SPARQL for a Web of Linked Data: Semantics and Computability

Olaf Hartig

Humboldt-Universität zu Berlin
hartig@informatik.hu-berlin.de

Abstract. The World Wide Web currently evolves into a Web of Linked Data where content providers publish and link data as they have done with hypertext for the last 20 years. While the declarative query language SPARQL is the de facto for querying a-priory defined sets of data from the Web, no language exists for querying the Web of Linked Data itself. However, it seems natural to ask whether SPARQL is also suitable for such a purpose.

In this paper we formally investigate the applicability of SPARQL as a query language for Linked Data on the Web. In particular, we study two query models: 1) a *full-Web semantics* where the scope of a query is the complete set of Linked Data on the Web and 2) a family of *reachability-based semantics* which restrict the scope to data that is reachable by traversing certain data links. For both models we discuss properties such as monotonicity and computability as well as the implications of querying a Web that is infinitely large due to data generating servers.

1 Introduction

The emergence of vast amounts of RDF data on the WWW has spawned research on storing and querying large collections of such data efficiently. The prevalent query language in this context is SPARQL [16] which defines queries as functions over an RDF dataset, that is, a fixed, a-priory defined collection of sets of RDF triples. This definition naturally fits the use case of querying a repository of RDF data copied from the Web.

However, most RDF data on the Web is published following the Linked Data principles [5], contributing to the emerging Web of Linked Data [6]. This practice allows for query approaches that access the most recent version of remote data on demand. More importantly, query execution systems may automatically discover new data by traversing data links. As a result, such a system answers queries based on data that is not only up-to-date but may also include initially unknown data. These features are the foundation for true serendipity, which we regard as the most distinguishing advantage of querying the Web itself, instead of a predefined, bounded collection of data.

While several research groups work on systems that evaluate SPARQL basic graph patterns over the Web of Linked Data (cf. [9], [10,12] and [13,14]), we notice a shortage of work on theoretical foundations and properties of such queries. Furthermore, there is a need to support queries that are more expressive than conjunctive (basic graph pattern based) queries [17]. However, it seems natural to assume that SPARQL could be used in this context because the Web of Linked Data is based on the RDF data model and SPARQL is a query language for RDF data. In this paper we challenge this assumption.

E. Simperl et al. (Eds.): ESWC 2012, LNCS 7295, pp. 8–23, 2012.

Contributions. In this paper we understand queries as functions over the Web of Linked Data as a whole. To analyze the suitability of SPARQL as a language for such queries, we have to adjust the semantics of SPARQL. More precisely, we have to redefine the scope for evaluating SPARQL algebra expressions. In this paper we discuss two approaches for such an adjustment. The first approach uses a semantics where the scope of a query is the complete set of Linked Data on the Web. We call this semantics *full-Web semantics*. The second approach introduces a family of *reachability-based semantics* which restrict the scope to data that is reachable by traversing certain data links. We emphasize that both approaches allow for query results that are based on data from initially unknown sources and, thus, enable applications to tap the full potential of the Web. Nevertheless, both approaches precisely define the (expected) result for any query.

As a prerequisite for defining the aforementioned semantics and for studying theoretical properties of queries under these semantics, we introduce a theoretical framework. The basis of this framework is a data model that captures the idea of a Web of Linked Data. We model such a Web as an infinite structure of documents that contain RDF data and that are interlinked via this data. Our model allows for infiniteness because the number of entities described in a Web of Linked Data may be infinite; so may the number of documents. The following example illustrates such a case:

Example 1. Let u_i denote an HTTP scheme based URI that identifies the natural number i. There is a countably infinite number of such URIs. The WWW server which is responsible for these URIs may be set up to provide a document for each natural number. These documents may be generated upon request and may contain RDF data including the RDF triple $(u_i, \text{http://.../next}, u_{i+1})$. This triple associates the natural number i with its successor i+1 and, thus, links to the data about i+1 [19]. An example for such a server is provided by the Linked Open Numbers project[1].

In addition to the data model our theoretical framework comprises a computation model. This model is based on a particular type of Turing machine which formally captures the limited data access capabilities of computations over the Web.

We summarize the main contributions of this paper as follows:

- We present a data model and a computation model that provide a theoretical framework to define and to study query languages for the Web of Linked Data.
- We introduce a full-Web semantics and a family of reachability-based semantics for a (hypothetical) use of SPARQL as a language for queries over Linked Data.
- We systematically analyze SPARQL queries under the semantics that we introduce. This analysis includes a discussion of satisfiability, monotonicity, and computability of queries under the different semantics, a comparison of the semantics, and a study of the implications of querying a Web of Linked Data that is infinite.

Related Work. Since its emergence the WWW has attracted research on declarative query languages for the Web. For an overview on early work in this area we refer to [8]. Most of this work understands the WWW as a hypertext Web. Nonetheless, some of the foundational work can be adopted for research on Linked Data. The computation model that we use in this paper is an adaptation of the ideas presented in [1] and [15].

[1] `http://km.aifb.kit.edu/projects/numbers/`

In addition to the early work on Web queries, query execution over Linked Data on the WWW has attracted much attention recently [9,10,12,13,14]. However, existing work primarily focuses on various aspects of (query-local) data management, query execution, and optimization. The only work we are aware of that aims to formally capture the concept of Linked Data and to provide a well-defined semantics for queries in this context is Bouquet et al.'s [7]. They define three types of query methods for conjunctive queries: a bounded method which only uses RDF data referred to in queries, a direct access method which assumes an oracle that provides all RDF graphs which are "relevant" for a given query, and a navigational method which corresponds to a particular reachability-based semantics. For the latter Bouquet et al. define a notion of reachability that allows a query execution system to follow *all* data links. As a consequence, the semantics of queries using this navigational method is equivalent to, what we call, c_{All}-semantics (cf. Section 5.1); it is the most general of our reachability-based semantics. Bouquet et al.'s navigational query model does not support other, more restrictive notions of reachability, as is possible with our model. Furthermore, Bouquet et al. do not discuss full SPARQL, theoretical properties of queries, or the infiniteness of the WWW.

While we focus on the query language SPARQL in the context of Linked Data on the Web, the theoretical properties of SPARQL as a query language for a fixed, predefined collection of RDF data are well understood today [2,3,16,18]. Particularly interesting in our context are semantical equivalences between SPARQL expressions [18] because these equivalences may also be used for optimizing SPARQL queries over Linked Data.

Structure of the Paper. The remainder of this paper is organized as follows. Section 2 introduces the preliminaries for our work. In Section 3 we present the data model and the computation model. Sections 4 and 5 discuss the full-Web semantics and the reachability-based semantics for SPARQL, respectively. We conclude the paper in Section 6. For full technical proofs of all results in this paper we refer to [11].

