Foundations of Traversal Based Query Execution over Linked Data

Olaf Hartig
Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin, Germany
hartig@informatik.hu-berlin.de

Johann-Christoph Freytag
Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin, Germany
freytag@informatik.hu-berlin.de

ABSTRACT

Query execution over the Web of Linked Data has attracted much attention recently. A particularly interesting approach is link traversal based query execution which proposes to integrate the traversal of data links into the creation of query results. Hence –in contrast to traditional query execution paradigms– this does not assume a fixed set of relevant data sources beforehand; instead, the traversal process discovers data and data sources on the fly and, thus, enables applications to tap the full potential of the Web.

While several authors have studied possibilities to implement the idea of link traversal based query execution and to optimize query execution in this context, no work exists that discusses theoretical foundations of the approach in general. Our paper fills this gap.

We introduce a well-defined semantics for queries that may be executed using a link traversal based approach. Based on this semantics we formally analyze properties of such queries. In particular, we study the computability of queries as well as the implications of querying a potentially infinite Web of Linked Data. Our results show that query computation in general is not guaranteed to terminate and that for any given query it is undecidable whether the execution terminates. Furthermore, we define an abstract execution model that captures the integration of link traversal into the query execution process. Based on this model we prove the soundness and completeness of link traversal based query execution and analyze an existing implementation approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; F.1.1 [**Computation by Abstract Devices**]: Models of Computation

General Terms

Management, Theory

Keywords

link traversal based query execution, query semantics, computability, Web of Data, Linked Data

HT'12, June 25–28, 2012, Milwaukee, Wisconsin, USA.

1. INTRODUCTION

During recent years an increasing number of data providers adopted the Linked Data principles for publishing and interlinking structured data on the World Wide Web (WWW) [10]. The Web of Linked Data that emerges from this process enables users to benefit from a virtually unbounded set of data sources and, thus, opens possibilities not conceivable before. Consequently, the Web of Linked Data has spawned research to execute declarative queries over multiple Linked Data sources. Most approaches adapt techniques that are known from the database literature (e.g. data warehousing or query federation). However, the Web of Linked Data is different from traditional database systems; distinguishing characteristics are its unbounded nature and the lack of a database catalog. Due to these characteristics it is impossible to know all data sources that might contribute to the answer of a query. In this context, traditional query execution paradigms are insufficient because those assume a fixed set of potentially relevant data sources beforehand. This assumption presents a restriction that inhibits applications to tap the full potential of the Web; it prevents a serendipitous discovery and utilization of relevant data from unknown sources.

An alternative to traditional query execution paradigms are exploration approaches that traverse links on the Web of Linked Data. These approaches enable a query execution system to automatically discover the most recent data from initially unknown data sources.

The prevalent example of an exploration based approach is link traversal based query execution. The idea of this approach is to intertwine the traversal of data links with the construction of the query result and, thus, to integrate the discovery of data into the query execution process [7]. This general idea may be implemented in various ways. For instance, Ladwig and Tran introduce an asynchronous implementation that adapts the concept of symmetric hash joins [13, 14]; Schmedding proposes an implementation that incrementally adjusts the answer to a query each time the execution system retrieves additional data [18]; our earlier work focuses on an implementation that uses a synchronous pipeline of iterators, each of which is responsible for a particular part of the query [6, 7]. All existing publications focus on approaches for implementing the idea of link traversal based query execution and on query optimization in the context of such an implementation. To our knowledge, no work exists that provides a general foundation for this new query execution paradigm.

We argue that a well-defined query semantics is essential to compare different query execution approaches and to verify implementations. Furthermore, a proper theoretical foundation enables a formal analysis of fundamental properties of queries and query executions. For instance, studying the computability of queries may answer whether particular query executions are guaranteed to terminate. In addition to these more theoretical questions, an under-

```
1  SELECT ?p ?l WHERE {
2     <http://bob.name> <http://.../knows> ?p .
3     ?p   <http://.../currentProject> ?pr .
4     ?pr  <http://.../label> ?l . }
```

Figure 1: Sample query presented in the language SPARQL.

standing of fundamental properties and limitations may help to gain new insight into challenges and possibilities for query planning and optimization. Therefore, in this paper we provide such a formal foundation of Linked Data queries and link traversal based query execution. Our contributions are:

1.) As a basis, we introduce a *theoretical framework* that comprises a data model and a computation model. The data model formalizes the idea of a Web of Linked Data; the computation model captures the limited data access capabilities of computations over the Web.

2.) We present a *query model* that introduces a well-defined semantics for conjunctive queries (which is the type of queries supported by existing link traversal based systems). Basically, the result of such a query is the set of all valuations that map the query to a subset of all Linked Data that is reachable, starting with entity identifiers mentioned in the query. We emphasize that our model does not prescribe a specific notion of reachability; instead, it is possible to make the notion of reachability applied to answer a query can be made explicit (by specifying which data links should be followed).

3.) We formally *analyze properties* of our query model. In particular, we study the implications of querying a potentially infinite Web and show that it is undecidable whether a query result will be finite or infinite. Furthermore, we analyze the computability of queries by adopting earlier work on Web queries which distinguishes finitely computable queries, eventually computable queries, and queries that are not even eventually computable. We prove that queries in our model are eventually computable. Hence, a link traversal based query execution system does not have to deal with queries that are not computable at all. However, we also show that it is undecidable whether a particular query execution terminates.

4.) We define an abstract *query execution model* that formalizes the general idea of link traversal based query execution. This model captures the approach of intertwining link traversal and result construction. Based on this model we prove the soundness and completeness of the new query execution paradigm.

5.) Finally, we use our execution model to formally analyze a particular implementation of link traversal based query execution.

This paper is organized as follows: In Section 2 we present an example that demonstrates the idea of link traversal based query execution. Section 3 defines our data model and our computation model. We present our query model in Section 4 and discuss its properties in Section 5. Section 6 introduces the corresponding execution model. Finally, we discuss related work in Section 7 and conclude the paper in Section 8. For all proofs of theorems, lemmas and propositions in this paper we refer to [8].

2. EXAMPLE EXECUTION

Link traversal based query execution is a novel query execution paradigm tailored to the Web of Linked Data. Since adhering to the Linked Data principles is the minimal requirement for publishing Linked Data on the WWW, the link traversal approach relies solely on these principles; it does not assume that each data source provides a data-local query interface (as would be required for query federation). The only way to obtain data is via URI look-ups.

Usually, Linked Data on the WWW is represented using the RDF data model [11] and queries are expressed using SPARQL [17]. SPARQL queries consist of RDF graph patterns that contain query variables, denoted with the symbol '?'. The semantics of SPARQL is based on pattern matching [16]. Figure 1 provides a SPARQL representation of a query that asks for projects of acquaintances of user Bob, who is identified by URI http://bob.name. In lines 2 to 4 the query contains a conjunctive query represented as a set of three SPARQL triple patterns. In the following we outline a link traversal based execution of this conjunctive query.

$$\big(\ \text{http://}\textbf{bob}\text{.name}\ ,\ \text{http://.../}\textbf{knows}\ ,\ \text{http://}\textbf{alice}\text{.name}\ \big) \in G_{\mathrm{b}}$$

$$\big(\ \text{http://}\textbf{alice}\text{.name}\ ,\ \text{http://.../}\textbf{name}\ ,\ \text{"Alice"}\ \big) \in G_{\mathrm{a}}$$

$$\big(\ \text{http://}\textbf{alice}\text{.name}\ ,\ \text{http://.../}\textbf{currentProject}\ ,\ \text{http://.../}\textbf{AlicesPrj}\ \big) \in G_{\mathrm{a}}$$

$$\big(\ \text{http://.../}\textbf{AlicesPrj}\ ,\ \text{http://.../}\textbf{label}\ ,\ \text{"Alice's Project"}\ \big) \in G_{\mathrm{p}}$$

Figure 2: Excerpts from Linked Data retrieved from the Web.

