One Size Doesn't Fit All
Fostering Diversity on the Semantic Web

Wouter Beek[1], Ruben Verborgh[2], Christophe Guéret[3], and Miel Vander Sande[2]

[1] Vrije Universiteit Amsterdam, The Netherlands
w.g.j.beek@vu.nl
[2] Ghent University – iMinds, Belgium
{ruben.verborgh,miel.vandersande}@ugent.be
[3] DANS, Royal Dutch Academy of Sciences, The Netherlands
christophe.gueret@dans.knaw.nl

Abstract. The current Semantic Web landscape is shaped by a collection of standards and practices, the philosophy of which is strongly rooted in its ancestor domains of databases and artificial intelligence. All too often, we seem to assume that the default solutions are sufficient for the majority of use cases, but are they really? While practice shows that, for instance, installing a SPARQL endpoint over a public dataset is not the definitive solution to offer reliable data access, the community still seems to assume issues will disappear by improving existing solutions instead of developing new ones. With nothing but the standards to fall back to, we need to think about alternative solutions—and how to start building these. This paper discusses cases in which the current one-size-fits-all approach fails, suggesting possible alternative directions. Embracing the diversity of the Web, rather than trying to retrofit things to a perhaps idealistic model, could be the more elegant way forward.

Keywords: Semantic Web, Linked Data, standards, diversity, Web

1 Introduction

Already from the start, the Web was designed to harness scalability and heterogeneity. It's fundamentals are based on the notion of hypertext [1]. In the original proposal, Tim Berners-Lee and Robert Cailliau stated *(emphasis added)*:

> "HyperText is a way to **link and access information of various kinds** as a web of nodes in which the user can browse at will. Potentially, HyperText provides a **single user-interface** to many large classes of stored information..."

Today, the existence of the Web proves that they were not mistaken. The gigantic expansion of the Web in the last 25 years, was made possible by the two original hypertext technologies: the *uniform interface* HTTP, and the format to *link and access information of various kinds* HTML.

The HTTP protocol allows the Web to be a *scale-free* network. It only defines a very high-level interface, as each Web server can freely choose how it maps its URL space to resources. This resource-oriented approach allows new information sources to seamlessly integrate in the Web. While they started by serving static files, their functionality fits also dynamic content and services, e.g., Web APIs.

HTTP offers a uniform interface to content, but does not constraint how this is structured. That makes data format heterogeneity common on the Web. This was initiated with the HTML language, which has very limited structural constraints, that allow infinite possible ways of shaping information into a page. Next, as the Web progressed, content was increasingly being consumed by applications, requiring more structured content, e.g., XML, and later JSON.

Currently, the Semantic Web is driven by two technologies as well: RDF and SPARQL. However, looking from the hypertext perspective at the Semantic Web, it did not inherit its flexibility on all levels, since machine-interpretable information requires a more rigid structure than content designed for human consumption. Therefore, the strictly triple-based structure of RDF is a necessity; and indeed, we can still freely choose the semantics we use to express information.

However, many other aspects are rigidly arranged in similar ways, and this leads to less heterogeneity on the Semantic Web—perhaps unnecessarily.

In this paper, we discuss the dominating technologies on the Semantic Web and their tendency to develop monopolies where *one-size-fits-all*. We identify use cases where their appliance is counterproductive and slows down adoption.

The remainder of this paper is structured as follows. First, we posit that current technologies and approaches are restricting the diversity of the Semantic Web in Section 2, and describe possible alternatives in Section 3. Then, Section 4 concludes with lessons learned for future embracing of diversity.

2 Killing diversity

In this section we explain why we believe that today's Semantic Web is showing too little diversity, and why this is a bad thing. We first enumerate the three main reasons why diversity is restricted on the Semantic Web today: standardization, research focus, and deployment practice.

Standardization Standardization has been a crucial step in creating the Semantic Web. The standardized encoding of data in RDF, as well as the use of standardized universal identifiers (URIs and IRIs), has allowed data to be universally interchanged on an unprecedented scale [3]. But even the inherently multifarious acts of querying and reasoning are being standardized today. We claim that standardization efforts, although crucial to the Semantic Web as a whole, might at the same time have counterproductive effects on certain individual parts of its stack.

Research focus A large part of the Semantic Web research effort over the past decade has focused on improving the size, performance, and manageability properties of centralized triple stores, often in comparison or competition with

the relational paradigm. Even though this effort has produced important results, we claim that Semantic Web research should actively explore a wider gamut of possible approaches and properties that need improvement.

Deployment practice We also see that there is not enough variation in the way in which Linked Datasets are disseminated and appropriated in practice. Linked Data is often stored in a centralized data repository and queried via an SQL-inspired interaction paradigm. Even though some of these systems are somewhat usable today, we claim that a wider range of deployment techniques should be explored.