2 Preliminaries

This section provides a brief introduction of RDF and the query language SPARQL.

We assume pairwise disjoint, countably infinite sets $\mathcal{U}$ (all HTTP scheme based URIs[2]), $\mathcal{B}$ (blank nodes), $\mathcal{L}$ (literals), and $\mathcal{V}$ (variables, denoted by a leading '?' symbol). An RDF triple t is a tuple $(s, p, o) \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. For any RDF triple $t = (s, p, o)$ we define $\mathrm{terms}(t) = \{s, p, o\}$ and $\mathrm{uris}(t) = \mathrm{terms}(t) \cap \mathcal{U}$. Overloading function terms, we write $\mathrm{terms}(G) = \bigcup_{t \in G} \mathrm{terms}(t)$ for any (potentially infinite) set G of RDF triples. In contrast to the usual formalization of RDF we allow for infinite sets of RDF triples which we require to study infinite Webs of Linked Data.

In this paper we focus on the core fragment of SPARQL discussed by Pérez et al. [16] and we adopt their formalization approach, that is, we use the algebraic syntax and the compositional set semantics introduced in [16]. *SPARQL expression*s are defined recursively: i) A *triple pattern* $(s, p, o) \in (\mathcal{V} \cup \mathcal{U}) \times (\mathcal{V} \cup \mathcal{U}) \times (\mathcal{V} \cup \mathcal{U} \cup \mathcal{L})$ is a

[2] For the sake of simplicity we assume in this paper that URIs are HTTP scheme based URIs. However, our models and result may be extended easily for all possible types of URIs.

SPARQL expression[3]. ii) If P_1 and P_2 are SPARQL expressions, then $(P_1 \text{ AND } P_2)$, $(P_1 \text{ UNION } P_2)$, $(P_1 \text{ OPT } P_2)$, and $(P_1 \text{ FILTER } R)$ are SPARQL expressions where R is a filter condition. For a formal definition of filter conditions we refer to [16]. To denote the set of all variables in all triple patterns of a SPARQL expression P we write $\text{vars}(P)$.

To define the semantics of SPARQL we introduce *valuations*, that are, partial mappings $\mu : \mathcal{V} \to \mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$. The *evaluation* of a SPARQL expression P over a potentially infinite set G of RDF triples, denoted by $[\![P]\!]_G$, is a set of valuations. In contrast to the usual case, this set may be infinite in our scenario. The evaluation function $[\![\cdot]\!]$ is defined recursively over the structure of SPARQL expressions. Due to space limitations, we do not reproduce the full formal definition of $[\![\cdot]\!]$ here. Instead, we refer the reader to the definitions given by Pérez et al. [16]; even if Pérez et al. define $[\![\cdot]\!]$ for finite sets of RDF triples, it is trivial to extend their formalism for infiniteness (cf. appendix in [11]).

A SPARQL expression P is *monotonic* if for any pair G_1, G_2 of (potentially infinite) sets of RDF triples such that $G_1 \subseteq G_2$, it holds that $[\![P]\!]_{G_1} \subseteq [\![P]\!]_{G_2}$. A SPARQL expression P is *satisfiable* if there exists a (potentially infinite) set G of RDF triples such that $[\![P]\!]_G \neq \varnothing$. It is trivial to show that any non-satisfiable expression is monotonic.

In addition to the traditional notion of satisfiability we shall need a more restrictive notion for the discussion in this paper: A SPARQL expression P is *nontrivially satisfiable* if there exists a (potentially infinite) set G of RDF triples and a valuation μ such that i) $\mu \in [\![P]\!]_G$ and ii) μ provides a binding for at least one variable; i.e. $\text{dom}(\mu) \neq \varnothing$.

Example 2. Let $P_{\mathsf{Ex2}} = tp$ be a SPARQL expression that consists of a single triple pattern $tp = (u_1, u_2, u_3)$ where $u_1, u_2, u_3 \in \mathcal{U}$; hence, tp actually is an RDF triple. For any set G of RDF triples for which $(u_1, u_2, u_3) \in G$ it is easy to see that the evaluation of P_{Ex2} over G contains a single, empty valuation $\mu_\varnothing$, that is, $[\![P_{\mathsf{Ex2}}]\!]_G = \{\mu_\varnothing\}$ where $\text{dom}(\mu_\varnothing) = \varnothing$. In contrast, for any other set G of RDF triples it holds $[\![P_{\mathsf{Ex2}}]\!]_G = \varnothing$. Hence, P_{Ex2} is *not* nontrivially satisfiable (although it is satisfiable).

3 Modeling a Web of Linked Data

In this section we introduce theoretical foundations which shall allow us to define and to analyze query models for Linked Data. In particular, we propose a data model and introduce a computation model. For these models we assume a *static view* of the Web; that is, no changes are made to the data on the Web during the execution of a query.

3.1 Data Model

We model the Web of Linked Data as a potentially infinite structure of interlinked documents. Such documents, which we call Linked Data documents, or *LD document*s for short, are accessed via URIs and contain data that is represented as a set of RDF triples.

Definition 1. *Let $\mathcal{T} = (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ be the infinite set of all possible RDF triples. A* ***Web of Linked Data*** *is a tuple $W = (D, data, adoc)$ where:*

[3] For the sake of a more straightforward formalization we do not permit blank nodes in triple patterns. In practice, each blank node in a SPARQL query can be replaced by a new variable.

- D *is a (finite or countably infinite) set of symbols that represent LD documents.*
- $data$ *is a total mapping* $data : D \to 2^{\mathcal{T}}$ *such that* $\forall\, d \in D : data(d)$ *is finite and* $\forall d_1, d_2 \in D : d_1 \neq d_2 \Rightarrow \text{terms}\big(data(d_1)\big) \cap \mathcal{B} \neq \text{terms}\big(data(d_2)\big) \cap \mathcal{B}$.
- $adoc$ *is a partial, surjective mapping* $adoc : \mathcal{U} \to D$.

While the three elements D, $data$, and $adoc$ completely define a Web of Linked Data in our model, we point out that these elements are abstract concepts and, thus, are not available to a query execution system. However, by retrieving LD documents, such a system may gradually obtain information about the Web. Based on this information the system may (partially) materialize these three elements. In the following we discuss the three elements and introduce additional concepts that we need to define queries.

We say a Web of Linked Data $W = (D, data, adoc)$ is *finite* if and only if D is finite; otherwise, W is *infinite*. Our model allows for infiniteness to cover cases where Linked Data about an infinite number of identifiable entities is generated on the fly. The Linked Open Numbers project (cf. Example 1) illustrates that such cases are possible in practice. Another example is the LinkedGeoData project[4] which provides Linked Data about any circular and rectangular area on Earth [4]. Covering these cases enables us to model queries over such data and analyze the effects of executing such queries.