Link traversal based query execution usually starts with an empty, query-local dataset. We obtain some seed data by looking up the URIs mentioned in the query: For the URI http://bob.name in our sample query we may retrieve a set G_{b} of RDF triples (cf. Figure 2), which we add to the local dataset. Now, we alternate between i) constructing valuations from RDF triples that match a pattern of our query in the query-local dataset, and ii) augmenting the dataset by looking up URIs which are part of these valuations. For the triple pattern in line 2 of our sample query the local dataset contains a matching triple, originating from G_{b}. Hence, we can construct a valuation $\mu_1 = \{\text{?p} \rightarrow \text{http://alice.name}\}$ that maps query variable ?p to the URI http://alice.name. By looking up this URI we may retrieve a set G_{a} of RDF triples, which we also add to the query-local dataset. Based on the augmented dataset we can extend μ_1 by adding a binding for ?pr. We obtain $\mu_2 = \{\text{?p} \rightarrow \text{http://alice.name}, \text{?pr} \rightarrow \text{http://.../AlicesPrj}\}$, which already covers the pattern in line 2 and 3. Notice, constructing μ_2 is only possible because we retrieved G_{a}. However, before we discovered and resolved the URI http://alice.name, we neither knew about G_{a} nor about the existence of the data source from which we retrieved G_{a}. Hence, the traversal of data links enables us to answer queries based on data from initially unknown sources.

We proceed with our execution strategy as follows: We discover and retrieve G_{p} by looking up the URI http://.../AlicesPrj and extend μ_2 to $\mu_3 = \{\text{?p} \rightarrow \text{http://alice.name}, \text{?pr} \rightarrow \text{http://.../AlicesPrj}, \text{?l} \rightarrow \text{"Alice's Project"}\}$, which now covers the whole, conjunctive query. Hence, μ_3 can be reported as the result of that query.

3. MODELING A WEB OF LINKED DATA

In this section we introduce theoretical foundations which shall allow us to define and to analyze queries over Linked Data. In particular, we propose a data model and 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

The WWW is the most prominent implementation of a Web of Linked Data and it shows that the idea of Linked Data scales to a virtually unlimited dataspace. Nonetheless, other implementations are possible (e.g. within the boundaries of a closed, globally distributed corporate network). Such an implementation may be based on the same technologies used for the WWW (i.e. HTTP, URIs, RDF, etc.) or it may use other, similar technologies. Consequently, our data model abstracts from the concrete technologies that implement Linked Data in the WWW and, thus, enables us to study queries over any Web of Linked Data.

As a basis for our model we use a simple, triple based data model for representing the data that is distributed over a Web of Linked

Data (similar to the RDF data model that is used for Linked Data on the WWW). We assume a countably infinite set $\mathcal{I}$ of possible identifiers (e.g. all URIs) and a countably infinite set $\mathcal{L}$ of all possible constant literals (e.g. all possible strings, natural numbers, etc.). $\mathcal{I}$ and $\mathcal{L}$ are disjoint. A *data triple* is a tuple $t \in \mathcal{I} \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{L})$. To denote the set of all identifiers in a data triple t we write $\mathrm{ids}(t)$.

We model a Web of Linked Data as a potentially infinite structure of interlinked documents. Such documents, which we call Linked Data documents, or *LD documents* for short, are accessed via identifiers in $\mathcal{I}$ and contain data that is represented as a set of data triples. The following definition captures our approach:

Definition 1. A **Web of Linked Data** W is a tuple $(D, data, adoc)$ where:

- D is a set of symbols that represent LD documents; D may be finite or countably infinite.
- $data : D \rightarrow 2^{\mathcal{I} \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{L})}$ is a total mapping such that $data(d)$ is finite for all $d \in D$.
- $adoc : \mathcal{I} \rightarrow D$ is a partial, surjective mapping.

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 not directly 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 remainder of this section we discuss the three elements and introduce additional concepts that we need to define our query model.

We say a Web of Linked Data $W = (D, data, adoc)$ is *infinite* if and only if D is infinite; otherwise, we say W is *finite*. Our model allows for infinite Webs to cover the possibility that Linked Data about an infinite number of identifiable entities is generated on the fly. 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].

Another example were data about an infinite number of entities may be generated is the LinkedGeoData project[2] which provides Linked Data about any circular and rectangular area on Earth [2]. These examples illustrate that an infinite Web of Linked Data is possible in practice. 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 is infinite, we require countability for D. We shall see that this requirement has nontrivial consequences: It limits the potential size of Webs of Linked Data in our model and, thus, allows us to use a Turing machine based model for analyzing computability of queries over Linked Data (cf. Section 5.2). We emphasize that the requirement of countability 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 data triples. In practice, these triples are obtained by parsing d after d has been retrieved from the Web. The actual retrieval mechanism depends on the technologies that are used to implement the Web of Linked Data. To denote the potentially infinite (but countable) set of *all data triples* in W we write $\mathrm{AllData}(W)$; i.e. it holds:

$$\mathrm{AllData}(W) = \bigcup_{d \in D} data(d)$$

Since we use elements in the set $\mathcal{I}$ as identifiers for entities, we say that an LD document $d \in D$ *describes* the entity identified by an identifier $id \in \mathcal{I}$ if $\exists (s,p,o) \in data(d) : (s = id \lor o = id)$. Notice, while there might be multiple LD documents in D that describe an entity identified by id, we do not assume that we can enumerate the set of all these documents; i.e., we cannot discover and retrieve all of them. The possibility to query search engines is out of scope of this paper. It is part of our future work to extend the semantics in our query model in order to take data into account, that is reachable by utilizing search engines. However, according to the Linked Data principles, each $id \in \mathcal{I}$ 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 id. We model the relationship between identifiers 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 identifiers (i.e. learning about the location of the corresponding, authoritative LD document) depends on the implementation of the Web of Linked Data and is not relevant for our model. For each identifier $id \in \mathcal{I}$ that cannot be dereferenced (i.e. "broken links") or that is not used in the Web it holds $id \notin \mathrm{dom}(adoc)$.

An identifier $id \in \mathcal{I}$ with $id \in \mathrm{dom}(adoc)$ that is used in the data of an LD document $d_1 \in D$ constitutes a *data link* to the LD document $d_2 = adoc(id) \in D$. To formally represent the graph structure that is formed by such data links, we introduce the notion of a *Web link graph*. The vertices in such a graph represent the LD documents of the corresponding Web of Linked Data; the edges represent data links and are labeled with a data triple that denotes the corresponding link in the source document. Formally:

Definition 2. Let $W = (D, data, adoc)$ be a Web of Linked Data. The **Web link graph for** W, denoted by G^W, is a directed, edge-labeled multigraph (V, E) where $V = D$ and

$$E = \big\{ (d_\mathrm{h}, d_\mathrm{t}, t) \,\big|\, d_\mathrm{h}, d_\mathrm{t} \in D \text{ and } t \in data(d_\mathrm{h}) \text{ and } \\ \exists\, id \in \mathrm{ids}(t) : adoc(id) = d_\mathrm{t} \big\}$$

In our query model we introduce the concept of reachable parts of a Web of Linked Data that are relevant for answering queries; similarly, our execution model introduces a concept for those parts of a Web of Linked Data that have been discovered at a certain point in the query execution process. To provide a formal foundation for these concepts we define the notion of an induced subweb which resembles the concept of induced subgraphs in graph theory.

Definition 3. Let $W = (D, data, adoc)$ be a Web of Linked Data. A Web of Linked Data $W' = (D', data', adoc')$ is an **induced subweb of** W if:

1. $D' \subseteq D$,
2. $\forall\, d \in D' : data'(d) = data(d)$, and
3. $\forall\, id \in \big\{ id \in \mathcal{I} \,\big|\, adoc(id) \in D' \big\} : adoc'(id) = adoc(id)$.

