every $e_i$ is either of the form $p$, or $p^*$, or $\langle \_ \rangle$. We are ready now to produce an LDQL query $Q_n(\?x)$ which is equivalent to $n$. The query is constructed as follows.

$$Q_n(\?x) = \pi_{\?x}\left( (\text{trans}_N(n_1), Q_1) \text{ UNION } \cdots \text{ UNION } (\text{trans}_N(n_k), Q_k) \text{ UNION } (\text{trans}_N(m_1), (\text{GRAPH} ?x P_1)) \text{ UNION } \cdots \text{ UNION } (\text{trans}_N(m_\ell), (\text{GRAPH} ?x P_\ell)) \right)$$

where query $Q_i$ depends on the form of $e_i$:

- if $e_i = p$ then $Q_i = (\text{GRAPH} ?u (\?u, p, \?x))$
- if $e_i = p^*$ then $Q_i = (\text{GRAPH} ?u (\?x, p, \?u))$
- if $e_i = \langle \_ \rangle$ then $Q_i = (\text{GRAPH} ?u (\?u, p, \?x))$

Now to prove the correctness of our construction, assume that $v \in [n]_W^n$. Then we know that $v \in [n_i/e_i]_W^n$ or $v \in [m_i/(\text{ASK} P_i)]_W^n$ for some $i$. If $v \in [n_i/e_i]_W^n$ we know that there exists a $v'$ such that $v' \in [n_i]_W^n$ and $v \in [e_i]_W^n$. Notice that, since $v \in [e_i]_W^n$, and $e_i$ is either $p$, or $p^*$, or $\langle \_ \rangle$ then we know that $v'$ is in dom(adoc). Thus we can apply our previous result to conclude from $v' \in [n_i]_W^n$ that $v' \in [\text{trans}_N(n_i)]_W^n[u]$. Now if $e_i = p$ then from $v \in [e_i]_W^n$ we conclude that $(v', p, v) \in \text{data}(\text{adoc}(v'))$ and thus $[\text{(GRAPH} ?u (\?u, p, \?x))]_D^\mathcal{P}_\mathcal{D}$ has $\mu$ as solution, with $D = \{\text{data}(\text{adoc}(v')), \langle v', \text{data}(\text{adoc}(v')) \rangle\}$. Given that $v' \in [\text{trans}_N(n_i)]_W^n[u]$, we have that

$$\mu = \{?u \rightarrow v', ?x \rightarrow v\} \in [\text{trans}_N(n_i), Q_i]_W^n[u]$$

Finally, given that $Q_n(\?x)$ only keep the $\?x$ variable, we have that $\{?x \rightarrow v\}$ is in $[Q_n(\?x)]_W^n[u]$, which is what we wanted to show. If $e_i = p^*$ or $e_i = \langle \_ \rangle$ the proof is the essentially the same.

Now assume that $v \in [m_i/(\text{ASK} P_i)]_W^n$. This implies that $v$ is in $[m_i]_W^n$ and that $[P_i]_\text{data}(\text{adoc}(v)) \neq \emptyset$. By the semantics of NautiLOD, we have that $v$ is in dom(\text{adoc}) (otherwise we could not have been able to evaluate $P$), and thus we can apply our result above to obtain that $v \in [\text{trans}_N(m_i)]_W^n[u]$. Now, given that $[P_i]_\text{data}(\text{adoc}(v)) \neq \emptyset$ we have that $[\text{(GRAPH} ?x P_i)]_D^\mathcal{P}_\mathcal{D} \neq \emptyset$ where $D = \{\text{data}(\text{adoc}(v)), \langle v, \text{data}(\text{adoc}(v)) \rangle\}$. Moreover, we have that every mapping $\mu$ in $[\text{(GRAPH} ?x P_i)]_D^\mathcal{P}_\mathcal{D}$ is such that $\mu(?x) = v$. All these facts implies that mapping $\mu' = \{?x \rightarrow v\}$ is in $[\text{(trans}_N(m_i), (\text{GRAPH} ?x P_i)]_W^n[u]$, and thus $\mu'$ is in $[Q_n(\?x)]_W^n[u]$ which is exactly what we wanted to prove.

If we start by assuming that $\mu = \{?x \rightarrow v\}$ is in $[Q_n(\?x)]_W^n[u]$, then following a similar reasoning as above one concludes that $v \in [n]_W^n$.

To complete the proof we have to cover the case in which $n$, as a regular expression, can produce the empty string. Then, by applying some classical regular languages properties, one can rewrite $n$ as $\varepsilon n'$ with $n'$ an expression that does not produce the empty string $\varepsilon$. Thus we can translate $n$ into the LDQL query

$$\langle \varepsilon, (\text{GRAPH} ?x \{ \} \rangle \text{ UNION } Q_n(\?x)$$
Notice that for every \( u \in \text{data}(\text{adoc}(v)) \) we have that \( \llbracket \langle \varepsilon, (\text{GRAPH} \ ?x \ \{ \} \rangle \rangle \rrbracket^u_W \) results in a single mapping \( \mu = \{ ?x \rightarrow u \} \).

**A.7 Proof of Theorem 4**

Recall that NautiLOD can only express paths and no combination of those paths via SPARQL operators is allowed. Thus, it is easy to prove that NautiLOD cannot express operators such as \text{SEED}, \text{AND}, \text{UNION} that are natively allowed in LDQL. Thus to make a stronger claim, we will prove that there exists simple LDQL query not using the mentioned operators, that cannot be expressed using NautiLOD. The proof is similar to the proof of Theorem 1.

Thus, consider the LDQL \( Q(\langle ?x, ?x, ?x \rangle) \) query given by

\[
Q(\langle +, p, _ \rangle, \langle ?x, ?x, ?x \rangle)
\]

with \( p \in \mathcal{U} \). Now assume that there exists a NautiLOD expression \( n \) such that

\[
\llbracket n \rrbracket^u_W = \llbracket Q(\langle ?x \rangle) \rrbracket^u_W
\]

for every Web of Linked Data \( W \) and \( v \in \text{dom}(\text{adoc}) \). Let \( u, u', a, b \) be different elements in \( \mathcal{U} \) that are not mentioned in \( n \). Consider now \( W_1 \) having only two documents \( d_1 = \{(u, p, u')\} \) and \( d_2 = \{(a, a, a)\} \) and such that \( \text{adoc}(u) = d_1 \) and \( \text{adoc}(u') = d_2 \). Moreover, consider \( W_2 \) having also two documents \( d_1 = \{(u, p, u')\} \) and \( d_3 = \{(b, b, b)\} \) such that \( \text{adoc}(u) = d_1 \) and \( \text{adoc}(u') = d_3 \). First notice that

\[
\llbracket Q(\langle ?x \rangle) \rrbracket^u_W = \{ ?x \rightarrow a \} \neq \llbracket Q(\langle ?x \rangle) \rrbracket^u_W = \{ ?x \rightarrow b \}
\]

We now prove that \( \llbracket n \rrbracket^u_W = \llbracket n \rrbracket^u_W \) which is a contradiction. To prove this, we show that for every subexpression \( e \) of \( n \), and for every possible URI \( v \), it holds that \( \llbracket e \rrbracket^u_W = \llbracket e \rrbracket^u_W \). First notice that \( W_1 \) and \( W_2 \) has only two URIs in \( \text{dom}(\text{adoc}) \), namely, \( u \) and \( u' \), thus, we only have to reason for the cases in which \( v = u \) or \( v = u' \). We proceed by induction.

