Programming the Semantic Web

Steffen Staab, Stefan Scheglmann, Martin Leinberger, and Thomas Gottron

Institute for Web Science and Technologies, University of Koblenz-Landau, Germany

Abstract. The Semantic Web changes the way we deal with data, because assumptions about the nature of the data that we deal with differ substantially from the ones in established database approaches. Semantic Web data is (i) provided by different people in an ad-hoc manner, (ii) distributed, (iii) semi-structured, (iv) (more or less) typed, (v) supposed to be used serendipitously. In fact, these are highly relevant assumptions and challenges, since they are frequently encountered in all kind of data-centric challenges also in cases where Semantic Web standards are not in use. However, they are only partially accounted for in existing programming approaches for Semantic Web data including (i) semantic search, (ii) graph programming, and (iii) traditional database programming approaches.

The main hypothesis of this talk is that we have not yet developed the right kind of programming paradigms to deal with the proper nature of Semantic Web data, because none of the mentioned approaches fully considers its characteristics. Thus, we want to outline empirical investigations of Semantic Web data and recent developments towards Semantic Web programming that target the reduction of the impedance mismatches between data engineering and programming approaches.

1 Introduction

The way data is published on the Semantic Web poses a challenge when making use of this data—in particular in the context of programming with Semantic Web data. The flexibility of distributed data providers to freely choose an individual schema layout and to use an appropriate mix of vocabularies causes the nature of the data to be substantially different from data obtained, e.g., from relational databases. For instance, data consumers and programmers do not know apriori how semantic vocabularies are actually used to describe data, which schematic layout underlies the data and where the data resides on the Web. This lack of knowledge affects three levels of interaction with data on the Semantic Web: (1) how programmers *select* the data they are interested in, (2) how they *fetch* and retrieve the data they incorporate in their own programs and (3) which concepts of programming methods they use for an *idiosyncratic programming* with the obtained data.

The *selection* of data is different from classical settings of programming against a known backend data store. Programmers cannot rely on a pre-defined and given schema of the data nor on any global convention of how data is modelled and published. Schematic patterns in the data are rather emerging properties of the Semantic Web. These patterns are a hybrid phenomenon which is caused by recommendations and best practices (e.g. Linked Data guidelines), social processes (e.g. reusing popular

V. Presutti et al. (Eds.): ESWC 2014, LNCS 8465, pp. 1–5, 2014.

vocabularies) and technical factors (e.g. common data conversion tools using similar approaches for representing relational data). Thus, the challenge in selecting the appropriate data pertains to the problem of how to describe the data which is needed in a particular application context.

Fetching data from the Semantic Web is also a non standardised process. A multitude of options is available on the Web, they range from browsing Linked Data over querying SPARQL endpoints to downloading bulk data files, using semantic search engines or even accessing proprietary programmable interfaces. These options do not only affect the way data is requested, but also the granularity, precision and volume of data which is retrieved and needs to be handled and managed.

The question of *idiosyncratic processing* of Semantic Web data is probably the least addressed so far. The issue here is that for selecting and fetching of data, one may consider generic approaches, but for arbitrary processing of data, one needs a paradigm that allows for ideosyncractic code to handle Semantic Web data. And this code should be easy to write, easy to maintain and deal with the characteristics that are common to Semantic Web data described above. For instance, programming Semantic Web data as triples is possible, but it does not ensure *any* consistency with regard to data types by the programming paradigm itself.

In this keynote we postulate, that we will need to challenge existing views of how to program the Semantic Web. Rather than a one-solution-fits all approach, we might actually need to look into a toolbox of methods which fit the programmer's need of how to select, fetch and program with semantic data. We will need to identify patterns of how to combine paradigms from the three levels in a suitable way. To underline our statement we will motivate and explain the challenges on each level, provide an overview of different processes and give examples for generic methods which could be assets in this toolbox and can be combined for supporting programmers in their specific application scenarios.

2 Selecting Data

The challenge in selecting data is twofold. On the one hand, it implies the choice of how to find and address the data. The range of possible choices comprises approaches for finding and selecting data via *structured query languages*, *semantic search* or *browsing* the data graph on the Web. Once the method for finding data has been chosen there is, on the other hand, still a challenge in identifying a suitable, concise and appropriate description. In analogy to the approaches for finding data, the description can be a declarative description, a key word based query or a traversal path in a graph of linked data items.

The choice of the paradigm for finding data might be influenced or even dictated by the use case setting or application requirements. However, the choice of the approach has an impact of how flexible and suitable it is to address certain characteristics of Semantic Web data. This impact is outlined in Table 1. Structured query languages are prone to difficulties in selecting data from distributed data sources. Approaches for query federation require information about the data sources as well as end points capable of answering the structured queries themselves. This does not fit well the ad-hoc manner provision of semi-structured data. Also semantic search has to cope with

Table 1. Approaches for selecting data from the Semantic Web

	Structured Query Languages	Semantic Search	Browsing
(i) ad-hoc manner	(√)	(√)	–
(ii) distributed sources	–	(√)	√
(iii) semi-structured	(√)	√	√
(iv) (more or less) typed	√	√	√
(v) used serendipitously	–	√	√

similar problems. The reason, however, is that semantic search needs to build index structures over semantic web data, which need to be maintained and updated. Finally, identifying a browsing path to a data source is less suitable for the ad-hoc provision of the data as novel data sources might not be well connected and are difficult to come across.