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

[2] http://linkedgeodata.org

It can be easily seen from Definition 3 that specifying D' is sufficient to define an induced subweb $(D', data', adoc')$ of a given Web of Linked Data unambiguously. Furthermore, it is easy to verify that for an induced subweb W' of a Web of Linked Data W it holds $\mathrm{AllData}(W') \subseteq \mathrm{AllData}(W)$.

3.2 Computation Model

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

In earlier work about computation on 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. We adopt the idea of such 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).

Encoding (fragments of) a Web of Linked Data $W = (D, data, adoc)$ on the tapes of such a machine is straightforward because all relevant structures, such as the sets D or $\mathcal{I}$, 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 data triple, a set of triples, a full Web of Linked Data, etc.). For a detailed definition of the encodings we use in this paper, we refer to the appendix in [8].

We now define our adaptation of the idea of Web machines:

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 $\mathrm{enc}(id)$ of some identifier $id \in \mathcal{I}$ and the word on the Web tape contains $\sharp\,\mathrm{enc}(id)\,\mathrm{enc}(adoc(id))\,\sharp$, then the machine appends $\mathrm{enc}(adoc(id))\,\sharp$ 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 an LD machine is limited in the way it may access a Web of Linked Data that is encoded on its Web (input) tape: Any LD document and its data is only available for the computation after the machine performed the expand procedure using a corresponding identifier. Hence, the expand procedure models a URI based lookup which is the (typical) data access method on the WWW.

In the following sections we use the notion of an LD machine for analyzing properties of our query model. In this context we aim to discuss decision problems that shall have a Web of Linked Data W as input. For these problems we assume that the computation may only be performed by an LD machine with $\mathrm{enc}(W)$ on its Web tape:

Definition 5. Let $\mathcal{W}$ be a (potentially infinite) set of Webs of Linked Data; 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, to 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. QUERY MODEL

This section introduces our query model by defining semantics for conjunctive queries over Linked Data.

4.1 Preliminaries

We assume an infinite set $\mathcal{V}$ of possible query variables that is disjoint from the sets $\mathcal{I}$ and $\mathcal{L}$ introduced in the previous section. These variables will be used to range over elements in $\mathcal{I} \cup \mathcal{L}$. Thus, *valuation*s in our context are total mappings from a finite subset of $\mathcal{V}$ to the set $\mathcal{I} \cup \mathcal{L}$. We denote the domain of a particular valuation μ by $\mathrm{dom}(\mu)$. Using valuations we define our general understanding of queries over a Web of Linked Data as follows:

Definition 6. Let $\mathcal{W}$ be a set of all possible Webs of Linked Data (i.e. all 3-tuples that correspond to Definition 1) and let Ω be a set of all possible valuations. A **Linked Data query** q is a total function $q : \mathcal{W} \rightarrow 2^{\Omega}$.

To express *conjunctive* Linked Data queries we adapt the notion of a SPARQL basic graph pattern [17] to our data model:

Definition 7. A **basic query pattern (BQP)** is a finite set $B = \{tp_1, \dots, tp_n\}$ of tuples $tp_i \in (\mathcal{V} \cup \mathcal{I}) \times (\mathcal{V} \cup \mathcal{I}) \times (\mathcal{V} \cup \mathcal{I} \cup \mathcal{L})$ (for $1 \leq i \leq n$). We call such a tuple a **triple pattern**.

In comparison to traditional notions of conjunctive queries, triple patterns are the counterpart of atomic formulas; furthermore, BQPs have no head, hence no bound variables. To denote the set of variables and identifiers that occur in a triple pattern tp we write $\mathrm{vars}(tp)$ and $\mathrm{ids}(tp)$, respectively. Accordingly, the set of variables and identifiers that occur in all triple patterns of a BQP B is denoted by $\mathrm{vars}(B)$ and $\mathrm{ids}(B)$, respectively. For a triple pattern tp and a valuation μ we write $\mu[tp]$ to denote the triple pattern that we obtain by replacing the variables in tp according to μ. Similarly, a valuation μ is applied to a BQP B by $\mu[B] = \{\mu[tp] \mid tp \in B\}$. The result of $\mu[tp]$ is a data triple if $\mathrm{vars}(tp) \subseteq \mathrm{dom}(\mu)$. Accordingly, we introduce the notion of *matching data triples*:

Definition 8. A data triple t **matches** a triple pattern tp if there exists a valuation μ such that $\mu[tp] = t$.

While BQPs are syntactic objects, we shall use them as a representation of Linked Data queries which have a certain semantics. In the remainder of this section we define this semantics. Due to the openness and distributed nature of Webs such as the WWW we cannot guarantee query results that are complete w.r.t. all Linked Data on a Web. Nonetheless, we aim to provide a well-defined semantics. Consequently, we have to limit our understanding of completeness. However, instead of restricting ourselves to data from a fixed set of sources selected or discovered beforehand, we introduce an approach that allows a query to make use of previously unknown data and sources. Our definition of query semantics is based on a two-phase approach: First, we define the part of a Web of Linked Data that is reached by traversing links using the identifiers in a query as a starting point. Then, we formalize the result of such a query as the set of all valuations that map the query to a subset of all data in the reachable part of the Web. Notice, while this two-phase approach provides for a straightforward definition of the query semantics in our model, it does not correspond to the actual query execution strategy of integrating the traversal of data links into the query execution process as illustrated in Section 2.

4.2 Reachability

To introduce the concept of a reachable part of a Web of Linked Data we first define reachability of LD documents. Informally, an LD document is reachable if there exists a (specific) path in the Web 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 entities mentioned (via their identifier) in the queries. However, allowing for arbitrary paths might be questionable in practice because it would require following *all* data links (recursively) for answering a query completely. A more restrictive approach is the notion of *query pattern based reachability* where a data link only qualifies as a part of paths to reachable LD documents, if that link corresponds to a triple pattern in the executed query. The link traversal based query execution illustrated in Section 2 applies this notion of query pattern based reachability (as we show in Section 6.3). Our experience in developing a link traversal based query execution system[3] suggests that query pattern based reachability is a good compromise for answering queries without crawling large portions of the Web that are likely to be irrelevant for the queries. However, other criteria for specifying which data links should be followed might prove to be more suitable in certain use cases. For this reason, we do not prescribe a specific criterion in our query model; instead, we enable our model to support any possible criterion by making this concept part of the model.

Definition 9. Let $\mathcal{T}$ be the infinite set of all possible data triples; let $\mathcal{B}$ be the infinite set of all possible BQPs. A **reachability criterion** c is a total computable function $c : \mathcal{T} \times \mathcal{I} \times \mathcal{B} \rightarrow \{\text{true}, \text{false}\}$.

An example for such a reachability criterion is c_{All} which corresponds to the approach of allowing for arbitrary paths to reach LD documents; hence, for each tuple $(t, id, B) \in \mathcal{T} \times \mathcal{I} \times \mathcal{B}$ it holds $c_{\mathsf{All}}(t, id, B) = \text{true}$. The complement of c_{All} is c_{None} which *always* returns false. Another example is c_{Match} which corresponds to the aforementioned query pattern based reachability. We define c_{Match} based on the notion of matching data triples:

$$c_{\mathsf{Match}}\Big(t, id, B\Big) = \begin{cases} \text{true} & \text{if } \exists\, tp \in B : t \text{ matches } tp, \\ \text{false} & \text{else.} \end{cases} \tag{1}$$

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

Using the concept of reachability criteria for data links we formally define reachability of LD documents:

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

1. there exists an $id \in S$ such that $adoc(id) = d$; or
2. there exist another LD document $d' \in D$, a $t \in data(d')$, and an $id \in \text{ids}(t)$ such that i) d' is (c, B)-reachable from S in W, ii) $c(t, id, B) = \text{true}$, and iii) $adoc(id) = d$.

We note that each LD document which is authoritative for an entity mentioned (via its identifier) in a finite set of seed identifiers S, is always reachable from S in the corresponding Web of Linked Data, independent of the reachability criterion and the BQP used.