We argue that, despite the undisputed usefulness of standardization efforts, centralized triple store research, and deployment practices inspired by traditional databases, the Semantic Web today shows too little alternative approaches and, because of this, too little diversity. We believe that while standards and practices should be built on established research, the former should not be used in order to fixate research goals. When needed, Semantic Web research should be able to move beyond existing practices in previously unanticipated ways. In addition to following standardization efforts and established practices, Semantic Web research should be able to critically reflect on those as well, e.g., by identifying the ideological underpinnings of standardization.

We now enumerate some of the adverse effects of having too little diversity in Semantic Web standards, research, and practice today.

Cost The development of a standardized querying language for the Web in the form of SPARQL [10] reveals a set of assumptions that are not made explicit in the standards document itself. In SPARQL, the disseminating server is specified as the primary location where computation occurs, thereby turning clients into relatively simple workers. By shifting most of the computation in querying to the server-side, SPARQL deployment requires a relatively big hardware footprint. Specifically, the hardware footprint for serving Semantic Web data is significantly higher than what is needed to serve the same data file on the traditional Web. Due to the cost associated with hardware requirements, current standards pose a financial barrier to the Semantic Web. This is an important departure from the inclusivity that was expressed in the original Semantic Web vision. Since few data owners will have the financial means necessary to deploy SPARQL services, standardization favors the data centralization paradigm. Besides favoring centralization, standardization also favors the developed world, since the absence of a centralized infrastructure means that standards-conforming Semantic Web technologies are difficult to deploy in developing countries.

Availability Empirical data coming from Linked Data Observatories [4] confirms that relatively few datasets are actually being disseminated. Moreover, not all Linked Datasets that are disseminated are able to guarantee high up-time. Use of the SPARQL standard in practice leads to low availability.

Restrictiveness Continuing on the previous theme, empirical data [4] shows that those SPARQL endpoints which do have high availability, often do so by enforcing certain restrictions on the queries a client is allowed to ask, and the

number of answers that can be returned. Even though Semantic Web querying technologies focus on completeness of query results, deployment of those technologies in practice often severely restricts those completeness properties.

Security/Privacy The database-inspired and centralized deployment strategy for Linked Data dissemination has several drawbacks. Firstly, a single data store provides a single point of failure in the case of unanticipated circumstances. Secondly, the network infrastructure connecting those datastores is surprisingly sparse [7]. A more distributed data storage paradigm, with more redundancy in terms of data duplication and cross-database infrastructure, would make the Semantic Web more sustainable. Such a sparse network architecture is relatively prone to privacy invasion, since much of the data traffic goes through only a limited number of spots. In addition, since data of multiple users is stores at a single centralized server, a single security breach can potentially affect multiple users.

Ownership Another downside to having a centralized infrastructure is that data dissemination and data storage often go together. This means that, in practice, individual data providers are handing over their potentially private data to a few big disseminating entities, thereby handing over (part of) their agency and ownership of the data as well. In this respect, the Semantic Web is following a similar trend towards the centralized storage of personal data that we see in the traditional Web, where the majority of email and document synchronization is handled by only a few organizations.

Multiple viewpoints The motto of the Semantic Web is that "anyone can say everything about anything"[1]. Adherence to this motto means that different viewpoints about the same subject matter are allowed to be expressed. However, putting together such multi-perspective data into one spot makes it difficult to tell these different views apart. If opinionated data is instead stored at the location of the individual or organization that expresses those views, the existing authority infrastructure of the Web can encode a lot of information that otherwise has to be explicitly tracked by provenance.

Reasoning Centralized deployment of Semantic Web data has adverse effects for reasoning tasks. From a semantic point of view, storing multiple heterogeneous data sources into a centralized database system may lead to inconsistencies, even though large subsets of the data would have been consistent when stored in a separate location. Inconsistency on the Semantic Web is *explosive*, i.e. from an inconsistent dataset anything can be deduced. Existing approaches for isolating consistent subsets have high computational complexity [13]. Methods for finding consistent subsets of Semantic Web data can be improved upon by utilizing locality properties of the data, provided that data is meaningfully distributed across an interconnected network of dissemination locations.

[1] `http://www.w3.org/DesignIssues/RDFnot.html`

3 Alternatives

Knowledge Bases (KBs) are created and used by the individuals that interface with them. Since individuals may have different goals and ideas, KBs are by nature incoherent, contradictory, heterogeneous and biased for locally relevant content. State-of-the-art approaches for the design of knowledge systems have to date aimed at finding the least common denominator characteristics of knowledge sharing among a target group of individuals, in order to produce a coherent subset of data that can be queried and reasoned over. We counter these approaches by researching alternatives that embrace the inherent diversity and messiness of human-created knowledge, by providing alternatives for the common publishing, querying and reasoning tasks.