Even if a Web of Linked Data $W = (D, data, adoc)$ is infinite, Definition 1 requires countability for D. We emphasize that this requirement does not restrict us in modeling the WWW as a Web of Linked Data: In the WWW we use URIs to locate documents that contain Linked Data. Even if URIs are not limited in length, they are words over a finite alphabet. Thus, the infinite set of all possible URIs is countable, as is the set of all documents that may be retrieved using URIs.

The mapping $data$ associates each LD document $d \in D$ in a Web of Linked Data $W = (D, data, adoc)$ with a finite set of RDF triples. In practice, these triples are obtained by parsing d after d has been retrieved from the Web. The actual retrieval mechanism is not relevant for our model. However, as prescribed by the RDF data model, Definition 1 requires that the data of each $d \in D$ uses a unique set of blank nodes.

To denote the (potentially infinite but countable) set of *all RDF triples* in W we write $\text{AllData}(W)$; i.e. it holds: $\text{AllData}(W) = \big\{data(d) \,|\, d \in D\big\}$.

Since we use URIs as identifiers for entities, we say that an LD document $d \in D$ *describes* the entity identified by URI $u \in \mathcal{U}$ if there exists $(s, p, o) \in data(d)$ such that $s = u$ or $o = u$. Notice, there might be multiple LD documents that describe an entity identified by u. However, according to the Linked Data principles, each $u \in \mathcal{U}$ may also serve as a reference to a specific LD document which is considered as an authoritative source of data about the entity identified by u. We model the relationship between URIs and authoritative LD documents by mapping $adoc$. Since some LD documents may be authoritative for multiple entities, we do not require injectivity for $adoc$. The "real world" mechanism for dereferencing URIs (i.e. learning about the location of the authoritative LD document) is not relevant for our model. For each $u \in \mathcal{U}$ that cannot be dereferenced (i.e. "broken links") or that is not used in W it holds $u \notin \text{dom}(adoc)$.

A URI $u \in \mathcal{U}$ with $u \in \text{dom}(adoc)$ that is used in the data of an LD document $d_1 \in D$ constitutes a *data link* to the LD document $d_2 = adoc(u) \in D$. These data links form a

[4] `http://linkedgeodata.org`

graph structure which we call *link graph*. The vertices in such a graph represent the LD documents of the corresponding Web of Linked Data; edges represent data links.

To study the monotonicity of queries over a Web of Linked Data we require a concept of containment for such Webs. For this purpose, we introduce the notion of an induced subweb which resembles the concept of induced subgraphs in graph theory.

Definition 2. *Let $W = (D, data, adoc)$ and $W' = (D', data', adoc')$ be Webs of Linked Data. W' is an **induced subweb of** W if i) $D' \subseteq D$, ii) $\forall\, d \in D' : data'(d) = data(d)$, and iii) $\forall\, u \in \mathcal{U}_{D'} : adoc'(u) = adoc(u)$ where $\mathcal{U}_{D'} = \{u \in \mathcal{U} \mid adoc(u) \in D'\}$.*

It can be easily seen from Definition 2 that specifying D' is sufficient to unambiguously define an induced subweb $(D', data', adoc')$ of a given Web of Linked Data. Furthermore, it is easy to verify that for an induced subweb W' of a Web of Linked Data W it holds $\mathrm{AllData}(W') \subseteq \mathrm{AllData}(W)$.

In addition to the structural part, our data model introduces a general understanding of queries over a Web of Linked Data:

Definition 3. *Let $\mathcal{W}$ be the infinite set of all possible Webs of Linked Data (i.e. all 3-tuples that correspond to Definition 1) and let Ω be the infinite set of all possible valuations. A **Linked Data query** q is a total function $q\colon \mathcal{W} \to 2^{\Omega}$.*

The notions of satisfiability and monotonicity carry over naturally to Linked Data queries: A Linked Data query q is *satisfiable* if there exists a Web of Linked Data W such that $q(W)$ is not empty. A Linked Data query q is *nontrivially satisfiable* if there exists a Web of Linked Data W and a valuation μ such that i) $\mu \in q(W)$ and ii) $\mathrm{dom}(\mu) \neq \varnothing$. A Linked Data query q is *monotonic* if for every pair W_1, W_2 of Webs of Linked Data it holds: If W_1 is an induced subweb of W_2, then $q(W_1) \subseteq q(W_2)$.

3.2 Computation Model

Usually, functions are computed over structures that are assumed to be fully (and directly) accessible. In contrast, we focus on Webs of Linked Data in which accessibility is limited: To discover LD documents and access their data we have to dereference URIs, but the full set of those URIs for which we may retrieve documents is unknown. Hence, to properly analyze a query model for Webs of Linked Data we must define a model for computing functions on such a Web. This section introduces such a model.

In the context of queries over a hypertext-centric view of the WWW, Abiteboul and Vianu introduce a specific Turing machine called Web machine [1]. Mendelzon and Milo propose a similar machine model [15]. These machines formally capture the limited data access capabilities on the WWW and thus present an adequate abstraction for computations over a structure such as the WWW. Based on these machines the authors introduce particular notions of computability for queries over the WWW. These notions are: *(finitely) computable queries*, which correspond to the traditional notion of computability; and *eventually computable queries* whose computation may not terminate but each element of the query result will eventually be reported during the computation. We adopt the ideas of Abiteboul and Vianu and of Mendelzon and Milo for our work. More precisely, we adapt the idea of a Web machine to our scenario of a Web of Linked

Data. We call our machine a *Linked Data machine* (or LD machine, for short). Based on this machine we shall define finite and eventual computability for Linked Data queries.