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

Definition 11. Let $W = (D, data, adoc)$ be a Web of Linked Data; let $S \subset \mathcal{I}$ be a finite set of seed identifiers; let c be a reachability criterion; and let B be a BQP. The **(S, c, B)-reachable part of W** is the induced subweb $W_c^{(S,B)} = (D_{\mathfrak{R}}, data_{\mathfrak{R}}, adoc_{\mathfrak{R}})$ of W that is defined by

$$D_{\mathfrak{R}} = \big\{ d \in D \,\big|\, d \text{ is } (c, B)\text{-reachable from } S \text{ in } W \big\}$$

4.3 Query Results

Based on the previous definitions we define the semantics of conjunctive Linked Data queries that are expressed via BQPs. Recall that Linked Data queries map from a Web of Linked Data to a set of valuations. Our interpretation of BQPs as Linked Data queries requires that each valuation μ in the result for a particular BQP B satisfies the following requirement: If we replace the variables in B according to μ (i.e. we compute $\mu[B]$), we obtain a set of data triples and this set must be a subset of all data in the part of the Web that is reachable according to the notion of reachability that we apply. Since our model supports a virtually unlimited number of notions of reachability, each of which is defined by a particular reachability criterion, the actual result of a query must depend on such a reachability criterion. The following definition formalizes our understanding of conjunctive Linked Data queries:

Definition 12. Let $S \subset \mathcal{I}$ be a finite set of seed identifiers; let c be a reachability criterion; and let B be a BQP; let W be a Web of Linked Data; let $W_c^{(S,B)}$ denote the (S, c, B)-reachable part of W. The **conjunctive Linked Data query (CLD query)** that uses B, S, and c, denoted by $\mathcal{Q}_c^{B,S}$, is a Linked Data query defined as:

$$\mathcal{Q}_c^{B,S}\big(W\big) = \big\{ \mu \,\big|\, \mu \text{ is a valuation with } \text{dom}(\mu) = \text{vars}(B) \\ \text{and } \mu[B] \subseteq \text{AllData}\big(W_c^{(S,B)}\big) \big\}$$

Each $\mu \in \mathcal{Q}_c^{B,S}\big(W\big)$ is a **solution for $\mathcal{Q}_c^{B,S}$ in W.**

Since we define the result of queries w.r.t. a reachability criterion, the semantics of such queries depends on this criterion. Thus, strictly speaking, our query model introduces a family of query semantics, each of which is characterized by a reachability criterion. Therefore, we refer to a CLD query for which we use a particular reachability criterion c as a CLD query *under c-semantics.*

5. PROPERTIES OF THE QUERY MODEL

In this section we discuss properties of our query model. In particular, we focus on the implications of querying Webs that are infinite and on the (LD machine based) computability of queries.

5.1 Querying an Infinite Web of Linked Data

From Definitions 10 and 11 in Section 4 it can be easily seen that any reachable part of a *finite* Web of Linked Data must also be finite, independent of the query that we want to answer and the

[3] http://squin.org

reachability criterion that we use. Consequently, the result of CLD queries over such a finite Web is also guaranteed to be finite. We shall see that a similarly general statement does not exist when the queried Web is infinite such as the WWW.

To study the implications of querying an infinite Web we first take a look at some example queries. For these examples we assume an *infinite* Web of Linked Data $W_{\mathsf{inf}} = (D_{\mathsf{inf}}, data_{\mathsf{inf}}, adoc_{\mathsf{inf}})$ that contains LD documents for all natural numbers (similar to the documents in Example 1). The data in these documents refers to the successor of the corresponding number and to all its divisors. Hence, for each natural number[4] $k \in \mathbb{N}^+$, identified by $\mathsf{no}_k \in \mathcal{I}$, exists an LD document $adoc_{\mathsf{inf}}(\mathsf{no}_k) = d_k \in D_{\mathsf{inf}}$ such that

$$data_{\mathsf{inf}}(d_k) = \Big\{(\mathsf{no}_k, \mathsf{succ}, \mathsf{no}_{k+1})\Big\} \cup \bigcup_{y \in \mathrm{Div}(k)} \Big\{(\mathsf{no}_k, \mathsf{div}, \mathsf{no}_y)\Big\}$$

where $\mathrm{Div}(k)$ denotes the set of all divisors of $k \in \mathbb{N}^+$, $\mathsf{succ} \in \mathcal{I}$ identifies the successor relation for $\mathbb{N}^+$, and $\mathsf{div} \in \mathcal{I}$ identifies the relation that associates a number $k \in \mathbb{N}^+$ with a divisor $y \in \mathrm{Div}(k)$.

Example 2. *Let $B_1 = \big\{(\mathsf{no}_2, \mathsf{succ}, ?x)\big\}$ be a BQP ($?x \in \mathcal{V}$) that asks for the successor of 2. Recall, $data_{\mathsf{inf}}(d_2)$ contains three data triples: $(\mathsf{no}_2, \mathsf{succ}, \mathsf{no}_3)$, $(\mathsf{no}_2, \mathsf{div}, \mathsf{no}_1)$, and $(\mathsf{no}_2, \mathsf{div}, \mathsf{no}_2)$. We consider reachability criteria c_{All}, c_{Match}, and c_{None} (cf. Section 4.2) and $S_1 = \{\mathsf{no}_2\}$: The $(S_1, c_{\mathsf{All}}, B_1)$-reachable part of W_{inf} is infinite and consists of[5] the LD documents $d_1, \dots, d_k, \dots$. In contrast, the $(S_1, c_{\mathsf{Match}}, B_1)$-reachable part $W^{(S_1,B_1)}_{c_{\mathsf{Match}}}$ and the $(S_1, c_{\mathsf{None}}, B_1)$-reachable part $W^{(S_1,B_1)}_{c_{\mathsf{None}}}$ are finite: $W^{(S_1,B_1)}_{c_{\mathsf{Match}}}$ consists of d_2 and d_3, whereas $W^{(S_1,B_1)}_{c_{\mathsf{None}}}$ only consists of d_2. The query result in all three cases contains a single solution μ for which $\mathrm{dom}(\mu) = \{?x\}$ and $\mu(?x) = \mathsf{no}_3$; i.e. $\mu = \{?x \rightarrow \mathsf{no}_3\}$.*

Example 3. *We now consider the BQP $B_2 = \big\{(\mathsf{no}_2, \mathsf{succ}, ?x),$ $(?x, \mathsf{succ}, ?y), (?z, \mathsf{div}, ?x)\big\}$ with $?x, ?y, ?z \in \mathcal{V}$ and $S_2 = \{\mathsf{no}_2\}$. Under c_{None}-semantics the query result is empty because the $(S_2, c_{\mathsf{None}}, B_2)$-reachable part of W_{inf} only consists of LD document d_2 (as in the previous example). For c_{All} and c_{Match} the reachable parts are infinite (and equal): Both consist of the documents $d_1, \dots, d_k, \dots$ (as was the case for c_{All} but not for c_{Match} in the previous example). While the query result is also equal for both criteria, it differs significantly from the previous example because it is infinite: $\mathcal{Q}^{B_2,S_2}_{c_{\mathsf{Match}}}\big(W_{\mathsf{inf}}\big) = \mathcal{Q}^{B_2,S_2}_{c_{\mathsf{All}}}\big(W_{\mathsf{inf}}\big) = \{\mu_1, \mu_2, \dots \mu_i, \dots\}$ where*

$$\begin{aligned} \mu_1 &= \{?x \rightarrow \mathsf{no}_3, ?y \rightarrow \mathsf{no}_4, ?z \rightarrow \mathsf{no}_3\}, \\ \mu_2 &= \{?x \rightarrow \mathsf{no}_3, ?y \rightarrow \mathsf{no}_4, ?z \rightarrow \mathsf{no}_6\}, \\ \textit{and, in general:} \quad \mu_i &= \{?x \rightarrow \mathsf{no}_3, ?y \rightarrow \mathsf{no}_4, ?z \rightarrow \mathsf{no}_{(3i)}\}. \end{aligned}$$