- Assume that \( e = r \in \mathcal{U} \). Given that in \( W_1 \) and \( W_2 \) the URI \( u \) is associated with the same document (document \( d_1 \)), then \( \llbracket r \rrbracket^u_{W_1} = \llbracket r \rrbracket^u_{W_2} \). Moreover, given that \( r \neq a \) and \( r \neq b \) (recall that \( n \) does not mention \( a \) or \( b \)), we have that \( \llbracket r \rrbracket^u_{W_1} = \llbracket r \rrbracket^u_{W_2} = \emptyset \).
- Assume that \( e = r^* \) with \( r \in \mathcal{U} \). Exactly the same argument as the above case applies.
- Assume that \( e = \langle \_ \rangle \). For the same reason as in the above two cases we have that \( \llbracket r \rrbracket^u_{W_1} = \llbracket r \rrbracket^u_{W_2} \). Now consider \( \llbracket \_ \rrbracket^u_{W_1} \). Then we have that URI \( v \) is in \( \llbracket \_ \rrbracket^u_{W_1} \) if and only if, there exists some \( p \) such that \( \langle u', p, v \rangle \in \text{data}(\text{adoc}(u')) \), but the only triple in \( \text{data}(\text{adoc}(u')) \) is \( \langle a, a, a \rangle \) and since \( a \neq u' \) we have that \( \llbracket \_ \rrbracket^u_{W_1} = \emptyset \). For a similar reason we obtain that \( \llbracket \_ \rrbracket^u_{W_2} = \emptyset \), completing this part of the proof.
- The cases \( e = r_1/r_2, e = r_1|r_2 \) and \( e = r^* \) follows from the base cases proved above.
Assume $e = r[(\text{ASK } P)]$. By definition we have that

$$[r[(\text{ASK } P)]]_W = \{ v' \mid v' \in [r]_W^v, v' \in \text{dom}(\text{adoc}) \text{ and } [P]_{\text{data}(\text{adoc}(v'))} \neq \emptyset \}$$

By induction hypothesis we have that $[r]_{W_1}^v = [r]_{W_2}^v$ for $v = u, u'$. Thus we only need to prove that the evaluation of $P$ is always the same. given that $\text{data}(\text{adoc}(u))$ is the same document in $W_1$ and $W_2$, we have that for $u$ the property holds. Now consider $[P]_{d_2}$ and $[P]_{d_3}$ with $d_1 = \{(a, a, a)\}$ and $d_2 = \{(b, b, b)\}$. Recall that $P$ does not mention $a$ or $b$, thus we have that if $\mu \in [P]_{d_2}$ then the mapping $\mu'$ obtained from $\mu$ by replacing every occurrence of $a$ by $b$, is in $[P]_{d_3}$, and vice versa. Thus we have that $[P]_{d_2} = \emptyset$ if and only if $[P]_{d_3} = \emptyset$. This proves that $[r[(\text{ASK } P)]]_{W_1}^v = [r[(\text{ASK } P)]]_{W_2}^v$ for $v = u, u'$.

We have finished the proof that $[n]_{W_1}^u = [n]_{W_2}^u$ thus contradicting the fact that $n$ is equivalent to $Q(x)$.

### A.8 Proof of Theorem 5

Let $P$ be an arbitrary SPARQL graph pattern, let $W = (D, \text{adoc})$ be an arbitrary Web of Linked Data, and let $S$ be some finite set of URIs. To prove the theorem we use the (basic) LDQL queries $\langle \text{ipe}_\text{cAll}, P \rangle$, $\langle \text{ipe}_\text{cNone}, P \rangle$, and $\langle \text{ipe}_\text{cMatch}, P \rangle$, with the following LPEs:

- $\text{ipe}_\text{cAll}$ is $\langle -, -, - \rangle^*$,
- $\text{ipe}_\text{cNone}$ is $\varepsilon$, and
- $\text{ipe}_\text{cMatch}$ is $\langle \langle s, q_1 \rangle | \langle p, q_1 \rangle | \langle o, q_1 \rangle | \ldots | \langle s, q_m \rangle | \langle p, q_m \rangle | \langle o, q_m \rangle \rangle^*$ where $s, p$ and $o$ are fresh variables (not used in $P$), $m$ is the number of triple patterns in $P$, and for each such triple pattern $t_{pk} (1 \leq k \leq m)$ there exists a subquery $q_k$ of the form $\langle \varepsilon, P_k \rangle$ with a SPARQL pattern $P_k$ that is constructed as follows: $P_k$ contains the triple pattern $\langle s, p, o \rangle$ and—depending on the form of the corresponding triple pattern $t_{pk} = \langle s_k, p_k, o_k \rangle$—may contain additional $\text{FILTER}$ operators; in particular, if $s_k \notin V$, then $P_k$ contains $\text{FILTER} s = s_k$; if $p_k \notin V$, then $P_k$ contains $\text{FILTER} p = p_k$; and if $o_k \notin V$, then $P_k$ contains $\text{FILTER} o = o_k$.

Then, for each reachability criterion $c \in \{ \text{cAll, cNone, cMatch} \}$ with its corresponding LPE $\text{ipe}_c$ as specified above, we have to show the following equivalence:

$$[P]^{R(c,S)}_W = [\langle \text{ipe}_c, P \rangle]_W^S. \quad (10)$$

By the definition of the reachability-based query semantics (cf. Section 4.3) and the definition of LDQL query semantics (cf. Definition 5), it is sufficient to prove the following lemma to show that (10) holds for each $c \in \{ \text{cAll, cNone, cMatch} \}$.

**Lemma 4.** For each $c \in \{ \text{cAll, cNone, cMatch} \}$, the set of all documents that are $(c, S, P)$-reachable in $W$ is equivalent to the following set of documents:

$$D^c_{\text{LPE}} = \{ \text{adoc}(u) \mid u \in \langle \text{ipe}_c \rangle_{W^u} \text{ for some } u_{\text{ctx}} \in S \}.$$
Notice that for each \( c \in \{ c_{\text{All}}, c_{\text{None}}, c_{\text{Match}} \} \), the set \( D^c_{\text{LPE}} \) is the set of documents selected by evaluating \( lpe^c \) over \( W \) using every URI in \( S \) as context URI. In the following, we prove Lemma 4 for each of the three reachability criteria, \( c_{\text{All}}, c_{\text{None}}, \) and \( c_{\text{Match}} \).

**c_{\text{All}}-semantics:** To prove Lemma 4 for \( c_{\text{All}} \) we show that the set \( D^c_{\text{LPE}} \) is both a subset and a superset of the set of all \( (c_{\text{All}}, S, P) \)-reachable documents in \( W \).

We begin with the former. Hence, for an arbitrary document in \( D^c_{\text{LPE}} \) we have to show that this document is \( (c_{\text{All}}, S, P) \)-reachable in \( W \). Let \( d_{\text{LPE}} \in D^c_{\text{LPE}} \) be such a document. Since \( d_{\text{LPE}} \in D^c_{\text{LPE}} \), we know that there exist two URIs, \( u_{\text{ctx}} \) and \( u \), such that

- \( u_{\text{ctx}} \in S \),
- \( u \in \llbracket lpe^{c_{\text{All}}} \rrbracket^u_{\text{ctx}} \), and
- \( d_{\text{LPE}} = adoc(u) \).

Then, either we have \( u_{\text{ctx}} = u \) or \( u_{\text{ctx}} \neq u \). In the following, we discuss these two cases.