Encoding (fragments of) a Web of Linked Data $W = (D, data, adoc)$ on the tapes of such an LD machine is straightforward because all relevant structures, such as the sets D or $\mathcal{U}$, are countably infinite. In the remainder of this paper we write $\mathrm{enc}(x)$ to denote the encoding of some element x (e.g. a single RDF triple, a set of triples, a full Web of Linked Data, a valuation, etc.). For a detailed definition of the encodings we use in this paper, we refer to the appendix in [11]. We now define LD machine:

Definition 4. *An **LD machine** is a multi-tape Turing machine with five tapes and a finite set of states, including a special state called* expand*. The five tapes include two, read-only input tapes: i) an ordinary input tape and ii) a right-infinite* Web tape *which can only be accessed in the expand state; two work tapes: iii) an ordinary, two-way infinite work tape and iv) a right-infinite* link traversal tape*; and v) a right-infinite, append-only output tape. Initially, the work tapes and the output tape are empty, the Web tape contains a (potentially infinite) word that encodes a Web of Linked Data, and the ordinary input tape contains an encoding of further input (if any). Any LD machine operates like an ordinary multi-tape Turing machine except when it reaches the expand state. In this case LD machines perform the following* expand procedure*: The machine inspects the word currently stored on the link traversal tape. If the suffix of this word is the encoding* $\mathrm{enc}(u)$ *of some URI* $u \in \mathcal{U}$ *and the word on the Web tape contains* $\sharp\,\mathrm{enc}(u)\,\mathrm{enc}(adoc(u))\,\sharp$*, then the machine appends* $\mathrm{enc}(adoc(u))\,\sharp$ *to the (right) end of the word on the link traversal tape by copying from the Web tape; otherwise, the machine appends* $\sharp$ *to the word on the link traversal tape.*

Notice how any LD machine M is limited in the way it may access a Web of Linked Data $W = (D, data, adoc)$ that is encoded on its Web tape: M may use the data of any particular $d \in D$ only after it performed the expand procedure using a URI $u \in \mathcal{U}$ for which $adoc(u) = d$. Hence, the expand procedure simulates a URI based lookup which conforms to the (typical) data access method on the WWW. We now use LD machines to adapt the notion of finite and eventual computability [1] for Linked Data queries:

Definition 5. *A Linked Data query q is **finitely computable** if there exists an LD machine which, for any Web of Linked Data W encoded on the Web tape, halts after a finite number of steps and produces a possible encoding of $q(W)$ on its output tape.*

Definition 6. *A Linked Data q query is **eventually computable** if there exists an LD machine whose computation on any Web of Linked Data W encoded on the Web tape has the following two properties: 1.) the word on the output tape at each step of the computation is a prefix of a possible encoding of $q(W)$ and 2.) the encoding $\mathrm{enc}(\mu')$ of any $\mu' \in q(W)$ becomes part of the word on the output tape after a finite number of computation steps.*

Any machine for a non-satisfiable query may immediately report the empty result. Thus:

Fact 1. *Non-satisfiable Linked Data queries are finitely computable.*

In our analysis of SPARQL-based Linked Data queries we shall discuss decision problems that have a Web of Linked Data W as input. For such problems we assume the computation may only be performed by an LD machine with $\mathrm{enc}(W)$ on its Web tape:

Definition 7. *Let $\mathcal{W}'$ be a (potentially infinite) set of Webs of Linked Data (each of which may be infinite itself); let $\mathcal{X}$ be an arbitrary (potentially infinite) set of finite structures; and let $DP \subseteq \mathcal{W}' \times \mathcal{X}$. The decision problem for DP, that is, decide for any $(W, X) \in \mathcal{W}' \times \mathcal{X}$ whether $(W, X) \in DP$, is **LD machine decidable** if there exist an LD machine whose computation on any $W \in \mathcal{W}'$ encoded on the Web tape and any $X \in \mathcal{X}$ encoded on the ordinary input tape, has the following property: The machine halts in an accepting state if $(W, X) \in DP$; otherwise the machine halts in a rejecting state.*

Obviously, any (Turing) decidable problem that does not have a Web of Linked Data as input, is also LD machine decidable because LD machines are Turing machines; for these problems the corresponding set $\mathcal{W}'$ is empty .

4 Full-Web Semantics

Based on the concepts introduced in the previous section we now define and study approaches that adapt SPARQL as a language for expressing Linked Data queries.

The first approach that we discuss is full-Web semantics where the scope of each query is the complete set of Linked Data on the Web. Hereafter, we refer to SPARQL queries under this full-Web semantics as $\text{SPARQL}_{\text{LD}}$ queries. The definition of these queries is straightforward and makes use of SPARQL expressions and their semantics:

Definition 8. *Let P be a SPARQL expression. The **$\text{SPARQL}_{\text{LD}}$ query that uses** P, denoted by $\mathcal{Q}^P$, is a Linked Data query that, for any Web of Linked Data W, is defined as: $\mathcal{Q}^P(W) = [\![P]\!]_{\text{AllData}(W)}$. Each valuation $\mu \in \mathcal{Q}^P(W)$ is a **solution** for $\mathcal{Q}^P$ in W.*

In the following we study satisfiability, monotonicity, and computability of $\text{SPARQL}_{\text{LD}}$ queries and we discuss implications of querying Webs of Linked Data that are infinite.

4.1 Satisfiability, Nontrivial Satisfiability, Monotonicity, and Computability

For satisfiability and monotonicity we may show the following dependencies.

Proposition 1. *Let $\mathcal{Q}^P$ be a $\text{SPARQL}_{\text{LD}}$ query that uses SPARQL expression P.*

1. *$\mathcal{Q}^P$ is satisfiable if and only if P is satisfiable.*
2. *$\mathcal{Q}^P$ is nontrivially satisfiable if and only if P is nontrivially satisfiable.*
3. *$\mathcal{Q}^P$ is monotonic if and only if P is monotonic.*

We now discuss computability. Since all non-satisfiable $\text{SPARQL}_{\text{LD}}$ queries are finitely computable (recall Fact 1), we focus on satisfiable $\text{SPARQL}_{\text{LD}}$ queries. Our first main result shows that the computability of such queries depends on their monotonicity:

Theorem 1. *If a satisfiable $\text{SPARQL}_{\text{LD}}$ query is monotonic, then it is eventually computable (but not finitely computable); otherwise, it is not even eventually computable.*

In addition to a direct dependency between monotonicity and computability, Theorem 1 shows that not any satisfiable $\text{SPARQL}_{\text{LD}}$ query is finitely computable; instead, such queries are at best eventually computable. The reason for this limitation is the infiniteness of $\mathcal{U}$: To (fully) compute a satisfiable $\text{SPARQL}_{\text{LD}}$ query, an LD machine requires

access to the data of *all* LD documents in the queried Web of Linked Data. Recall that, initially, the machine has no information about what URI to use for performing an expand procedure with which it may access any particular document. Hence, to ensure that all documents have been accessed, the machine must expand all $u \in \mathcal{U}$. This process never terminates because $\mathcal{U}$ is infinite. Notice, a real query system for the WWW would have a similar problem: To guarantee that such a system sees all documents, it must enumerate and lookup all (HTTP scheme) URIs.