A special type of CLD queries not covered by the examples are queries that use an empty set of seed identifiers. However, it is easily verified that answering such queries is trivial:

Fact 1. *Let W be a Web of Linked Data. For each CLD query $\mathcal{Q}^{B,S}_c$ for which $S = \varnothing$, it holds: The set of LD documents in the (S, c, B)-reachable part of W is empty and, thus, $\mathcal{Q}^{B,S}_c\big(W\big) = \varnothing$.*

Due to its triviality, an empty set of seed identifiers presents a special case that we exclude from most of our results. We now summarize the conclusions that we draw from Examples 2 and 3:

Proposition 1. *Let $S \subset \mathcal{I}$ be a finite but nonempty set of seed identifiers; let c and c' be reachability criteria; let B be a BQP; and let W be an* infinite *Web of Linked Data. It holds:*

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

Proposition 1 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 the following questions: Does the execution of a given CLD query reach an infinite number of LD documents? Do we have to expect an infinite query result? We formalize these questions as (LD machine) decision problems:

Problem:	FinitenessReachablePart
Web Input:	a (potentially infinite) Web of Linked Data W
Ordin. Input:	a CLD query $\mathcal{Q}^{B,S}_c$ where S is nonempty and c is less restrictive than c_{None}
Question:	Is the (S, c, B)-reachable part of W finite?

Problem:	FinitenessQueryResult
Web Input:	a (potentially infinite) Web of Linked Data W
Ordin. Input:	a CLD query $\mathcal{Q}^{B,S}_c$ where S is nonempty and c is less restrictive than c_{None}
Question:	Is the query result $\mathcal{Q}^{B,S}_c\big(W\big)$ finite?

Unfortunately, it is impossible to define a general algorithm for answering these problems as our following result shows.

Theorem 1. *The problems* FinitenessReachablePart *and* FinitenessQueryResult *are not LD machine decidable.*

5.2 Computability of Linked Data Queries

Example 3 illustrates that some CLD queries may have a result that is infinitely large. Even if a query has a finite result it may still be necessary to retrieve infinitely many LD documents to ensure that the computed result is complete. Hence, any attempt to answer such queries completely induces a non-terminating computation.

In what follows, we formally analyze feasibility and limitations for computing CLD queries. For this analysis we adopt notions of computability that Abiteboul and Vianu introduce in the context of queries over a hypertext-centric view of the WWW [1]. 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. While Abiteboul and Vianu define these notions of computability using their concept of a Web machine (cf. Section 3.2), our adaptation for Linked Data queries uses an LD machine:

Definition 13. 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 14. 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

[4] In this paper we write $\mathbb{N}^+$ to denote the set of all natural numbers without zero. $\mathbb{N}^0$ denotes all natural numbers, including zero.

[5] We assume $\mathsf{succ} \notin \mathrm{dom}(adoc_{\mathsf{inf}})$ and $\mathsf{div} \notin \mathrm{dom}(adoc_{\mathsf{inf}})$.

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.

We now analyze the computability of CLD queries. As a preliminary we identify a dependency between the computation of a CLD query over a particular Web of Linked Data and the (in)finiteness of the corresponding reachable part of that Web:

Lemma 1. *The result of a CLD query $\mathcal{Q}_c^{B,S}$ over a (potentially infinite) Web of Linked Data W can be computed by an LD machine that halts after a finite number of computation steps if and only if the (S, c, B)-reachable part of W is finite.*

The following, immediate consequence of Lemma 1 is trivial.

Corollary 1. *CLD queries that use an empty set of seed identifiers and CLD queries under c_{None}-semantics are finitely computable.*

While Corollary 1 covers some special cases, the following result identifies the computability of CLD queries in the general case.

Theorem 2. *Each CLD query is either finitely computable or eventually computable.*

Theorem 2 emphasizes that execution systems for CLD queries do not have to deal with queries that are not even eventually computable. Theorem 2 also shows that query computations in the general case are not guaranteed to terminate. The reason for this result is the potential infiniteness of Webs of Linked Data. However, even if a CLD query is only eventually computable, its computation over a particular Web of Linked Data may still terminate (even if this Web is infinite). Thus, in practice, we are interested in criteria that allow us to decide whether a particular query execution is guaranteed to terminate. We formalize this decision problem:

Problem:	COMPUTABILITYCLD
Web Input:	a (potentially infinite) Web of Linked Data W
Ordin. Input:	a CLD query $\mathcal{Q}_c^{B,S}$ where S is nonempty and c is less restrictive than c_{None}
Question:	Does an LD machine exist that i) computes $\mathcal{Q}_c^{B,S}(W)$ and ii) halts?

Unfortunately:

Theorem 3. COMPUTABILITYCLD *is not LD machine decidable.*

As a consequence of the results in this section we note that any system which executes CLD queries over an infinite Web of Linked Data (such as the WWW) must be prepared for query executions that do not terminate and that discover an infinite amount of data.

6. QUERY EXECUTION MODEL

In Section 4 we use a two-phase approach to define (a family of) semantics for conjunctive queries over Linked Data. A query execution system that would directly implement this two-phase approach would have to retrieve all LD documents before it could generate the result for a query. Hence, the first solutions could only be generated after all data links (that qualify according to the used reachability criterion) have been followed recursively. Retrieving the complete set of reachable documents may exceed the resources of the execution system or it may take a prohibitively long time; it is even possible that this process does not terminate at all (cf. Section 5.2). The link traversal based query execution that we demonstrate in Section 2 applies an alternative strategy: It intertwines the link traversal based retrieval of data with a pattern matching process that generates solutions incrementally. Due to such an integration of link traversal and result construction it is possible to report first solutions early, even if not all links have been followed and not all data has been retrieved. To describe link traversal based query execution formally, we introduce an abstract query execution model. In this section we present this model and use it for proving soundness and completeness of the modeled approach.

6.1 Preliminaries

Usually, queries are executed over a finite structure of data (e.g. an instance of a relational schema or an RDF dataset) that is assumed to be fully available to the execution system. However, in this paper we are concerned with queries over a Web of Linked Data that may be infinite and that is fully unknown at the beginning of a query execution process. To learn about such a Web we have to dereference identifiers and parse documents that we retrieve. Conceptually, dereferencing an identifier corresponds to achieving partial knowledge of the set D and mapping $adoc$ with which we model the queried Web of Linked Data $W = (D, data, adoc)$. Similarly, parsing documents retrieved from the Web corresponds to learning mapping $data$. To formally represent what we know about a Web of Linked Data at any particular point in a query execution process we introduce the concept of discovered parts.

Definition 15. A **discovered part** of a Web of Linked Data W is an induced subweb of W that is finite.

We require finiteness for discovered parts of a Web of Linked Data W. This requirement models the fact that we obtain information about W only gradually; thus, at any point in a query execution process we only know a finite part of W, even if W is infinite.

The (link traversal based) execution of a CLD query $\mathcal{Q}_c^{B,S}$ over a Web of Linked Data $W = (D, data, adoc)$ starts with a discovered part $\mathfrak{D}_{\mathsf{init}}^{S,W}$ (of W) which contains only those LD documents from W that can be retrieved by dereferencing identifiers from S; hence, $\mathfrak{D}_{\mathsf{init}}^{S,W} = (D_0, data_0, adoc_0)$ is defined by:

$$D_0 = \big\{ adoc(id) \,\big|\, id \in S \text{ and } id \in \mathrm{dom}(adoc) \big\} \tag{2}$$

In the remainder of this section we first define how we may use data from a discovered part to construct (partial) solutions for a CLD query in an incremental fashion. Furthermore, we formalize how the link traversal approach expands such a discovered part in order to construct further solutions. Finally, we discuss an abstract procedure that formally captures how the approach intertwines the expansion of discovered parts with the construction of solutions.