If \( u_{\text{ctx}} = u \), then \( d_{\text{LPE}} = adoc(u_{\text{ctx}}) \) and, thus, document \( d_{\text{LPE}} \) is \( (c_{\text{All}}, S, P) \)-reachable in \( W \) because it satisfies the first of the two alternative conditions for reachability as given in Section 4.3.

If \( u_{\text{ctx}} \neq u \), then, given that \( u \in \llbracket lpe^{c_{\text{All}}} \rrbracket^u_{\text{ctx}} \), there exists a nonempty sequence of link graph edges

\[
\langle d_1, (t_1, u_1), d'_1 \rangle \in G_W, \quad \langle d_2, (t_2, u_2), d'_2 \rangle \in G_W, \quad \ldots, \quad \langle d_n, (t_n, u_n), d'_n \rangle \in G_W
\]

such that

- \( d_1 = adoc(u_{\text{ctx}}) \),
- \( d'_i = d_{i+1} \) for all \( i \in \{1, \ldots, n-1\} \), and
- \( d'_n = d_{\text{LPE}} \) (and \( u_n = u \)).

Then, since \( d_1 = adoc(u_{\text{ctx}}) \) and \( u_{\text{ctx}} \in S \), we have that document \( d_1 \) is \( (c_{\text{All}}, S, P) \)-reachable in \( W \) (the document satisfies the first of the two conditions for reachability as given in Section 4.3). As a consequence, we can use the fact that \( d'_i = d_{i+1} \) for all \( i \in \{1, \ldots, n-1\} \) to show that all other documents connected by the sequence of link graph edges are also \( (c_{\text{All}}, S, P) \)-reachable in \( W \) (they satisfy the second condition). Therefore, due to \( d'_n = d_{\text{LPE}} \), document \( d_{\text{LPE}} \) is \( (c_{\text{All}}, S, P) \)-reachable in \( W \).

After showing that in both cases, \( u_{\text{ctx}} = u \) and \( u_{\text{ctx}} \neq u \), document \( d_{\text{LPE}} \in D^c_{\text{LPE}} \) is \( (c_{\text{All}}, S, P) \)-reachable in \( W \), we conclude that the set \( D^c_{\text{LPE}} \) is a subset of the set of all \( (c_{\text{All}}, S, P) \)-reachable documents in \( W \). It remains to show that \( D^c_{\text{LPE}} \) is also a superset.

To this end, let \( d_R \) be a document that is \( (c_{\text{All}}, S, P) \)-reachable in \( W \). We have to show that \( d_R \) is in \( D^c_{\text{LPE}} \). We note that document \( d_R \) may be \( (c_{\text{All}}, S, P) \)-reachable in \( W \) because it satisfies either the first or the second of the two alternative conditions for reachability as given in Section 4.3. In the following, we discuss both cases.

If \( d_R \) satisfies the first condition, there exists a URI \( u_R \in S \) such that \( adoc(u_R) = d_R \). Since \( lpe^{c_{\text{All}}} = (\text{-}, \text{-}, \text{-})^* \), we also have \( u_R \in \llbracket lpe^{c_{\text{All}}} \rrbracket^u_{\text{LPE}} \). Therefore, we can use URI \( u_R \) as both \( u_{\text{ctx}} \) and \( u \) in the definition of \( D^c_{\text{LPE}} \), which shows that \( d_R \in D^c_{\text{LPE}} \).
If \( d_R \) satisfies the second condition, then there exist both a seed URI \( u_0 \in S \) and a nonempty sequence of link graph edges

\[
\langle d_1, (t_1, u_1), d'_1 \rangle \in G_W, \quad \langle d_2, (t_2, u_2), d'_2 \rangle \in G_W, \quad \ldots, \quad \langle d_n, (t_n, u_n), d'_n \rangle \in G_W
\]

such that

- \( d_1 = adoc(u_0) \),
- \( d'_i = e_i + 1 \) for all \( i \in \{1, \ldots, n - 1\} \), and
- \( d'_n = d_R \) and, thus, \( d_R = adoc(u_n) \).

Moreover, every such link graph edge \( \langle d_j, (t_j, u_j), d'_j \rangle \) matches link pattern \( \langle \ast, \ast, \ast \rangle \) in the context of URI \( u_{j-1} \) \((1 \leq j \leq n)\). Therefore, since \( lpe_{\text{All}} \) is \( \langle \ast, \ast, \ast \rangle^n \), we have \( u_n \in \{lpe_{\text{All}}\}_{W}^{u_n} \). Then, with \( d_R = adoc(u_n) \) and \( u_0 \in S \), we can use \( u_n \) as \( u \) and \( u_0 \) as \( u_{ctx} \) in the definition of \( D_{LPE}^{\text{All}} \), which shows that \( d_R \in D_{LPE}^{\text{All}} \).

In conclusion, independent of whether \( d_R \) satisfies the first or the second condition for being \( (c_{\text{All}}, S, P) \)-reachable in \( W \), we find that \( d_R \in D_{LPE}^{\text{All}} \). Hence, \( D_{LPE}^{\text{All}} \) is not only a subset of all \( (c_{\text{All}}, S, P) \)-reachable documents in \( W \), but also a superset thereof, which shows that both sets are equivalent (as claimed in Lemma 4).

**\( c_{\text{None}} \)-semantics:** To prove Lemma 4 for \( c_{\text{None}} \) we show that the set \( D_{LPE}^{\text{None}} \) is both a subset and a superset of the set of all \( (c_{\text{None}}, S, P) \)-reachable documents in \( W \).

To begin with the former, assume an arbitrary document in \( d_{LPE} \in D_{LPE}^{\text{None}} \). We have to show that this document is \( (c_{\text{None}}, S, P) \)-reachable in \( W \). Since \( d_{LPE} \in D_{LPE}^{\text{All}} \), we know that there exist two URIs, \( u_{ctx} \) and \( u \), such that

- \( u_{ctx} \in S \),
- \( u \in \{lpe_{\text{None}}\}_{W}^{u_{ctx}} \), and
- \( d_{LPE} = adoc(u) \).

Given that \( lpe_{\text{None}} \) is \( \varepsilon \), by \( u \in \{lpe_{\text{None}}\}_{W}^{u_{ctx}} \) and Definition 5, we obtain that \( u = u_{ctx} \), and, thus, \( d_{LPE} = adoc(u_{ctx}) \). Therefore, document \( d_{LPE} \) is \( (c_{\text{None}}, S, P) \)-reachable in \( W \) because it satisfies the first of the two alternative conditions for reachability as given in Section 4.3. As a consequence, we can conclude that the set \( D_{LPE}^{\text{None}} \) is a subset of the set of all \( (c_{\text{None}}, S, P) \)-reachable documents in \( W \).

To show that \( D_{LPE}^{\text{None}} \) is also a superset, let \( d_R \) be an arbitrary document that is \( (c_{\text{None}}, S, P) \)-reachable in \( W \). We have to show that \( d_R \) is in \( D_{LPE}^{\text{None}} \). We note that \( d_R \) can be \( (c_{\text{None}}, S, P) \)-reachable in \( W \) only if it satisfies the first of the two alternative conditions for reachability as given in Section 4.3 for \( c_{\text{None}} \), the second condition cannot be satisfied by any document because \( c_{\text{None}}(t, u, P) = \text{false} \) for all \( (t, u, P) \in T \times U \times P \).