The computability of any Linked Data query is a general, input independent property which covers the worst case (recall, the requirements given in Definitions 5 and 6 must hold for any Web of Linked Data). As a consequence, in certain cases the computation of some (eventually computable) SPARQL$_{\mathsf{LD}}$ queries may still terminate:

Example 3. Let $\mathcal{Q}^{P_{\mathsf{Ex2}}}$ be a monotonic SPARQL$_{\mathsf{LD}}$ query which uses the SPARQL expression $P_{\mathsf{Ex2}} = (u_1, u_2, u_3)$ that we introduce in Example 2. Recall, P_{Ex2} is satisfiable but *not* nontrivially satisfiable. The same holds for $\mathcal{Q}^{P_{\mathsf{Ex2}}}$ (cf. Proposition 1). An LD machine for $\mathcal{Q}^{P_{\mathsf{Ex2}}}$ may take advantage of this fact: As soon as the machine discovers an LD document which contains RDF triple (u_1, u_2, u_3), the machine may halt (after reporting $\{\mu_\varnothing\}$ with $\mathrm{dom}(\mu_\varnothing) = \varnothing$ as the complete query result). In this particular case the machine would satisfy the requirements for finite computability. However, $\mathcal{Q}^{P_{\mathsf{Ex2}}}$ is still only eventually computable because there exist Webs of Linked Data that do not contain any LD document with RDF triple (u_1, u_2, u_3); any (complete) LD machine based computation of $\mathcal{Q}^{P_{\mathsf{Ex2}}}$ over such a Web cannot halt (cf. proof of Theorem 1).

The example illustrates that the computation of an eventually computable query over a particular Web of Linked Data may terminate. This observation leads us to a decision problem which we denote as TERMINATION(SPARQL$_{\mathsf{LD}}$). This problem takes a Web of Linked Data W and a satisfiable SPARQL$_{\mathsf{LD}}$ query $\mathcal{Q}^P$ as input and asks whether an LD machine exists that computes $\mathcal{Q}^P(W)$ and halts. For discussing this problem we note that the query in Example 3 represents a special case, that is, SPARQL$_{\mathsf{LD}}$ queries which are satisfiable but not nontrivially satisfiable. The reason why an LD machine for such a query may halt, is the implicit knowledge that the query result is complete once the machine identified the empty valuation $\mu_\varnothing$ as a solution. Such a completeness criterion does not exist for any nontrivially satisfiable SPARQL$_{\mathsf{LD}}$ query:

Lemma 1. *There is not any nontrivially satisfiable SPARQL$_{\mathsf{LD}}$ query $\mathcal{Q}^P$ for which exists an LD machine that, for any Web of Linked Data W encoded on the Web tape, halts after a finite number of computation steps and outputs an encoding of $\mathcal{Q}^P(W)$.*

Lemma 1 shows that the answer to TERMINATION(SPARQL$_{\mathsf{LD}}$) is negative in most cases. However, the problem in general is undecidable (for LD machines) since the input for the problem includes queries that correspond to the aforementioned special case.

Theorem 2. TERMINATION(SPARQL$_{\mathsf{LD}}$) *is not LD machine decidable.*

4.2 Querying an Infinite Web of Linked Data

The limited computability of SPARQL$_{\mathsf{LD}}$ queries that our results in the previous section show, is a consequence of the infiniteness of $\mathcal{U}$ and not of a possible infiniteness of the

queried Web. We now focus on the implications of potentially infinite Webs of Linked Data for SPARQL$_{\mathsf{LD}}$ queries. However, we assume a *finite* Web first:

Proposition 2. *SPARQL$_{\mathsf{LD}}$ queries over a finite Web of Linked Data have a finite result.*

The following example illustrates that a similarly general statement does not exist when the queried Web is infinite such as the WWW.

Example 4. Let $W_{\mathsf{inf}} = (D_{\mathsf{inf}}, data_{\mathsf{inf}}, adoc_{\mathsf{inf}})$ be an *infinite* Web of Linked Data that contains LD documents for all natural numbers (similar to the documents in Example 1). Hence, for each natural number[5] $k \in \mathbb{N}^+$, identified by $u_k \in \mathcal{U}$, exists an LD document $adoc_{\mathsf{inf}}(u_k) = d_k \in D_{\mathsf{inf}}$ such that $data_{\mathsf{inf}}(d_k) = \{(u_k, \mathsf{succ}, u_{k+1})\}$ where $\mathsf{succ} \in \mathcal{U}$ identifies the successor relation for $\mathbb{N}^+$. Furthermore, let $P_1 = (u_1, \mathsf{succ}, ?v)$ and $P_2 = (?x, \mathsf{succ}, ?y)$ be SPARQL expressions. It can be seen easily that the result of SPARQL$_{\mathsf{LD}}$ query $\mathcal{Q}^{P_1}$ over W_{inf} is finite, whereas, $\mathcal{Q}^{P_2}(W_{\mathsf{inf}})$ is infinite.

The example demonstrates that some SPARQL$_{\mathsf{LD}}$ queries have a finite result over some infinite Web of Linked Data and some queries have an infinite result. Consequently, we are interested in a decision problem FINITENESS(SPARQL$_{\mathsf{LD}}$) which asks, given a (potentially infinite) Web of Linked Data W and a satisfiable SPARQL expression P, whether $\mathcal{Q}^P(W)$ is finite. Unfortunately, we cannot answer the problem in general:

Theorem 3. FINITENESS(SPARQL$_{\mathsf{LD}}$) *is not LD machine decidable.*

5 Reachability-Based Semantics

Our results in the previous section show that SPARQL queries under full-Web semantics have a very limited computability. As a consequence, any SPARQL-based query approach for Linked Data that uses full-Web semantics requires some ad hoc mechanism to abort query executions and, thus, has to accept incomplete query results. Depending on the abort mechanism the query execution may even be nondeterministic. If we take these issues as an obstacle, we are interested in an alternative, well-defined semantics for SPARQL over Linked Data. In this section we discuss a family of such semantics which we call *reachability-based semantics*. These semantics restrict the scope of queries to data that is reachable by traversing certain data links using a given set of URIs as starting points. Hereafter, we refer to queries under any reachability-based semantics as *SPARQL$_{\mathsf{LD(R)}}$ queries*. In the remainder of this section we formally introduce reachability-based semantics, discuss theoretical properties of SPARQL$_{\mathsf{LD(R)}}$ queries, and compare SPARQL$_{\mathsf{LD(R)}}$ to SPARQL$_{\mathsf{LD}}$.

5.1 Definition

The basis of any reachability-based semantics is a notion of reachability of LD documents. Informally, an LD document is reachable if there exists a (specific) path in the

[5] In this paper we write $\mathbb{N}^+$ to denote the set of all natural numbers without zero.

link graph of a Web of Linked Data to the document in question; the potential starting points for such a path are LD documents that are authoritative for a given set of entities. However, allowing for arbitrary paths might be questionable in practice because this approach would require following *all* data links (recursively) for answering a query completely. Consequently, we introduce the notion of a reachability criterion that supports an explicit specification of what data links should be followed.