6.2 Constructing Solutions

The query execution approach that we aim to capture with our query execution model constructs solutions for a query incrementally (cf. Section 2). To formalize the intermediate products of such a construction we introduce the concept of partial solutions.

Definition 16. A **partial solution** for CLD query $\mathcal{Q}_c^{B,S}$ in a Web of Linked Data W is a pair (P, μ) where $P \subseteq B$ and $\mu \in \mathcal{Q}_c^{P,S}\big(W\big)$.

According to Definition 16 each partial solution (P, μ) for a CLD query $\mathcal{Q}_c^{B,S}$ is a solution for the CLD query $\mathcal{Q}_c^{P,S}$ that uses BQP P (instead of B). Since P is a part of B we say that partial solutions *cover* only a part of the queries that we want to answer.

The (link traversal based) execution of a CLD query $\mathcal{Q}_c^{B,S}$ over a Web of Linked Data W starts with an *empty partial solution* $\sigma_0 = (P_0, \mu_0)$ which covers the empty part $P_0 = \varnothing$ of B (i.e. $\mathrm{dom}(\mu_0) = \varnothing$). During query execution we (incrementally) extend partial solutions to cover larger parts of B. Those partial solutions that cover the whole query can be reported as solutions for

$\mathcal{Q}_c^{B,S}$ in W. However, to extend a partial solution we may use data only from LD documents that we have already discovered. Consequently, the following definition formalizes the extension of a partial solution based on a discovered part of a Web of Linked Data.

Definition 17. Let $W_{\mathfrak{D}}$ be a discovered part of a Web of Linked Data W; let $\mathcal{Q}_c^{B,S}$ be a CLD query; and let $\sigma = (P, \mu)$ be a partial solution for $\mathcal{Q}_c^{B,S}$ in W. If there exist a triple pattern $tp \in B \setminus P$ and a data triple $t \in \mathrm{AllData}\big(W_{\mathfrak{D}}\big)$ such that t matches tp then the (t, tp)**-augmentation of** σ **in** $W_{\mathfrak{D}}$, denoted by $aug_{t,tp}^{W_{\mathfrak{D}}}(\sigma)$, is a pair (P', μ') such that $P' = P \cup \{tp\}$ and μ' extends μ as follows: 1.) $\mathrm{dom}(\mu') = \mathrm{vars}(P')$ and 2.) $\mu'[P'] = \mu[P] \cup \{t\}$.

The following proposition shows that the result of augmenting a partial solution is again a partial solution, as long as the discovered part of the Web that we use for such an augmentation is fully contained in the reachable part of the Web.

Proposition 2. *Let $W_{\mathfrak{D}}$ be a discovered part of a Web of Linked Data W and let $\mathcal{Q}_c^{B,S}$ be a CLD query. If $W_{\mathfrak{D}}$ is an induced subweb of the (S, c, B)-reachable part of W and σ is a partial solution for $\mathcal{Q}_c^{B,S}$ in W, then $aug_{t,tp}^{W_{\mathfrak{D}}}(\sigma)$ is also a partial solution for $\mathcal{Q}_c^{B,S}$ in W, for all possible t and tp.*

6.3 Traversing Data Links

During query execution we may traverse data links to expand the discovered part. Such an expansion may allow us to compute further augmentations for partial solutions. The link traversal based approach implements such an expansion by dereferencing identifiers that occur in valuations μ of partial solutions (cf. Section 2). Formally, we define such a valuation based expansion as follows:

Definition 18. Let $W_{\mathfrak{D}} = (D_{\mathfrak{D}}, data_{\mathfrak{D}}, adoc_{\mathfrak{D}})$ be a discovered part of a Web of Linked Data $W = (D, data, adoc)$ and let μ be a valuation. The μ**-expansion** of $W_{\mathfrak{D}}$ in W, denoted by $exp_{\mu}^{W}\big(W_{\mathfrak{D}}\big)$, is an induced subweb $(D'_{\mathfrak{D}}, data'_{\mathfrak{D}}, adoc'_{\mathfrak{D}})$ of W, defined by $D'_{\mathfrak{D}} = D_{\mathfrak{D}} \cup \Delta^{W}(\mu)$ where

$$\Delta^{W}(\mu) = \big\{ adoc\big(\mu(?v)\big) \,\big|\, ?v \in \mathrm{dom}(\mu) \text{ and } \mu(?v) \in \mathrm{dom}(adoc) \big\}$$

The following propositions show that expanding discovered parts is a monotonic operation (Proposition 3) and that the set of all possible discovered parts is closed under this operation (Proposition 4).

Proposition 3. *Let $W_{\mathfrak{D}}$ be a discovered part of a Web of Linked Data W, then $W_{\mathfrak{D}}$ is an induced subweb of $exp_{\mu}^{W}\big(W_{\mathfrak{D}}\big)$, for all possible μ.*

Proposition 4. *Let $W_{\mathfrak{D}}$ be a discovered part of a Web of Linked Data W, then $exp_{\mu}^{W}\big(W_{\mathfrak{D}}\big)$ is also a discovered part of W, for all possible μ.*

We motivate the expansion of discovered parts of a queried Web of Linked Data by the possibility that data obtained from additionally discovered documents may allow us to construct more (partial) solutions. However, Proposition 2 indicates that the augmentation of partial solutions is only sound if the discovered part that we use for the augmentation is fully contained in the corresponding reachable part of the Web. Thus, in order to use a discovered part that has been expanded based on (previously constructed) partial solutions, it should be guaranteed that the expansion never exceeds the reachable part. Under c_{Match}-semantics we have such a guarantee:

Proposition 5. *Let $\sigma = (P, \mu)$ be a partial solution for a CLD query $\mathcal{Q}_{c_{\mathsf{Match}}}^{B,S}$ (under c_{Match}-semantics) in a Web of Linked Data W; and let $W_{c_{\mathsf{Match}}}^{(S,B)}$ denote the $(S, c_{\mathsf{Match}}, B)$-reachable part of W. If a discovered part $W_{\mathfrak{D}}$ of W is an induced subweb of $W_{c_{\mathsf{Match}}}^{(S,B)}$, then $exp_{\mu}^{W}\big(W_{\mathfrak{D}}\big)$ is also an induced subweb of $W_{c_{\mathsf{Match}}}^{(S,B)}$.*

We explain the restriction to c_{Match}-semantics in Proposition 5 as follows: During link traversal based query execution we expand the discovered part of the queried Web only by using valuations that occur in partial solutions (cf. Section 2). Due to this approach, we only dereference identifiers for which there exists a data triple that matches a triple pattern in our query. Hence, this approach indirectly enforces query pattern based reachability (cf. Section 4.2). As a result, link traversal based query execution only supports CLD queries under c_{Match}-semantics; so does our query execution model.

6.4 Combining Construction and Traversal

Although incrementally expanding the discovered part of the reachable subweb and recursively augmenting partial solutions may be understood as separate processes, the idea of link traversal based query execution is to combine these two processes. We now introduce an abstract procedure which captures this idea formally.

As a basis for our formalization we represent the state of a query execution by a pair $\big(\mathfrak{P}, \mathfrak{D}\big)$; $\mathfrak{P}$ denotes the (finite) set of partial solutions that have already been constructed at the current point in the execution process; $\mathfrak{D}$ denotes the currently discovered part of the queried Web of Linked Data. As discussed before, we initialize $\mathfrak{P}$ with the empty partial solution σ_0 (cf. Section 6.2) and $\mathfrak{D}$ with $\mathfrak{D}_{\mathsf{init}}^{S,W}$ (cf. Section 6.1). During the query execution process $\mathfrak{P}$ and $\mathfrak{D}$ grow monotonically: We augment partial solutions from $\mathfrak{P}$ and add the results back to $\mathfrak{P}$; additionally, we use partial solutions from $\mathfrak{P}$ to expand $\mathfrak{D}$. However, conceptually we combine these two types of tasks, augmentation and expansion, into a single type:

Definition 19. Let $\mathcal{Q}_{c_{\mathsf{Match}}}^{B,S}$ be a CLD query (under c_{Match}-semantics); let $\big(\mathfrak{P}, \mathfrak{D}\big)$ represent a state of a (link traversal based) execution of $\mathcal{Q}_{c_{\mathsf{Match}}}^{B,S}$. An **AE task for** $\big(\mathfrak{P}, \mathfrak{D}\big)$ is a tuple (σ, t, tp) for which it holds i) $\sigma = (P, \mu) \in \mathfrak{P}$, ii) $t \in \mathrm{AllData}\big(\mathfrak{D}\big)$, iii) $tp \in B \setminus P$, and iv) t matches tp .