Therefore, given that \( d_R \) satisfies the first condition, there exists a URI \( u_R \in S \) such that \( adoc(u_R) = d_R \). Since \( lpe_{\text{None}} \) is \( \varepsilon \), we also have \( u_R \in \{lpe_{\text{None}}\}_{W}^{u} \). Therefore, we can use URI \( u_R \) as both \( u_{ctx} \) and \( u \) in the definition of \( D_{LPE}^{\text{All}} \), and, thus, obtain that \( d_R \in D_{LPE}^{\text{None}} \), which shows that the set \( D_{LPE}^{\text{None}} \) is a superset of the set of all \( (c_{\text{None}}, S, P) \)-reachable documents in \( W \). Since we have shown before that \( D_{LPE}^{\text{None}} \) is also a subset of the set of all documents that are \( (c_{\text{None}}, S, P) \)-reachable in \( W \), we conclude that both sets are equivalent. Hence, Lemma 4 holds for reachability criterion \( c_{\text{None}} \).
**CMatch-semantics:** It remains to prove Lemma 4 for CMatch. To this end, we show that the set $D_{\text{CMatch}}$ is both a subset and a superset of the set of all documents that are (CMatch, $S$, $P$)-reachable in $W$. As before, we begin with the former.

Let $d_{\text{LPE}}$ be an arbitrary document in $D_{\text{CMatch}}$. We have to show that this document is (CMatch, $S$, $P$)-reachable in $W$. Since $d_{\text{LPE}} \in D_{\text{CMatch}}$, we know by the definition of $D_{\text{CMatch}}$ (as given in Lemma 4) that there exist two URIs, $u_{\text{ctx}}$ and $u$, such that

- $u_{\text{ctx}} \in S$,
- $u \in [lpe_{\text{CMatch}}]_{W}$,
- $d_{\text{LPE}} = \text{adoc}(u)$.

Given that $u \in [lpe_{\text{CMatch}}]_{W}$, there exists a nonempty sequence of URIs $u_0, u_1, \ldots, u_n$ and a corresponding sequence of documents $d_0, d_1, \ldots, d_n$ such that

- $d_i = \text{adoc}(u_i)$ for each $i \in \{0, \ldots, n\}$,
- $u_0 = u_{\text{ctx}}$,
- $u_n = u$ (and, thus, $d_n = d_{\text{LPE}}$), and
- for each $i \in \{1, \ldots, n\}$, there exists a triple pattern $t p_k$ in $P$ ($1 \leq k \leq n$) such that $u_i \in \{s, t, u\}_{W}^{u_{i-1}}$ where $s \in \{s, t, p, o\}$ and $q_k$ is the LDQL query that corresponds to $t p_k$ as specified in the definition of $lpe_{\text{CMatch}}$ above.

We show by induction over $n$ that all $n+1$ documents, $d_0, d_1, \ldots, d_n$, are (CMatch, $S$, $P$)-reachable in $W$, and, thus, so is $d_{\text{LPE}} = d_n$.

**Base case ($n = 0$):** $d_0$ is (CMatch, $S$, $P$)-reachable in $W$ because $d_0 = \text{adoc}(u_0)$, $u_0 = u_{\text{ctx}}$, and $u_{\text{ctx}} \in S$; i.e., $d_0$ satisfies the first condition as specified in Section 4.3.

**Induction step ($n > 0$):** By induction, we assume that document $d_{n-1}$ is (CMatch, $S$, $P$)-reachable in $W$. To show that $d_n$ is also (CMatch, $S$, $P$)-reachable in $W$ we aim to show that $d_n$ satisfies the second condition for reachability as given in Section 4.3. That is, we aim to show that there exists a link graph edge $(d_{n-1}, (t, u), d_t) \in G_W$ such that (i) $d_{n-1}$ is (CMatch, $S$, $P$)-reachable in $W$, (ii) $\text{CMatch}(t, u, P) = \text{true}$, (iii) $u = u_n$, and (iv) $d_t = d_n$. Let $d_{\text{ctx}}$ be $d_{n-1}$, which is (CMatch, $S$, $P$)-reachable in $W$ by our inductive hypothesis. Hence, it remains to show the existence of a link graph edge $(d_{n-1}, (t, u_n), d_n) \in G_W$ for which $\text{CMatch}(t, u_n, P) = \text{true}$. To this end, we use the fourth of the four aforementioned properties of the sequence of URIs $u_0, u_1, \ldots, u_n$.

Let $t p_k = \langle s, p, o \rangle$ be a triple pattern in $P$ such that $u_n \in \{s, t, p, o\}_{W}^{u_{n-1}}$ where $s \in \{s, t, p, o\}$ and $q_k$ is the LDQL query that corresponds to $t p_k$ as specified in the definition of $lpe_{\text{CMatch}}$ above; i.e., $q_k$ is a basic LDQL query of the form $(\varepsilon, P_k)$ where SPARQL pattern $P_k$ contains the triple pattern $(s, p, o)$ and (i) if $s_k \notin V$, then $P_k$ contains $\text{FILTER} s = s_k$, (ii) if $p_k \notin V$, then $P_k$ contains $\text{FILTER} p = p_k$, and (iii) if $o_k \notin V$, then $P_k$ contains $\text{FILTER} o = o_k$.

Since $u_n \in \{s, t, p, o\}_{W}^{u_{n-1}}$, by Definition 5, there exists a solution mapping $\mu$ such that $\mu \in \{q_k\}_{W}^{u_{n-1}}$ and $\mu(s) = u_n$. Moreover, since $q_k$ is of the form $(\varepsilon, P_k)$, we have $\mu \in [P_k]_{\varepsilon}$, where $\Sigma = \text{dataset}_W \{u_{n-1}\}$ with default graph $G = \text{data}(d_{n-1})$. Then, due to the construction of $P_k$, it is easily verified that there exists an RDF triple $t \in \text{data}(d_{n-1})$ such that $\mu(t p_k) = t$ and $u_n \in \text{uris}(t)$. As a consequence, (i) $\text{CMatch}(t, u_n, t p_k) = \text{true}$ and (ii) by Definition 2, there exists a link graph edge
that there exists a triple
\(\langle d_{n-1}, (t, u_n), d_n \rangle \in \mathcal{G}_W\). Finally, since \(tp_k\) is a triple pattern in \(P\), we also have \(c_{\text{Match}}(t, u_n, P) = \text{true}\).

While this concludes showing that the set \(D_{LPE}^{\text{Match}}\) is a subset of the set of all documents that are \((c_{\text{Match}}, S, P)\)-reachable in \(W\), we now show that it is also a superset thereof.

Let \(d_R\) be a document that is \((c_{\text{Match}}, S, P)\)-reachable in \(W\). We have to show that \(d_R\) is in \(D_{LPE}^{\text{Match}}\). Since \(d_R\) is \((c_{\text{Match}}, S, P)\)-reachable in \(W\), there exist a nonempty sequence of URIs \(u_0, u_1, \ldots, u_n\), a corresponding sequence of documents
\[d_0 = \text{adoc}(u_0), \; d_1 = \text{adoc}(u_1), \; d_2 = \text{adoc}(u_2), \; \ldots, \; d_n = \text{adoc}(u_n),\]
and a corresponding sequence of link graph edges
\[\langle d'_1, (t_1, u_1), d_1 \rangle \in \mathcal{G}_W, \; \langle d'_2, (t_2, u_2), d_2 \rangle \in \mathcal{G}_W, \; \ldots, \; \langle d'_n, (t_n, u_n), d_n \rangle \in \mathcal{G}_W\]
such that
- \(u_0 \in S\),
- \(c_{\text{Match}}(t_i, u_i, P) = \text{true}\) for all \(i \in \{1, \ldots, n\}\),
- \(d'_i = d_{i-1}\) for all \(i \in \{1, \ldots, n\}\), and
- \(d_n = d_R\) and, thus, \(d_R = \text{adoc}(u_n)\).