Definition 9. *Let $\mathcal{T}$ be the infinite set of all possible RDF triples and let $\mathcal{P}$ be the infinite set of all possible SPARQL expressions. A* ***reachability criterion*** *c is a (Turing) computable function $c : \mathcal{T} \times \mathcal{U} \times \mathcal{P} \to \{\text{true}, \text{false}\}$.*

An example for a reachability criterion is c_{All} which corresponds to the aforementioned approach of allowing for arbitrary paths to reach LD documents; hence, for each tuple $(t, u, Q) \in \mathcal{T} \times \mathcal{U} \times \mathcal{Q}$ it holds $c_{\mathsf{All}}(t, u, Q) = \text{true}$. The complement of c_{All} is c_{None} which *always* returns false. Another example is c_{Match} which specifies the notion of reachability that we use for link traversal based query execution [10,12].

$$c_{\mathsf{Match}}\Big(t, u, P\Big) = \begin{cases} \text{true} & \text{if there exists a triple pattern } tp \text{ in } P \text{ and } t \text{ matches } tp, \\ \text{false} & \text{else.} \end{cases}$$

where an RDF triple $t = (x_1, x_2, x_3)$ *matches* a triple pattern $tp = (\tilde{x_1}, \tilde{x_2}, \tilde{x_3})$ if for all $i \in \{1, 2, 3\}$ holds: If $\tilde{x_i} \notin \mathcal{V}$, then $\tilde{x_i} = x_i$.

We call a reachability criterion c_1 *less restrictive than* another criterion c_2 if i) for each $(t, u, P) \in \mathcal{T} \times \mathcal{U} \times \mathcal{P}$ for which $c_2(t, u, P) = \text{true}$, also holds $c_1(t, u, P) = \text{true}$ and ii) there exist a $(t', u', P') \in \mathcal{T} \times \mathcal{U} \times \mathcal{P}$ such that $c_1(t', u', P') = \text{true}$ but $c_2(t', u', P') = \text{false}$. It can be seen that c_{All} is the least restrictive criterion, whereas c_{None} is the most restrictive criterion. We now define reachability of LD documents:

Definition 10. *Let $S \subset \mathcal{U}$ be a finite set of seed URIs; let c be a reachability criterion; let P be a SPARQL expression; and let $W = (D, data, adoc)$ be a Web of Linked Data. An LD document $d \in D$* ***is*** *(c, P)-**reachable from** S **in** W if either*

1. *there exists a URI $u \in S$ such that $adoc(u) = d$; or*
2. *there exist $d' \in D$, $t \in data(d')$, and $u \in \text{uris}(t)$ such that i) d' is (c, P)-reachable from S in W, ii) $adoc(u) = d$, and iii) $c(t, u, P) = \text{true}$.*

Based on reachability of LD documents we define reachable parts of a Web of Linked Data. Such a part is an induced subweb covering all reachable LD documents. Formally:

Definition 11. *Let $S \subset \mathcal{U}$ be a finite set of URIs; let c be a reachability criterion; let P be a SPARQL expression; and let $W = (D, data, adoc)$ be a Web of Linked Data. The (S, c, P)-**reachable part of** W, denoted by $W_c^{(S,P)}$, is an induced subweb $(D_{\mathfrak{R}}, data_{\mathfrak{R}}, adoc_{\mathfrak{R}})$ of W such that $D_{\mathfrak{R}} = \big\{d \in D \mid d \text{ is } (c, P)\text{-reachable from } S \text{ in } W\big\}$.*

We now use the concept of reachable parts to define $\text{SPARQL}_{\mathsf{LD(R)}}$ queries.

Definition 12. *Let $S \subset \mathcal{U}$ be a finite set of URIs; let c be a reachability criterion; and let P be a SPARQL expression. The* ***$\text{SPARQL}_{\mathsf{LD(R)}}$ query that uses** P, S, **and** c, denoted by $\mathcal{Q}_c^{P,S}$, is a Linked Data query that, for any Web of Linked Data W, is defined as $\mathcal{Q}_c^{P,S}(W) = [\![P]\!]_{\text{AllData}(W_c^{(S,P)})}$ (where $W_c^{(S,P)}$ is the (S, c, P)-reachable part of W).*

As can be seen from Definition 12, our notion of SPARQL$_{\mathsf{LD(R)}}$ consists of a family of (reachability-based) query semantics, each of which is characterized by a certain reachability criterion. Therefore, we refer to SPARQL$_{\mathsf{LD(R)}}$ queries for which we use a particular reachability criterion c as SPARQL$_{\mathsf{LD(R)}}$ queries *under c-semantics*.

Definition 12 also shows that query results depend on the given set $S \subset \mathcal{U}$ of seed URIs. It is easy to see that any SPARQL$_{\mathsf{LD(R)}}$ query which uses an empty set of seed URIs is not satisfiable and, thus, monotonic and finitely computable. We therefore consider only nonempty sets of seed URIs in the remainder of this paper.

5.2 Completeness and Infiniteness

Definition 12 defines precisely what the sound and complete result of any SPARQL$_{\mathsf{LD(R)}}$ query $\mathcal{Q}_c^{P,S}$ over any Web of Linked Data W is. However, in contrast to SPARQL$_{\mathsf{LD}}$, it is not guaranteed that such a (complete) SPARQL$_{\mathsf{LD(R)}}$ result is complete w.r.t. all data on W. This difference can be attributed to the fact that the corresponding (S, c, P)-reachable part of W may not cover W as a whole. We emphasize that such an incomplete coverage is even possible for the reachability criterion c_{All} because the link graph of W may not be connected; therefore, c_{All}-semantics differs from full-Web semantics. The following result relates SPARQL$_{\mathsf{LD(R)}}$ queries to their SPARQL$_{\mathsf{LD}}$ counterparts.

Proposition 3. *Let $\mathcal{Q}_c^{P,S}$ be a SPARQL$_{\mathsf{LD(R)}}$ query; let $\mathcal{Q}^P$ be the SPARQL$_{\mathsf{LD}}$ query that uses the same SPARQL expression as $\mathcal{Q}_c^{P,S}$; let W be a Web of Linked Data. It holds:*

1. *If $\mathcal{Q}^P$ is monotonic, then $\mathcal{Q}_c^{P,S}(W) \subseteq \mathcal{Q}^P(W)$.*
2. *$\mathcal{Q}_c^{P,S}(W) = \mathcal{Q}^P(W_c^{(S,P)})$. (recall, $W_c^{(S,P)}$ is the (S, c, P)-reachable part of W)*

Since any SPARQL$_{\mathsf{LD}}$ query over a finite Web of Linked Data has a finite result (cf. Proposition 2), we use Proposition 3, case 2, to show the same for SPARQL$_{\mathsf{LD(R)}}$:

Proposition 4. *The result of any SPARQL$_{\mathsf{LD(R)}}$ query $\mathcal{Q}_c^{P,S}$ over a finite Web of Linked Data W is finite; so is the (S, c, P)-reachable part of W.*

For the case of an *infinite* Web of Linked Data the results of SPARQL$_{\mathsf{LD(R)}}$ queries may be either finite or infinite. In Example 4 we found the same heterogeneity for SPARQL$_{\mathsf{LD}}$. However, for SPARQL$_{\mathsf{LD(R)}}$ we may identify the following dependencies.