Performing an AE task (σ, t, tp) for $\big(\mathfrak{P}, \mathfrak{D}\big)$ comprises two steps: 1.) changing $\mathfrak{P}$ to $\mathfrak{P} \cup \{(P', \mu')\}$, where $(P', \mu') = aug_{t,tp}^{\mathfrak{D}}\big(\sigma\big)$ is the (t, tp)-augmentation of σ in $\mathfrak{D}$, and 2.) expanding $\mathfrak{D}$ to the μ'-expansion of $\mathfrak{D}$ in W. Notice, constructing the augmentation in the first step is always possible because the prerequisites for AE tasks, as given in Definition 19, correspond to the prerequisites for augmentations (cf. Definition 17). However, not all possible AE tasks may actually change $\mathfrak{P}$ and $\mathfrak{D}$; instead, some tasks (σ, t, tp) may produce an augmentation $aug_{t,tp}^{\mathfrak{D}}\big(\sigma\big)$ that turns out to be a partial solution which has already been produced for another task. Thus, to guarantee progress during a query execution process we must only perform those AE tasks that produce new augmentations. To identify such tasks we introduce the concept of *open AE tasks*.

Definition 20. An AE task (σ, t, tp) for the state $(\mathfrak{P}, \mathfrak{D})$ of a link traversal based query execution is **open** if $aug_{t,tp}^{\mathfrak{D}}\big(\sigma\big) \notin \mathfrak{P}$. To denote the set of all open AE tasks for $(\mathfrak{P}, \mathfrak{D})$ we write $Open\big(\mathfrak{P}, \mathfrak{D}\big)$.

We now use the introduced concepts to present our abstract procedure $ltbExec$ (cf. Algorithm 1) with which we formalize the general idea of link traversal based query execution. After initializing $\mathfrak{P}$ and $\mathfrak{D}$ (lines 1 and 2 in Algorithm 1), the procedure amounts to a continuous execution of open AE tasks. We represent this continuous process by a loop (lines 3 to 9); each iteration of this loop performs an open AE task (lines 5 to 7) and checks whether the newly constructed partial solution $\big(P', \mu'\big)$ covers the executed

Algorithm 1 $ltbExec(S, B, W)$ – Report all $\mu \in \mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}(W)$.

```
1: $\mathfrak{P} := \{\sigma_0\}$
2: $\mathfrak{D} := \mathfrak{D}^{S,W}_{\mathsf{init}}$
3: while $Open(\mathfrak{P}, \mathfrak{D}) \neq \varnothing$ do
4:     Choose open AE task $(\sigma, t, tp) \in Open(\mathfrak{P}, \mathfrak{D})$
5:     $(P', \mu') := aug^{\mathfrak{D}}_{t,tp}(\sigma)$
6:     $\mathfrak{P} := \mathfrak{P} \cup \{(P', \mu')\}$        // indirectly changes $Open(\mathfrak{P}, \mathfrak{D})$
7:     $\mathfrak{D} := exp^{W}_{\mu'}(\mathfrak{D})$
8:     if $P' = B$ then report $\mu'$ endif
9: end while
```

CLD query as a whole, in which case the valuation μ' in (P', μ') must be reported as a solution for the query (line 8). We emphasize that the set $Open(\mathfrak{P}, \mathfrak{D})$ of all open AE tasks always changes when $ltbExec$ performs such a task. The loop terminates when no more open AE tasks for (the current) $(\mathfrak{P}, \mathfrak{D})$ exist (which may never be the case as we know from Lemma 1).

We emphasize the abstract nature of Algorithm 1. The fact that we model $ltbExec$ as a single loop which performs (open) AE tasks sequentially, does not imply that the link traversal based query execution paradigm has to be implemented in such a form. Instead, different implementation approaches are possible, some of which have already been proposed in the literature [6, 7, 13, 14]. In contrast to the concrete (implementable) algorithms discussed in this earlier work, we understand Algorithm 1 as an instrument for presenting and for studying the general idea that is common to all link traversal based query execution approaches.

6.5 Application of the Model

Based on our query execution model we now show that the idea of link traversal based query execution is sound and complete, that is, the set of all valuations reported by $ltbExec(S, B, W)$ is equivalent to the query result $\mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}(W)$. Formally:

Theorem 4. *Let W be a Web of Linked Data and let $\mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}$ be a CLD query (under c_{Match}-semantics).*

- *Soundness: For any valuation μ reported by an execution of $ltbExec(S, B, W)$ holds $\mu \in \mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}(W)$.*
- *Completeness: Any $\mu \in \mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}(W)$ will eventually be reported by any execution of $ltbExec(S, B, W)$.*

Theorem 4 formally verifies the applicability of link traversal based query execution for answering conjunctive queries over a Web of Linked Data. For experimental evaluations that demonstrate the feasibility of link traversal based execution of queries over Linked Data on the WWW we refer to [6, 7, 13, 14]. We note, however, that the implementation approaches used for these evaluations do not allow for an explicit specification of seed identifiers S. Instead, these approaches use the identifiers in the BQP of a query as seed and, thus, only support CLD queries $\mathcal{Q}^{B,S}_{c}$ for which $S = \mathrm{ids}(B)$. Theorem 4 highlights that this is a limitation of these particular implementation approaches and not a general property of link traversal based query execution.

In the remainder of this section we use our (abstract) execution model to analyze the iterator based implementation of link traversal based query execution that we introduce in [6, 7]. The analysis of this implementation approach is particularly interesting because this approach trades completeness of query results for the guarantee that all query executions terminate as we shall see.

The implementation approach applies a synchronized pipeline of operators that evaluate the BQP $B = \{tp_1, \dots, tp_n\}$ of a CLD query in a fixed order. This pipeline is implemented as a chain of iterators $I_1, \dots, I_n$; iterator I_k is responsible for triple pattern tp_k (for all $1 \leq k \leq n$) from the *ordered* BQP. While the selection of an order for the BQP is an optimization problem [6], we assume a given order for the following analysis (in fact, the order is irrelevant for the analysis). Each iterator I_k provides valuations that are solutions for CLD query $\mathcal{Q}^{P_k,S}_{c_{\mathsf{Match}}}$ where $P_k = \{tp_1, \dots, tp_k\}$. To determine these solutions each iterator I_k executes the following four steps repetitively: First, I_k consumes a valuation μ' from its direct predecessor and applies this valuation to its triple pattern tp_k, resulting in a triple pattern $tp'_k = \mu'[tp_k]$; second, I_k (tries to) generate solutions by finding matching triples for tp'_k in the query-local dataset; third, I_k uses the generated solutions to expand the query-local dataset; and, fourth, I_k (iteratively) reports each of the generated solutions. For a more detailed description of this implementation approach we refer to [6].

In terms of our abstract execution model, each iterator performs a particular subset of all possible open AE tasks: For each open AE task (σ, t, tp) performed by iterator I_k it holds i) $tp = tp_k$ and ii) $\sigma = (P_{k-1}, \mu)$ where $P_{k-1} = \{tp_1, \dots, tp_{k-1}\}$. However, I_k may not perform all (open) AE tasks which have these properties.