We aim to show that each of the \(n + 1\) documents, \(d_0, d_1, d_2, \ldots, d_n\), is in \(D_{LPE}^{\text{Match}}\), and, thus, so is \(d_R = d_n\). To this end, it is sufficient to show that each of the \(n + 1\) URIs, \(u_0, u_1, \ldots, u_n\), is in \(\text{LPE}^{\text{Match}}W\). Then, with \(u_0 \in S\), for each \(i \in \{0, \ldots, n\}\) we can use URI \(u_i\) as \(u\) and \(u_0\) as \(u_{\text{ctx}}\) in the definition of \(D_{LPE}^{\text{Match}}\), which shows that document \(d_i = \text{adoc}(u_i)\) is in \(D_{LPE}^{\text{Match}}\). We use proof by induction.

**Base case \((n = 0)\):** Since \(\text{LPE}^{\text{Match}}\) is of the form \((\cdot)^*\), we have \(u_0 \in \text{LPE}^{\text{Match}}W\).

**Induction step \((n > 0)\):** By induction, we assume that \(u_{n-1} \in \text{LPE}^{\text{Match}}W\). Then, to show that \(u_n\) is also in \(\text{LPE}^{\text{Match}}W\), it is sufficient to show that \(u_n\) is in \(\text{LPE}^{\text{Match}}W\) where \(\text{LPE}^{\text{Match}}\) is \((\cdot)^*\). Due to the existence of link graph edge \(\langle d'_n, (t_n, u_n), d_n \rangle \in \mathcal{G}_W\), with \(d'_n = d_{n-1}\), we know by Definition 2 that there exists a triple \(t_n = \text{data}(d_{n-1})\) with \(u_n \in \text{uris}(t_n)\). Moreover, since \(c_{\text{Match}}(t_n, u_n, P) = \text{true}\), there exist both a triple pattern \(tp_k\) in \(P\) and a solution mapping \(\mu\) such that \(\mu[tp_k] = t_n\). Then, given the LDQL query \(q_k = \langle s, P_k \rangle\) that is constructed for \(tp_k\) as specified in the definition of \(\text{LPE}^{\text{Match}}\), it is easy to verify that there exists a solution mapping \(\mu'\) such that \(\mu' \in [P_k]_G^D\) with \(\mu'(v) = u_n\), where \(v \in \{s, ?s, p, ?o\}\) and \(D = \text{dataset}_W(\{u_{n-1}\})\) with default graph \(G = \text{data}(d_{n-1})\). Then, by Definition 5, we also have \(\mu \in [q_k]W^{u_{n-1}}\) and, thus, \(u_n \in [q_k]W^{u_{n-1}}\). Since \(\langle ?v, q_k \rangle\) is a disjunct in \(\text{LPE}^{\text{Match}}\), we also obtain \(u_n \in [q_k]W^{u_{n-1}}\) and, thus, \(u_n \in [\text{LPE}^{\text{Match}}]W\).

As argued before, as a consequence of \(u_n \in [\text{LPE}^{\text{Match}}]W\) (and \(u_0 \in S\)), we can show that document \(d_R = \text{adoc}(u_n)\) is in \(D_{LPE}^{\text{Match}}\) by using \(u_n\) as \(u\) and \(u_0\) as \(u_{\text{ctx}}\) in the definition of \(D_{LPE}^{\text{Match}}\) (cf. Lemma 4). Therefore, the set \(D_{LPE}^{\text{Match}}\) is not only a subset of the set of all documents that are \((c_{\text{Match}}, S, P)\)-reachable in \(W\) (as shown before), but also a superset. Hence, both sets are equivalent and, thus, Lemma 4 holds for \(c_{\text{Match}}\).
A.9 Proof of Theorem 6

In the proof we use the following simple LDQL query $Q(\langle x \rangle)$ given by

$$\langle \langle p, \_ \rangle, \langle x, \_ \rangle, \langle \_ , \_ \rangle \rangle.$$  

We prove first that the reachability criterion $c_{\text{None}}$ cannot express $Q(\langle x \rangle)$. On the contrary, assume that there exists a SPARQL pattern $P$ such that

$$[P]_{W}^{R(c_{\text{None}},S)} = [Q(\langle x \rangle)]_{W}^{S}$$

for every $S$ and $W$. Let $u, u', a, b$ be different elements in $\mathcal{U}$ that are not mentioned in $P$. Consider now $W_1$ having only two documents $d_1 = \langle (u, p, u') \rangle$ and $d_2 = \langle (a, a, a) \rangle$ and such that $\text{adoc}(u) = d_1$ and $\text{adoc}(u') = d_2$. Moreover, consider $W_2$ having also two documents $d_1 = \langle (u, p, u') \rangle$ and $d_3 = \langle (b, b, b) \rangle$ such that $\text{adoc}(u) = d_1$ and $\text{adoc}(u') = d_3$. First notice that

$$[Q(\langle x \rangle)]_{W_1}^{u} = \{\langle x \rightarrow a \rangle\} \neq [Q(\langle x \rangle)]_{W_2}^{u} = \{\langle x \rightarrow b \rangle\}$$

It is easy to see that $[P]_{W_1}^{R(c_{\text{None}},\{u\})} = [P]_{W_2}^{R(c_{\text{None}},\{u\})}$. Just notice that from $\{u\}$, the set of reachable documents following the $c_{\text{None}}$ criterion is the same set $\{d_1\}$ in both $W_1$ and $W_2$. Thus we have that $[P]_{W_1}^{R(c_{\text{None}},\{u\})} = [P]_{W_2}^{R(c_{\text{None}},\{u\})}$ but $[Q(\langle x \rangle)]_{W_1}^{u} \neq [Q(\langle x \rangle)]_{W_2}^{u}$ which is a contradiction.

To continue with the proof, we now show that the reachability criterion $c_{\text{All}}$ cannot express $Q(\langle x \rangle)$. To obtain a contradiction, assume that there exists a pattern $P$ such that

$$[P]_{W}^{R(c_{\text{All}},S)} = [Q(\langle x \rangle)]_{W}^{S}$$

for every $S$ and $W$. Let $u, u', a, b$ be different elements in $\mathcal{U}$ that are not mentioned in $P$. Consider now $W_1 = \{d_1, d_2, d_3\}$ having three documents $d_1 = \langle (u, p, u') \rangle$, $d_2 = \langle (a, a, a) \rangle$ and $d_3 = \langle (b, b, b) \rangle$ and such that $\text{adoc}(u) = d_1$, $\text{adoc}(u') = d_2$ and $\text{adoc}(a) = d_3$. Moreover, consider $W_2 = \{d_1, d_2, d_3\}$ having exactly the same documents as $W_1$, and such that $\text{adoc}(u) = d_1$, $\text{adoc}(u') = d_3$ and $\text{adoc}(b) = d_3$. First notice that

$$[Q(\langle x \rangle)]_{W_1}^{u} = \{\langle x \rightarrow a \rangle\} \neq [Q(\langle x \rangle)]_{W_2}^{u} = \{\langle x \rightarrow b \rangle\}.$$  

Now notice that from $\{u\}$, the set of reachable documents in $W_1$ following the $c_{\text{All}}$ criterion is the set $\{d_1, d_2, d_3\}$; $d_1$ is the document associated to $u$, $d_2$ is reachable from $d_1$ via the URI $u'$, and $d_3$ is reachable from $d_2$ via the URI $a$. Moreover, the set reachable documents from $\{u\}$ in $W_2$ is also $\{d_1, d_2, d_3\}$; $d_1$ is the document associated to $u$, $d_3$ is reachable from $d_2$ via the URI $a'$, and $d_2$ is reachable from $d_3$ via URI $b$. Given that the set of reachable documents is the same in both $W_1$ and $W_2$ we have $[P]_{W_1}^{R(c_{\text{All}},\{u\})} = [P]_{W_2}^{R(c_{\text{All}},\{u\})}$. Given that $[Q(\langle x \rangle)]_{W_1}^{u} \neq [Q(\langle x \rangle)]_{W_2}^{u}$ we obtain our desired contradiction.