3.1 Alternative for publication

Entity-centric Knowledge Systems such as those used for the Web of Data, the Domain Name System (DNS) or the Digital Object Identifiers, typically associate descriptions, *i.e.* lists of property/value pairs, to uniquely identified entities. The entity data gets then stored in central (*e.g.*, LOD), hierarchical (*e.g.*, DNS) or distributed locations (*e.g.* HBase).

The Entity Registry System (ERS) [6] brings a different solution to the design of entity registries, featuring a totally decentralised system with loose binding among the nodes. There are three type of nodes defined in ERS. *Contributor* nodes self-publish part of the description of entities and use the content made available by other peers. Some of these nodes can be used as *Bridges* which do not publish content, but act as relays between nodes to enable exchanges that are asynchronous and that can reach across closed networks. Finally, *Aggregator* nodes can be introduced to collect content from the Bridges and provide a single point of access to collectively hosted data.

ERS illustrates approach for publishing entity data. The focus is set on preserving the locality of information, enabling multiple viewpoints, enhancing privacy and security, and lowering the costs. All this essentially by parting from the canon of centralised data hosting.

3.2 Alternatives for querying

This section tackles two different aspects of querying: *interfaces* that give the client access to the data to be queried, and *engines* that are used to process the data and get to an answer.

3.2.1 Interfaces

At the moment, there are three predominant types of HTTP interfaces for accessing RDF triples:

Data dumps are the most basic type of interface, in which one URL corresponds to a (compressed version of) a single file that contains all the triples of a certain dataset. As a result, server responses are rather large (typically gigabytes). In

order to evaluate queries over the dataset, clients have to download the entire dump and interpret it locally by either loading the data into a private triple store, or executing the query in a streaming way (if possible).

Linked Data documents contain data related to a specific subject and are accessible through the URI that denotes this subject. This means that information is available in a more granular way, so individual responses are smaller. Queries can be evaluated by linked-traversal-based querying [11]. However, because data is only available in a per-subject way, some types of queries are hard to answer. For instance, finding all subjects of a certain type (e.g., `{ ?s a foaf:Person. }`) within a dataset is hard, because there is no single document that contains this information. Simply dereferencing the URI `foaf:Person` will not help, because this will point to the ontological concept, not to a list in the target dataset.

SPARQL endpoints allow clients to request very specific queries over a dataset. They currently provide the most flexible HTTP interface to RDF, but this comes at a cost: the availability of public SPARQL endpoints is very low compared to other HTTP servers on the Web [4]. The drawback of servers providing very unrestrictive interfaces to clients it that they risk performing arbitrarily much work to complete each individual request.

In the past, these interfaces have been regarded as completely distinct from each other. This is why Linked Data Fragments [16] were introduced as a uniform way to discuss *all* HTTP interfaces to RDF triples. At their core, each of the above interfaces provide *fragments* of a dataset, their main difference being how the selection of member triples of their fragments happens. A data dump contains all triples of a dataset; a Linked Data document contains all triples related to a specific subject; a result of a SPARQL `CONSTRUCT` query contains all triples that match this query. Such a view on existing interfaces allows to compare their features and trade-offs in a consistent way. Furthermore, it allows to define new kinds of fragments with a different balance of characteristics.

The current lack of diversity with regard to querying becomes apparent if we realize that the only approach to fully query live public data, i.e., through a SPARQL endpoint, only works reliably with a very expensive server infrastructure (see Section 2). This means that parties with valuable open data, but a limited budget, cannot provide queryable access to their data in a reliable way. The growth of the number of public SPARQL endpoints is indeed still very low for a technology that is considered part of the Semantic Web's core infrastructure. Perhaps we need to take a step back from the idea that we need to be able to solve *everything* already; because at the moment, the low reliability of SPARQL endpoints often prevents us from solving *anything*.

One of the new interfaces defined with Linked Data Fragments are *triple pattern fragments* [16]. They were designed with the observation in mind that selecting triples that match a certain triple pattern is a straightforward operation on the server side because of the way indexes are built up. This means that a server can host triple pattern fragments of a dataset at low cost without risking

low availability. At the same time, triple pattern fragments enable clients to easily evaluate basic graph patterns of SPARQL queries.

In contrast to Linked Data documents, predicate- and object-based queries are possible. Furthermore, since each page of a triple pattern fragment contains an estimate of the total number of matches, clients can evaluate basic graph patterns in an optimal order. The crucial difference with SPARQL endpoints is that servers themselves are not *intelligent*, i.e., sufficiently advanced to answer complex queries. Instead, they *enable* clients to exhibit intelligent behavior, by offering (only) the data and metadata that allow more complex tasks to be performed, such as evaluating SPARQL queries client-side. The idea behind this is generalizable to other interfaces that can be realized with Linked Data Fragments. That way, we can define new interfaces, and thus arrive at a continuum of possibilities rather than the three options currently considered by the Semantic Web community. This helps us to counter the one-size-fits-all practice of SPARQL endpoints, which are not a good match for environments where a high availability/cost ratio has to be maintained.