Proposition 5. *Let $S \subset \mathcal{U}$ be a finite, nonempty set of URIs; let c and c' be reachability criteria; and let P be a SPARQL expression. Let W be an* infinite *Web of Linked Data.*

1. *$W_{c_{\mathsf{None}}}^{(S,P)}$ is always finite; so is $\mathcal{Q}_{c_{\mathsf{None}}}^{P,S}(W)$.*
2. *If $W_c^{(S,P)}$ is finite, then $\mathcal{Q}_c^{P,S}(W)$ is finite.*
3. *If $\mathcal{Q}_c^{P,S}(W)$ is infinite, then $W_c^{(S,P)}$ is infinite.*
4. *If c is less restrictive than c' and $W_c^{(S,P)}$ is finite, then $W_{c'}^{(S,P)}$ is finite.*
5. *If c' is less restrictive than c and $W_c^{(S,P)}$ is infinite, then $W_{c'}^{(S,P)}$ is infinite.*

Proposition 5 provides valuable insight into the dependencies between reachability criteria, the (in)finiteness of reachable parts of an infinite Web, and the (in)finiteness of query results. In practice, however, we are primarily interested in answering two

decision problems: FINITENESSREACHABLEPART and FINITENESS($\text{SPARQL}_{\text{LD(R)}}$). While the latter problem is the $\text{SPARQL}_{\text{LD(R)}}$ equivalent to FINITENESS($\text{SPARQL}_{\text{LD}}$) (cf. Section 4.2), the former has the same input as FINITENESS($\text{SPARQL}_{\text{LD(R)}}$) (that is, a Web of Linked Data and a $\text{SPARQL}_{\text{LD(R)}}$ query) and asks whether the corresponding reachable part of the given Web is finite. Both problems are undecidable in our context:

Theorem 4. FINITENESSREACHABLEPART *and* FINITENESS($\text{SPARQL}_{\text{LD(R)}}$) *are not LD machine decidable.*

5.3 Satisfiability, Nontrivial Satisfiability, Monotonicity, and Computability

We now investigate satisfiability, nontrivial satisfiability, monotonicity, and computability of $\text{SPARQL}_{\text{LD(R)}}$ queries. First, we identify the following dependencies.

Proposition 6. *Let $\mathcal{Q}_c^{P,S}$ be a SPARQL$_{\text{LD(R)}}$ query that uses a nonempty $S \subset \mathcal{U}$.*

1. *$\mathcal{Q}_c^{P,S}$ is satisfiable if and only if P is satisfiable.*
2. *$\mathcal{Q}_c^{P,S}$ is nontrivially satisfiable if and only if P is nontrivially satisfiable.*
3. *$\mathcal{Q}_c^{P,S}$ is monotonic if P is monotonic.*

Proposition 6 reveals a first major difference between $\text{SPARQL}_{\text{LD(R)}}$ and $\text{SPARQL}_{\text{LD}}$: The statement about monotonicity in that proposition is only a material conditional, whereas it is a biconditional in the case of $\text{SPARQL}_{\text{LD}}$ (cf. Proposition 1). The reason for this disparity are $\text{SPARQL}_{\text{LD(R)}}$ queries for which monotonicity is independent of the corresponding SPARQL expression. The following proposition identifies such a case.

Proposition 7. *Any SPARQL$_{\text{LD(R)}}$ query $\mathcal{Q}_{c_{\text{None}}}^{P,S}$ is monotonic if $|S| = 1$.*

Before we may come back to the aforementioned disparity, we focus on the computability of $\text{SPARQL}_{\text{LD(R)}}$ queries. We first show the following, noteworthy result.

Lemma 2. *Let $\mathcal{Q}_c^{P,S}$ be a SPARQL$_{\text{LD(R)}}$ query that is nontrivially satisfiable. There exists an LD machine that computes $\mathcal{Q}_c^{P,S}$ over any (potentially infinite) Web of Linked Data W and that halts after a finite number of computation steps with an encoding of $\mathcal{Q}_c^{P,S}(W)$ on its output tape if and only if the (S, c, P)-reachable part of W is finite.*

The importance of Lemma 2 lies in showing that some computations of nontrivially satisfiable $\text{SPARQL}_{\text{LD(R)}}$ queries may terminate. This possibility presents another major difference between $\text{SPARQL}_{\text{LD(R)}}$ and $\text{SPARQL}_{\text{LD}}$ (recall Lemma 1 which shows that any possible computation of nontrivially satisfiable $\text{SPARQL}_{\text{LD}}$ queries never terminates). Based on Lemma 2 we may even show that a particular class of satisfiable $\text{SPARQL}_{\text{LD(R)}}$ queries are finitely computable. This class comprises all queries that use a reachability criterion which ensures the finiteness of reachable parts of any queried Web of Linked Data. We define this property of reachability criteria as follows:

Definition 13. *A reachability criterion c **ensures finiteness** if for any Web of Linked Data W, any (finite) set $S \subset \mathcal{U}$ of seed URIs, and any SPARQL expression P, the (S, c, P)-reachable part of W is finite.*

We may now show the aforementioned result:

Proposition 8. *Let c be a reachability criterion that ensures finiteness. SPARQL$_{\text{LD(R)}}$ queries under c-semantics are finitely computable.*

While it remains an open question whether the property to ensure finiteness is decidable for all reachability criteria, it is easy to verify the property for criteria which always only accept a given, constant set of data links. For a formal discussion of such criteria, which we call *constant reachability criteria*, we refer to the appendix in [11]. c_{None} is a special case of these criteria; Proposition 5, case 1, verifies that c_{None} ensures finiteness.