The challenge of finding a suitable, concise and appropriate description is mainly related to structured query languages and semantic search. The reasons for this challenge is that it does not only involve the program's need for data but also the way the data providers have published their data. This means, we need to enable the programmers to create a match between their need and the description of available data. A common approach for enabling this match is to analyze data on the Semantic Web for the purpose of detecting patterns and agreements on how data is modeled and published. One such observable pattern is the schema of Semantic Web data. It is not predefined but emerges on the basis of how data appears on the Web. In this context, we think of schema information in two forms. On the one hand, the schema is given by RDF types associated with the modeled entities. On the other hand, the schema is also described by the properties of entities, i.e. the RDF predicates used to interlink entities. To render an emerging schema into an explicit form, it needs to be induced from the data. Various approaches have been investigated in this direction: statistical schema induction [5], explicit description [1] and structural analysis [4]. Questions such as "Which RDF types are commonly appearing together?", "Which properties are typically used in the context of given types?" and "Which kind of entities can I expect to find when looking at the objects in the range of certain predicates?" can directly be answered on a schema level. Recently, automated methods have been developed to help and support a programmer in this explorative process [2,3].

3 Fetching Data

Ways to handle and manage the data are just as many as to access the data. Approaches range from downloading readily packaged *bulk data* over *querying* an index to *dereferencing URIs* and referring to the data providers for the original data.

A bulk download of data is hardly possible if many distributed data sources are addressed. The problem is that this option might simply not be available. Furthermore, ad-hoc updates and modifications of the data need to be tracked and might not be reflected

Table 2. Approaches for fetching data from the Semantic Web

	Bulk Download	Querying	Dereferencing
(i) ad-hoc manner	(√)	(√)	√
(ii) distributed sources	–	–	√
(iii) semi-structured	√	√	√
(iv) (more or less) typed	√	√	√
(v) used serendipitously	√	–	(√)

Table 3. Approaches for programming with data from the Semantic Web

	Graph-Centric	Triple-Centric	Schema-Centric
(i) ad-hoc manner	√	√	–
(ii) distributed sources	(√)	(√)	–
(iii) semi structured	√	√	–
(iv) (more or less) typed	√	√	–
(v) used serendipitously	√	√	√

in a bulk download. The challenge of federating queries has already been mentioned before and obviously extends also to fetching data from distributed data sources. Moreover, the need to precisely describe the requested data when querying a SPARQL endpoint does not suit the idea of serendipitous use. Finally, also endpoints for querying are not omnipresent in the Semantic Web. Thus, also this option is not available for all data sources. Dereferencing URIs and fetching live data from the Semantic Web, instead, is less susceptible to these problems. Looking up live data from online sources does not pose challenges with outdated information and is by definition capable of dealing with distributed data sources. This mode of fetching data is also entirely independent of the structure or typing of the data. However, the serendipitous use of data is limited as the URIs of data sources need to be known a-priori.

4 Idiosyncratic Programming

While we discussed the selection and fetching of data so far, programming against such data also means to manipulate it. This involves changing of existing properties or even the creation of completely new instances. There are several different approaches to an APIs that allows such a manipulation identifiable. Tables 3 tries to give an overview over these.

Low level RDF APIs mostly use a **graph-centric** or **triple-centric** data access model. In a graph-centric approach, the data is provided as nodes (subjects, objects) and edges (predicates). The slightly more specific triple-centric approach provides the data as subject-predicate-object triples (or n-tuples). Both approaches have the advantage that they can cope with the whole flexibility of the RDF data model, such as semi-structured

data the ad-hoc manner. Since they build on the atomic entities of the RDF model, they are also robust against changes in the data as well as source and type independent. Tackling the distribution of data sources needs to be implemented in the according API, but it is generally feasible.

However, for more sophisticated applications it is tenacious to deal with data on such a low level. On top of low-level APIs several object persistence APIs exist. Most of them apply mappings similar to object-relational mappings. Such a **schema-centric** manner allows us to access data on type/entity-level, but the object-triple mapping is dependent on schematic information. Due to its reliance on a schema it is challenging to facilitate the ad-hoc manner, the semi-structuredness and the inconsistent use of types in Linked Data. Also, the handling of distributed data is only possible if a global schema exists.

5 Conclusion

The Semantic Web may change the way we deal with data, but we still don't have an approach that fully considers its characteristics. While there are techniques that deal with the three aspects - selecting, fetching and programming - individually, we haven't yet found a perfect pattern to combine the three. All of the existing approaches fall short in some areas. However, before the Semantic Web can reach its full potential, we need to deal with this impedance mismatch.

Acknowledgements. This work has been supported by Microsoft. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013), REVEAL (Grant agree number 610928).

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets with the void vocabulary, `http://www.w3.org/TR/void/` (last visited August 30, 2011)
2. Gottron, T., Scherp, A., Krayer, B., Peters, A.: LODatio: Using a Schema-Based Index to Support Users in Finding Relevant Sources of Linked Data. In: K-CAP 2013: Proceedings of the Conference on Knowledge Capture, pp. 105–108 (2013)
3. Gottron, T., Scherp, A., Scheglmann, S.: Providing alternative declarative descriptions for entity sets using parallel concept lattices. In: Presutti, V., d'Amato, C., Gandon, F. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 362–376. Springer, Heidelberg (2014)
4. Konrath, M., Gottron, T., Staab, S., Scherp, A.: SchemEX—Efficient Construction of a Data Catalogue by Stream-based Indexing of Linked Data. Web Semantics: Science, Services and Agents on the World Wide Web 16(5), 52–58 (2011)
5. Völker, J., Niepert, M.: Statistical schema induction. 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. 124–138. Springer, Heidelberg (2011)