We consider now the case of $c_{\text{Match}}$, and prove that it cannot express $Q(\langle x \rangle)$. To obtain a contradiction, assume that there exists a pattern $P$ such that

$$[P]_{W}^{R(c_{\text{Match}},S)} = [Q(\langle x \rangle)]_{W}^{S}.$$
for every $S$ and $W$. Let $u, u', u''$, $a$ be different elements in $\mathcal{U}$ that are not mentioned in $P$. Consider now $W_1$ having two documents $d_1 = \{(u, p, u')\}$ and $d_2 = \{(a, a, a)\}$ and such that $adoc(u) = d_1$ and $adoc(u') = d_2$. Moreover, consider $W_2$ having also two documents $d'_1 = \{(u'', p, u')\}$ and $d'_2 = \{(a, a, a)\}$ such that $adoc(u) = d'_1$ and $adoc(u') = d'_2$. First notice that
\[
[Q(?x)]^{u}_{W_1} = \{?x \rightarrow a\} \neq [Q(?x)]^{u}_{W_2} = \emptyset.
\]

We prove now that $[P]^{R(c_{\text{Match}}, \{u\})}_{W_1} = [P]^{R(c_{\text{Match}}, \{u\})}_{W_2}$. Now given that $d_1$ is the document associated to $u$ in $W_1$, we have that $d'_1$ is $(c_{\text{Match}}, \{u\}, P)$-reachable in $W_1$. Similarly, we know that $d'_2$ is $(c_{\text{Match}}, \{u\}, P)$-reachable in $W_2$. Moreover, given that $P$ does not mention $u, u'$ and $u''$ we have that $(u, p, u')$ matches a triple pattern in $P$ if and only if $(u'', p, u')$ matches a triple pattern in $P$. Thus we have that $d_2$ is $(c_{\text{Match}}, \{u\}, P)$-reachable in $W_1$ if and only if $d'_2$ is $(c_{\text{Match}}, \{u\}, P)$-reachable in $W_2$. Thus we have only two cases, either

- $\{d_1\}$ is the set of $(c_{\text{Match}}, \{u\}, P)$-reachable documents in $W_1$, and $\{d'_1\}$ is the set of $(c_{\text{Match}}, \{u\}, P)$-reachable documents in $W_2$, or
- $\{d_1, d_2\}$ is the set of $(c_{\text{Match}}, \{u\}, P)$-reachable documents in $W_1$, and $\{d'_1, d'_2\}$ is the set of $(c_{\text{Match}}, \{u\}, P)$-reachable documents in $W_2$.

In the first case we have that $[P]^{R(c_{\text{Match}}, \{u\})}_{W_1}$ is obtained by evaluating $P$ over graph $G_1 = \{(u, p, u')\}$ and $[P]^{R(c_{\text{Match}}, \{u\})}_{W_2}$ is obtained by evaluating $P$ over graph $G_2 = \{(u'', p, u')\}$. Given that $P$ does not mention $u, u'$ and $u''$ we obtain that the evaluation of $P$ over $G_1$ is the same as the evaluation of $P$ over $G_2$ which implies that $[P]^{R(c_{\text{Match}}, \{u\})}_{W_1} = [P]^{R(c_{\text{Match}}, \{u\})}_{W_2}$.

In the second case, $[P]^{R(c_{\text{Match}}, \{u\})}_{W_1}$ is obtained by evaluating $P$ over graph $G_1 = \{(u, p, u'), (a, a, a)\}$ and $[P]^{R(c_{\text{Match}}, \{u\})}_{W_2}$ is obtained by evaluating $P$ over graph $G_2 = \{(u'', p, u'), (a, a, a)\}$. For the same reason as above we have that the evaluation of $P$ is the same over $G_1$ and over $G_2$ which implies that $[P]^{R(c_{\text{Match}}, \{u\})}_{W_1} = [P]^{R(c_{\text{Match}}, \{u\})}_{W_2}$. We have proved that $[P]^{R(c_{\text{Match}}, \{u\})}_{W_1} = [P]^{R(c_{\text{Match}}, \{u\})}_{W_2}$, which is our desired contradiction.

### A.10 Proof of Proposition 2

**Property 1** Let $q$ be an arbitrary basic LDQL query of the form $\langle lpe, P \rangle$ such that $lpe$ is Web-safe. To show that $q$ is Web-safe we provide Algorithm 1. In line 3 the algorithm calls a subroutine, EXECLPE, that evaluates a given LPE in the context of a given URI (cf. Algorithm 2). The correctness of the algorithm and its subroutine is easily checked. Moreover, a trivial proof by induction on the possible structure of LPEs can show that for any Web-safe LPE, the given subroutine looks up a finite number of URIs only. The crux of such a proof is twofold: First, the evaluation of LPEs of the form $lpe^\ast$ (lines 26 to 34 in Algorithm 2) is guaranteed to reach a fixed point for any finite Web of Linked Data. Second, the evaluation of LPEs of the form $\langle ?v, q \rangle$ (lines 38 to 42) uses an algorithm for subquery $q$ that has the properties as required in Definition 6. Due to the Web-safeness of the given LPE and, thus, of $q$, such an algorithm exists.
Algorithm 1 Execution of a basic LDQL query \( (lpe, P) \) using a set \( S \) of URIs as seed.

| Line | Description |
|------|-------------|
| 1    | \( \Phi := \text{a new empty set of URIs} \) |
| 2    | for all \( u \in S \) do |
| 3    | \( \Phi := \Phi \cup \text{ExecLPE}(lpe, u) \) |
| 4    | end for |
| 5    | \( G := \text{a new empty set of RDF triples (i.e., an empty RDF graph)} \) |
| 6    | \( \mathcal{N} := \text{a new empty set of pairs consisting of a URI and an RDF graph} \) |
| 7    | for all \( u \in \Phi \) do |
| 8    | if looking up URI \( u \) results in retrieving a document, say \( d \) then |
| 9    | \( G := G \cup \text{data}(d) \) |
| 10   | \( \mathcal{N} := \mathcal{N} \cup \{ \langle u, \text{data}(d) \rangle \} \) |
| 11   | end if |
| 12   | end for |
| 13   | return \[ P \mathcal{N}_\mathcal{G}^{(G, \mathcal{N})} \] // \[ P \mathcal{N}_\mathcal{G}^{(G, \mathcal{N})} \] can be computed by using any algorithm that implements // the standard (set-based) SPARQL evaluation function [2] |

Property 2 First, let \( q \) be an LDQL query of the form \( \pi_V q' \) such that subquery \( q' \) is Web-safe. Due to the Web-safeness of \( q' \), there exists an algorithm for \( q' \) that has the properties as required in Definition 6. We may use this algorithm to construct an algorithm for \( q \); that is, our algorithm for \( q \) calls the algorithm for \( q' \), applies the projection operator to the result, and returns the set of solution mappings resulting from this projection. Since the application of the projection operator does not involve URI lookups, the constructed algorithm for \( q \) has the properties as required in Definition 6. Second, let \( q \) be an LDQL query of the form \( \text{SEED} \ U q' \) such that \( q' \) is Web-safe. Hence, there exists an algorithm for \( q' \) that has the properties as required in Definition 6. Then, showing the Web-safeness of \( q \) is trivial because the algorithm for \( q' \) can also be used for \( q \).