Notice, for any reachability criterion c that ensures finiteness, the computability of SPARQL$_{\text{LD(R)}}$ queries under c-semantics does not depend on the monotonicity of these queries. This independence is another difference to SPARQL$_{\text{LD}}$ queries (recall Theorem 1). However, for any other reachability criterion (including c_{Match} and c_{All}), we have a similar dependency between monotonicity and computability of (satisfiable) SPARQL$_{\text{LD(R)}}$ queries, that we have for SPARQL$_{\text{LD}}$ queries (recall Theorem 1):

Theorem 5. *Let c_{nf} be a reachability criterion that does not ensure finiteness. If a satisfiable SPARQL$_{\text{LD(R)}}$ query $\mathcal{Q}^{P,S}_{c_{nf}}$ (under c_{nf}-semantics) is monotonic, then $\mathcal{Q}^{P,S}_{c_{nf}}$ is either finitely computable or eventually computable; otherwise, $\mathcal{Q}^{P,S}_{c_{nf}}$ may not even be eventually computable.*

By comparing Theorems 1 and 5 we notice that SPARQL$_{\text{LD}}$ queries and SPARQL$_{\text{LD(R)}}$ queries (that use a reachability criterion which does not ensure finiteness) feature a similarly limited computability. However, the reasons for both of these results differ significantly: In the case of SPARQL$_{\text{LD}}$ the limitation follows from the infiniteness of $\mathcal{U}$, whereas, for SPARQL$_{\text{LD(R)}}$ the limitation is a consequence of the possibility to query an infinitely large Web of Linked Data.

However, even if the computability of many SPARQL$_{\text{LD(R)}}$ queries is as limited as that of their SPARQL$_{\text{LD}}$ counterparts, there is another major difference: Lemma 2 shows that for (nontrivially satisfiable) SPARQL$_{\text{LD(R)}}$ queries which are not finitely computable, the computation over some Webs of Linked Data may still terminate; this includes all finite Webs (cf. Proposition 4) but also some infinite Webs (cf. proof of Lemma 2). Such a possibility does not exist for nontrivially satisfiable SPARQL$_{\text{LD}}$ queries (cf. Lemma 1). Nonetheless, the termination problem for SPARQL$_{\text{LD(R)}}$ is undecidable in our context.

Theorem 6. TERMINATION(SPARQL$_{\text{LD(R)}}$) *is not LD machine decidable.*

We now come back to the impossibility for showing that SPARQL$_{\text{LD(R)}}$ queries (with a nonempty set of seed URIs) are monotonic *only if* their SPARQL expression is monotonic. Recall, for some SPARQL$_{\text{LD(R)}}$ queries monotonicity is irrelevant for identifying the computability (cf. Proposition 8). We are primarily interested in the monotonicity of all other (satisfiable) SPARQL$_{\text{LD(R)}}$ queries because for those queries computability depends on monotonicity as we show in Theorem 5. Remarkably, for those queries it is possible to show the required dependency that was missing from Proposition 6:

Proposition 9. *Let $\mathcal{Q}^{P,S}_{c_{nf}}$ be a SPARQL$_{\text{LD(R)}}$ query that uses a finite, nonempty $S \subset \mathcal{U}$ and a reachability criterion c_{nf} which does not ensure finiteness. $\mathcal{Q}^{P,S}_{c_{nf}}$ is monotonic only if P is monotonic.*

6 Conclusions

Our investigation of SPARQL as a language for Linked Data queries reveals the following main results. Some special cases aside, the computability of queries under any of the studied semantics is limited and no guarantee for termination can be given. For reachability-based semantics it is at least possible that some of the (non-special case) query computations terminate; although, in general it is undecidable which. As a consequence, any SPARQL-based query system for Linked Data on the Web must be prepared for query executions that discover an infinite amount of data and that do not terminate.

Our results also show that –for reachability-based semantics– the aforementioned issues must be attributed to the possibility for infiniteness in the queried Web (which is a result of data generating servers). Therefore, it seems worthwhile to study approaches for detecting whether the execution of a $\text{SPARQL}_{\text{LD(R)}}$ query traverses an infinite path in the queried Web. However, the mentioned issues may also be addressed by another, alternative well-defined semantics that restricts the scope of queries even further (or differently) than our reachability-based semantics. It remains an open question how such an alternative may still allow for queries that tap the full potential of the Web.

We also show that computability depends on satisfiability and monotonicity and that for (almost all) SPARQL-based Linked Data queries, these two properties directly correspond to the same property for the used SPARQL expression. While Arenas and Pérez show that the core fragment of SPARQL without OPT is monotonic [3], it requires further work to identify (non-)satisfiable and (non-)monotonic fragments and, thus, enable an explicit classification of SPARQL-based Linked Data queries w.r.t. computability.

References

1. Abiteboul, S., Vianu, V.: Queries and computation on the web. Theoretical Computer Science 239(2) (2000)
2. Angles, R., Gutierrez, C.: The Expressive Power of SPARQL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 114–129. Springer, Heidelberg (2008)
3. Arenas, M., Pérez, J.: Querying Semantic Web Data with SPARQL. In: PODS (2011)
4. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData: Adding a Spatial Dimension to the Web of Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 731–746. Springer, Heidelberg (2009)
5. Berners-Lee, T.: Linked Data (2006), `http://www.w3.org/DesignIssues/LinkedData.html`
6. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – the story so far. Journal on Semantic Web and Information Systems 5(3) (2009)
7. Bouquet, P., Ghidini, C., Serafini, L.: Querying the Web of Data: A Formal Approach. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 291–305. Springer, Heidelberg (2009)
8. Florescu, D., Levy, A.Y., Mendelzon, A.O.: Database techniques for the world-wide web: A survey. SIGMOD Record 27(3) (1998)
9. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.-U., Umbrich, J.: Data Summaries for On-Demand Queries over Linked Data. In: WWW (2010)

10. Hartig, O.: Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 154–169. Springer, Heidelberg (2011)
11. Hartig, O.: SPARQL for a Web of Linked Data: Semantics and Computability (Extended Version). CoRR abs/1203.1569 (2012), http://arxiv.org/abs/1203.1569
12. Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL Queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
13. Ladwig, G., Tran, T.: Linked Data Query Processing Strategies. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 453–469. Springer, Heidelberg (2010)
14. Ladwig, G., Tran, T.: SIHJoin: Querying Remote and Local Linked Data. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 139–153. Springer, Heidelberg (2011)
15. Mendelzon, A.O., Milo, T.: Formal models of web queries. Inf. Systems 23(8) (1998)
16. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Transactions on Database Systems 34(3) (2009)
17. Picalausa, F., Vansummeren, S.: What are real SPARQL queries like? In: SWIM (2011)
18. Schmidt, M., Meier, M., Lausen, G.: Foundations of sparql query optimization. In: Proc. of the 13th Int. Conference on Database Theory, ICDT (2010)
19. Vrandecić, D., Krötzsch, M., Rudolph, S., Lösch, U.: Leveraging non-lexical knowledge for the linked open data web. In: RAFT (2010)