Lemma 2. *During an iterator execution of an arbitrary CLD query $\mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}$ (that uses c_{Match}) over an arbitrary Web of Linked Data W it holds: The set of AE tasks performed by each iterator is finite.*

Based on Lemma 2 we easily see that an iterator execution of a CLD query $\mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}$ may not perform all possible (open) AE tasks. Thus, we may show the following result as a corollary of Lemma 2.

Theorem 5. *Any iterator based execution of a CLD query $\mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}$ (that uses c_{Match}) over an arbitrary Web of Linked Data W reports a finite subset of $\mathcal{Q}^{B,S}_{c_{\mathsf{Match}}}(W)$ and terminates.*

Theorem 5 shows that the analyzed implementation of link traversal based query execution trades completeness of query results for the guarantee that all query executions terminate. The degree to which the reported subset of a query result is complete depends on the order selected for the BQP of the executed query as our experiments in [6] show. A formal analysis of this dependency is part of our future work.

7. RELATED WORK

Since its emergence the World Wide Web has spawned research to adapt declarative query languages for retrieval of information from the WWW [4]. Most of these works understand the WWW as a graph of objects interconnected by hypertext links; in some models objects have certain attributes (e.g. title, modification date) [15] or an internal structure [5, 12]. Query languages studied in this context allow a user to either ask for specific objects [12], for their attributes [15], or for specific object content [5]. However, there is no explicit connection between data that may be obtained from different objects (in contrast to the more recent idea of Linked Data). Nonetheless, some of the foundational work such as [1] and [15] can be adapted to query execution over a Web of Linked Data. In this paper we analyze the computability of CLD queries by adopting Abiteboul and Vianu's notions of computability [1], for which we have to adapt their machine model of computation on the Web.

In addition to the early work on Web queries, query execution over Linked Data on the WWW has attracted much attention recently. In [9] we provide an overview of different approaches and refer to the relevant literature. However, the only work we are aware of that formally captures the concept of Linked Data and provides a well-defined semantics for queries in this context is Bouquet et al. [3]. In contrast to our more abstract, technology-independent data model, their focus is Linked Data on the WWW, implemented using concrete technologies such as URIs and RDF. They adopt the common understanding of a set of RDF triples as graphs [11]. Consequently, Bouquet et al. model a Web of Linked Data as a "graph space", that is, a set of RDF graphs, each of which is associated with a URI that, when dereferenced on the WWW, allows a system to obtain that graph. Hence, RDF graphs in Bouquet et al.'s graph space correspond to the LD documents in our data model; the URIs associated with RDF graphs in a graph space have a role similar to that of those identifiers in our data model for which the corresponding mapping $adoc$ returns an actual LD document (i.e. all identifiers in $\mathrm{dom}(adoc)$). Therefore, RDF graphs in a graph space form another type of (higher level) graph, similar to the Web link graph in our model (although, Bouquet et al. do not define that graph explicitly). Based on their data model, Bouquet et al. define three types of query methods for conjunctive queries: a bounded method which only uses those RDF graphs that are referred to in queries, a navigational method which corresponds to our query model, and a direct access method which assumes an oracle that provides all RDF graphs which are "relevant" for a given query. For the navigational method the authors define a notion of reachability that allows a query execution system to follow all data links. Hence, the semantics of queries using this navigational method is equivalent to CLD queries under c_{All}-semantics in our query model. 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 the computability of queries and the infiniteness of the WWW.

8. CONCLUSIONS AND FURTHER WORK

Link traversal based query execution is a novel query execution approach tailored to the Web of Linked Data. The ability to discover data from unknown sources is its most distinguishing advantage over traditional query execution paradigms which assume a fixed set of potentially relevant data sources beforehand. In this paper we provide a formal foundation for this new approach.

We introduce a family of well-defined semantics for conjunctive Linked Data queries, taking into account the limited data access capabilities that are typical for the WWW. We show that the execution of such queries may not terminate (cf. Theorem 2) because –due to the existence of data generating servers– the WWW is infinite (at any point in time). Moreover, queries may have a result that is infinitely large. We show that it is impossible to provide an algorithm for deciding whether any given query (in our model) has a finite result (cf. Theorem 1). Furthermore, it is also impossible to decide (in general) whether a query execution terminates (cf. Theorem 3), even if the expected result would be known to be finite.

In addition to our query model we introduce an execution model that formally captures the link traversal based query execution paradigm. This model abstracts from any particular approach to implement this paradigm. Based on this model we prove that the general idea of link traversal based query execution is sound and complete for conjunctive Linked Data queries (cf. Theorem 4).

Our future work focuses on more expressive types of Linked Data queries. In particular, we aim to study which other features of query languages such as SPARQL are feasible in the context of querying a Web of Linked Data and what the implications of supporting such features are. Moreover, we will extend our models to capture the dynamic nature of the Web and, thus, to study the implications of changes in data sources during the execution of a query.

9. ACKNOWLEDGEMENTS

We thank our colleague Matthias Sax for a fruitful discussion that led to one of the proofs in this paper.

10. REFERENCES

[1] S. Abiteboul and V. Vianu. Queries and computation on the Web. *Theoretical Computer Science*, 239(2), 2000.

[2] S. Auer, J. Lehmann, and S. Hellmann. LinkedGeoData – adding a spatial dimension to the Web of Data. In *Proc. of the 8th Int. Semantic Web Conference (ISWC)*, 2009.

[3] P. Bouquet, C. Ghidini, and L. Serafini. Querying the Web of Data: A formal approach. In *Proc of the 4th Asian Semantic Web Conference (ASWC)*, 2009.

[4] D. Florescu, A. Y. Levy, and A. O. Mendelzon. Database techniques for the world-wide Web: A survey. *SIGMOD Record*, 27(3), 1998.

[5] T. Guan, M. Liu, and L. V. Saxton. Structure-based queries over the world wide Web. In *Proc. of the 17th Int. Conference on Conceptual Modeling (ER)*, 1998.

[6] O. Hartig. Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In *Proc. of the 8th Ext. Semantic Web Conference (ESWC)*, 2011.

[7] O. Hartig, C. Bizer, and J.-C. Freytag. Executing SPARQL queries over the Web of Linked Data. In *Proc. of the 8th Int. Semantic Web Conference (ISWC)*, 2009.

[8] O. Hartig and J. C. Freytag. Foundations of traversal based query execution over Linked Data (extended version). *CoRR*, abs/1108.6328, 2011. Online: http://arxiv.org/abs/1108.6328.

[9] O. Hartig and A. Langegger. A database perspective on consuming Linked Data on the Web. *Datenbank-Spektrum*, 10(2), 2010.

[10] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition, 2011.

[11] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Rec., Online at http://www.w3.org/TR/rdf-concepts/, Feb. 2004.

[12] D. Konopnicki and O. Shmueli. W3qs: A query system for the world-wide Web. In *Proc. of 21th Int. Conference on Very Large Data Bases (VLDB)*, 1995.

[13] G. Ladwig and D. T. Tran. Linked Data query processing strategies. In *Proc. of the 9th Int. Semantic Web Conference (ISWC)*, 2010.

[14] G. Ladwig and D. T. Tran. SIHJoin: Querying remote and local linked data. In *Proc. of the 8th Ext. Semantic Web Conference (ESWC)*, 2011.

[15] A. O. Mendelzon and T. Milo. Formal models of Web queries. *Information Systems*, 23(8), 1998.

[16] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34, 2009.

[17] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C Rec., Online at http://www.w3.org/TR/rdf-sparql-query/, Jan. 2008.

[18] F. Schmedding. Incremental SPARQL evaluation for query answering on Linked Data. In *Proc. of the 2nd Int.Workshop on Consuming Linked Data (COLD) at ISWC*, 2011.

[19] D. Vrandecić, M. Krötzsch, S. Rudolph, and U. Lösch. Leveraging non-lexical knowledge for the linked open data web. In *RAFT*, 2010.