Property 3 Let \( q \) be an LDQL query of the form \( (q_1 \cup \ldots \cup q_n) \) such that each subquery \( q_i \) \( (1 \leq i \leq n) \) is Web-safe. Hence, for each subquery \( q_i \), there exists an algorithm that has the properties as required in Definition 6. Then, the Web-safeness of query \( q \) is easily shown by specifying another algorithm that calls the algorithms of the subqueries sequentially and unions their results.

A.11 Proof of Lemma 3

Lemma 3 follows from Definition 7 and Buil-Aranda et al.’s result [4, Proposition 1].

A.12 Proof of Theorem 7

We prove Theorem 7 based on Algorithm 3, which is an iterative algorithm that generalizes the execution strategy outlined for query \( q''_\mathcal{ex} \) in Example 6. That is, the algorithm executes the subqueries \( q_1, q_2, \ldots, q_m \) sequentially in the order \( \prec \) such that each iteration step (lines 2 to 24) executes one of the subqueries by using the solution mappings computed during the previous step (which are passed on via the sets \( \Omega_0, \Omega_1, \ldots, \Omega_m \)).
Algorithm 2 EXEC\textsc{LPE}(lpe, u_{ctx})

1: if looking up URI $u_{ctx}$ results in retrieving a document, say $d_{ctx}$ then
2: if $lpe$ is $\varepsilon$ then
3: return a new singleton set $\{u_{ctx}\}$
4: else if $lpe$ is a link pattern $lp = (y_1, y_2, y_3)$ then
5: $lp' := (y'_1, y'_2, y'_3)$, where $(y'_1, y'_2, y'_3)$ is a link pattern generated from $lp$ such that any occurrence of symbol $+$ in $lp$ is replaced by URI $u_{ctx}$
6: $\Phi := a$ new empty set of URIs
7: for all $\langle x_1, x_2, x_3 \rangle \in \text{data}(d_{ctx})$ do
8: if $(y'_1 = x_1$ or $y'_1 = $ ) and $(y'_2 = x_2$ or $y'_2 = $) and $(y'_3 = x_3$ or $y'_3 = $) then
9: for all $i \in \{1, 2, 3\}$ do
10: if $y'_i = $ and $x_i$ is a URI whose lookup retrieves a document then
11: $\Phi := \Phi \cup \{x_i\}$
12: end if
13: end for
14: end if
15: end for
16: return $\Phi$
17: else if $lpe$ is of the form $lpe_1/lpe_2$ then
18: $\Phi' := \text{EXEC\textsc{LPE}}(lpe_1, u_{ctx})$
19: $\Phi := a$ new empty set of URIs
20: for all $u' \in \Phi'$ do $\Phi := \Phi \cup \text{EXEC\textsc{LPE}}(lpe_2, u')$ end for
21: return $\Phi$
22: else if $lpe$ is of the form $lpe_1/lpe_2$ then
23: $\Phi_1 := \text{EXEC\textsc{LPE}}(lpe_1, u_{ctx})$
24: $\Phi_2 := \text{EXEC\textsc{LPE}}(lpe_2, u_{ctx})$
25: return $\Phi_1 \cup \Phi_2$
26: else if $lpe$ is of the form $l'$ then
27: $\Phi_{cur} := \text{EXEC\textsc{LPE}}(\varepsilon, u_{ctx})$
28: $lpe' := l$
29: repeat
30: $\Phi_{prev} := \Phi_{cur}$
31: $\Phi_{cur} := \Phi_{cur} \cup \text{EXEC\textsc{LPE}}(lpe', u_{ctx})$
32: $lpe' :=$ an LPE of the form $lpe'/l$
33: until $\Phi_{cur} = \Phi_{prev}$
34: return $\Phi_{cur}$
35: else if $lpe$ is of the form $[lpe']$ then
36: $\Phi := \text{EXEC\textsc{LPE}}(lpe', u_{ctx})$
37: if $\Phi \neq \emptyset$ then return a new singleton set $\{u_{ctx}\}$ else return a new empty set end if
38: else if $lpe$ is of the form $(?v, q)$ then
39: $\Omega := \text{EXEC}(q, \{u_{ctx}\})$ // where EXEC denotes an arbitrary algorithm that can be used // to compute the $\{u_{ctx}\}$-based evaluation of $q$ over the queried // Web of Linked Data
40: $\Phi := a$ new empty set of URIs
41: for all $\mu \in \Omega$ for which $?v \in \text{dom}(\mu)$ and $\mu(?v) \in \mathcal{U}$ do $\Phi := \Phi \cup \{\mu(?v)\}$ end for
42: return $\Phi$
43: end if
44: else
45: return a new empty set
46: end if
Algorithm 3 Execution of an LDQL query \( q \) of the form \( (q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m) \) using a finite set \( S \) of URIs as seed.

Require: \( m \geq 1 \)

Require: LDQL query \( q \) is given as an array \( Q \) consisting of all subqueries of \( q \) such that the order of the subqueries in this array satisfies the conditions as given in Theorem 7.

\begin{algorithm}
\begin{algorithmic}
\State \( \Omega_0 \) := \( \{\mu_q\} \), where \( \mu_q \) is the empty solution mapping; i.e., \( \text{dom}(\mu_q) = \emptyset \)
\For {\( j := 1, \ldots, m \)}
\State \( \Omega_{\text{tmp}} \) := a new empty set of solution mappings
\State \( q_j \) := the \( j \)-th subquery in array \( Q \)
\If {\( q_j \) is of the form \((\text{SEED } \mathcal{U} v q')\)}
\State \( U_{\text{tmp}} \) := a new empty set of URIs
\ForAll {\( \mu \in \Omega_{j-1} \)}
\If {\( \mu(?v) \) is a URI}
\State \( U_{\text{tmp}} := U_{\text{tmp}} \cup \{\mu(?v)\} \)
\EndIf
\EndFor
\ForAll {\( u \in U_{\text{tmp}} \)}
\State \( \Omega_{\text{tmp}} := \Omega_{\text{tmp}} \cup \text{EXEC}(q', \{u\}) \) // where EXEC denotes an arbitrary algorithm that can be used to compute the \( \{u\} \)-based evaluation of \( q' \) over the queried Web of Linked Data
\EndFor
\Else
\State \( \Omega_{\text{tmp}} := \text{EXEC}(q_j, S) \) // where EXEC denotes an arbitrary algorithm that can be used to compute the \( S \)-based evaluation of \( q_j \) over the queried Web of Linked Data
\EndIf
\EndFor
\State \( \Omega_1 \) := a new empty set of solution mappings
\ForAll {\( \mu \in \Omega_{j-1} \)}
\ForAll {\( \mu' \in \Omega_{\text{tmp}} \)}
\If {\( \mu \) and \( \mu' \) are compatible}
\State \( \Omega_j := \Omega_j \cup \{\mu_{\text{join}}\} \), where \( \mu_{\text{join}} = \mu \cup \mu' \)
\EndIf
\EndFor
\EndFor
\EndFor
\State return \( \Omega_m \)
\end{algorithmic}
\end{algorithm}

To prove that Algorithm 3 has the properties as required in Definition 6 we have to show that the algorithm is sound and complete (i.e., for any finite set \( S \) of URIs and any Web of Linked Data \( W \), the algorithm returns \( \llbracket q \rrbracket^S_W \) and that it is guaranteed to look up a finite number of URIs only. We show these properties by induction on the \( m \) iteration steps performed by the algorithm. To this end, we assume that the indices as used for the subqueries \( q_1, q_2, \ldots, q_m \) reflect the order \( \prec \), that is, subquery \( q_1 \) is the first according to \( \prec \), subquery \( q_2 \) is the second, and so on.

Base Case \((m = 1)\): By the conditions in Theorem 7, the first subquery (according to \( \prec \) must be Web-safe and, thus, cannot be of the form \((\text{SEED } \mathcal{U} v q')\). Hence, the
algorithm enters the corresponding else-branch (line 14). Due to the Web-safeness of \( q_1 \), there exists an algorithm for subquery \( q_1 \), say \( A_1 \), that has the properties as required in Definition 6. Algorithm 3 uses algorithm \( A_1 \) to obtain \( \Omega_{tmp} = [q_1]_W \) (where \( W \) is the queried Web of Linked Data), which requires only a finite number of URI lookups. Thereafter, Algorithm 3 computes \( \Omega_1 = \Omega_0 \times \Omega_{tmp} \) (lines 16 to 23) and returns \( \Omega_1 \) (line 25), which does not require any more URI lookups. Hence, for \( m = 1 \), the algorithm looks up a finite number of URIs (if the queried Web of Linked Data is finite).

Since \( \Omega_0 \) contains only the empty solution mapping \( \mu_0 \) (line 1), which is compatible with any other solution mapping, we have \( \Omega_1 = \Omega_{tmp} \) and, thus, \( \Omega_1 = [q_1]_W \).

**Induction Step** \((m > 1)\): By induction we assume that after completing the \((m-1)\)-th iteration, the algorithm has looked up a finite number of URIs only and the current intermediate result \( \Omega_{m-1} \) covers the conjunction of subqueries \( q_1, q_2, \ldots, q_{m-1} \); that is, \( \Omega_{m-1} = \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_{m-1})\}_W^S \). We show that the \( m \)-th iteration also looks up a finite number of URIs only and that \( \Omega_m = \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m)\}_W^S \).

If subquery \( q_m \) is Web-safe, it is not difficult to see these properties: Since \( q_m \) is Web-safe, there exists an algorithm for \( q_m \), say \( A_m \), that has the properties as required in Definition 6. The corresponding call of algorithm \( A_m \) in line 14 of Algorithm 3 looks up a finite number of URIs only, and the subsequent join computation in lines 16 to 23 does not require any more lookups. Moreover, the result of calling algorithm \( A_m \) in line 14 is \( \Omega_{tmp} = [q_m]_W \) and, since the subsequent join computation returns \( \Omega_m = \Omega_{m-1} \times \Omega_{tmp} \), we have \( \Omega_m = \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m)\}_W^S \), as desired.

It remains to discuss the case of subquery \( q_m \) being of the form \((\text{seed} \ ?v \ q')\), where, by the conditions in Theorem 7, subquery \( q' \) is Web-safe. Hence, there exists an algorithm for \( q' \), say \( A' \), that has the properties as required in Definition 6. In this case, Algorithm 3 first iterates over all solution mappings in \( \Omega_{m-1} \) to populate a set \( U_{tmp} \) with all URIs that any of these mappings binds to variable \( ?v \) (lines 6 to 9). Due to the finiteness assumed for all queried Webs of Linked Data (cf. Definition 6), \( \Omega_{m-1} \) is finite. Hence, the resulting set \( U_{tmp} \) contains a finite number of URIs. Therefore, the subsequent loop in lines 10 to 12 calls algorithm \( A' \) a finite number of times and, thus, the \( m \)-th iteration looks up a finite number of URIs only. To show the remaining claim, \( \Omega_m = \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m)\}_W^S \), we first show \( \Omega_m \subseteq \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m)\}_W^S \). Let \( \mu_{\text{join}} \) be an arbitrary solution mapping in \( \Omega_m \); i.e., \( \mu_{\text{join}} \in \Omega_m \). By lines 17 to 23, there exist solution mappings \( \mu \) and \( \mu' \) such that (i) \( \mu \in \Omega_{m-1} = \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_{m-1})\}_W^S \), (ii) \( \mu' \in \Omega_{tmp} = \bigcup_{u \in U_{tmp}} [q'_u]_W^S \), (iii) \( \mu \) and \( \mu' \) are compatible, and (iv) \( \mu_{\text{join}} = \mu \cup \mu' \). Then, by Definition 5, we have \( \mu_{\text{join}} \in \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m)\}_W^S \) and, thus, \( \Omega_m \subseteq \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m)\}_W^S \).

Finally, we show \( \Omega_m \supseteq \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m)\}_W^S \). Assume an arbitrary solution mapping \( \mu^* \in \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m)\}_W^S \). Then, by Definition 5, there exist two solution mappings \( \mu_1^* \) and \( \mu_2^* \) such that (i) \( \mu_1^* \in \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_{m-1})\}_W^S \), (ii) \( \mu_2^* \in \{q_m\}_W^S \), (iii) \( \mu_1^* \) and \( \mu_2^* \) are compatible, and (iv) \( \mu^* = \mu_1^* \cup \mu_2^* \). By our induction hypothesis, we have \( \mu_1^* \in \Omega_{m-1} \). Then, given lines 17 to 23, we have to show that \( \mu_2^* \in \Omega_{tmp} \) where \( \Omega_{tmp} \) is the set of solution mappings computed during the \( m \)-th iteration. Since \( q_m \) is of the form \((\text{seed} \ ?v \ q')\), it holds that \( \Omega_{tmp} = \bigcup_{u \in U_{tmp}} [q'_u]_W^S \) where \( U_{tmp} = \{u \in U \mid ?v u = \text{for some } u \in \{(q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_{m-1})\}_W^S \} \).

Hence, to show that \( \mu_2^* \in \Omega_{tmp} \) we show that there exists a URI \( u \in U_{tmp} \) such that \( \mu_2^* \)}
is in \([q']_W^{[u]}\). Since \(\mu_2^* \in \{q_m\}_{W}^{[u]}\), by Definition 5, solution mapping \(\mu_2^*\) binds variable \(?v\) to a URI, say \(u^*\); i.e., \(?v \in \text{dom}(\mu_2^*)\) and \(\mu_2^*(?v) = u^*\) with \(u^* \in U\). Furthermore, by Lemma 3 and the condition in Theorem 7 (i.e., \(?v \in \bigcup_{k \neq m} \text{sbvars}(q_k)\)), solution mapping \(\mu_2^*\) also has a binding for variable \(?v\), and, since \(\mu_1^*\) and \(\mu_2^*\) are compatible, these bindings are the same, that is, \(\mu_1^*(?v) = \mu_2^*(?v)\). Hence, for URI \(u^* = \mu_2^*(?v)\) it holds that \(u^* \in U_{\text{tmp}}\). Then, by Definition 5, we obtain that \(\mu_2^* \in \{q'\}_W^{[u]}\), which shows that \(\mu_2^* \in \Omega_{\text{tmp}}\) and, thus, we can conclude that \(\Omega_m \supseteq \{q_1 \text{ AND } q_2 \text{ AND } \ldots \text{ AND } q_m\\}_{W}^{[u]}\).

### A.13 Proof of Corollary 1

Corollary 1 is an immediate consequence of Lemma 2.