3.2.2 Engines

Currently, the approach taken to solve a query expressed over RDF data is directly inspired and derived from the work done in logics and relational databases. Queries are expressed as a conjunction of statement patterns. Those patterns are individually evaluated against the data set to get to lists of partial results which get joined using shared variables. Query planning takes an important part in this process to ensure the join operations are performed in an optimal way.

The query engine of triple pattern fragments is no exception to this, the main difference with other SPARQL engines is in the interface to the data. Therefore, eRDF [8, 9] proposes an alternative approach based on techniques from Computational Intelligence. In contrast, the query engine of eRDF does not rely on the resolution mechanism to provide answers to queries. The "e" stands for "evolutionary algorithm", meaning that eRDF uses such a computational approach to effectively *guess* the correct answers instead of constructing them. Candidate query solutions are randomly generated and validated against the dataset, the best ones are combined and slightly modified until a solution that can not be further improved is found. This solution is then returned and the search is resumed at another location of the search space.

Looking back at the challenges highlighted in Section 2, we can observe that, taking a direction that is clearly out of the commonly explored path, eRDF was the first system to propose any-time responses, streaming the results as they were found. It was also the first engine that is able to deal with very large and very complex queries, as its approach makes query planning unnecessary [8].

3.3 Alternatives for reasoning

As with classical, non-paraconsistent logics, inconsistency on the Semantic Web is *explosive*. This means that whenever two statements in an RDF database contradict each other, every statement that is expressible in the language can

be deduced from that database. This effectively means that the entire database becomes unusable from a reasoning perspective. Existing approaches are able to search for maximally consistent subsets [13], but they have the following problems:

1. Which maximally consistent subset is most useful for reasoning may depend on the application at hand.
2. For many real-world applications, the consistent subset does not have to be maximal, but only has to be large enough to be able to make certain deductions.
3. Existing methods have computational characteristics that make them useless for very big datasets.

DataHives[2] uses an agent paradigm in order to perform locality-induced consistency checking for reasoning tasks. This means that deduction is no longer carried out over the entire database *all at once*, but over a steadily increasing subset of the database for which consistency can be guaranteed. The initial subset that is chosen is application-specific, and the incremental addition of new statements is based on the structural properties of a graph representation of the LOD could. Agents in DataHives perform graph traversal according to different navigation strategies (e.g., random walk, ant swarm, bee swarm). Reasoning tasks in DataHives exhibit anytime behavior, since the traversed subset is always consistent and monotonically increasing. However, as a consequence, subsets need not be maximal.

As an example, we describe a search application in which bee agents are programmed to search for painters that are defined in the LOD cloud. Leaving implementation-specific details aside, the consistent subgraph for this application can be defined as the following graph pattern, closed under RDF(S) and OWL entailment: $\langle$`?x`, `a`, `yago:Painter110391653`$\rangle$. A swarm of bee agents initially consists of scouts that search for data sources. The search for data sources is performed by using resource dereferencing, queries on SPARQL endpoint, datasource descriptions (e.g., VoID), and datadumps, while looking for triples that contain the term `yago:Painter110391653`. Once a scout finds such a triple, it spawns a large number of forager agents that traverse the locality surrounding that triple, by following those edges of the LOD cloud graph that have relevant schema properties. While the scouts search for additional data sources, the foragers report triples that match the original search description back to the consistent repository of intermediate results, called the *data hive*. The data hive only accepts matches that do not produce a contradictory state. Scout traversal in a specific region lasts until no novel matches are being found in that region anymore. The hive is continuously being materialized in order to reflect all logical consequences, specifically those identifying resources that denote painters, i.e. the actual search results.

Contrary to existing approaches in Semantic Web reasoning, applications that use DataHives do not depend on a curated triple store for which semantic

[2] `https://github.com/wouterbeek/DataHives`

consistency is ensured by a preliminary data selection and integration process. Consistency of results is guaranteed, but completeness (which is meaningless for contradictory data anyway) and maximality of the result set are not guaranteed. DataHives is able to handle the dynamics of data on the Semantic Web, by using an aging operation for data that is stored in the hive. DataHives delivers anytime results for further processing by an application. As such, termination is application-specific as well, indicating when a sufficiently large result set has been produced.

4 Conclusions

The Semantic Web has not yet reached the same level of diversity as the Web. Therefore, this paper analyzed causes of the homogeneity of the current Semantic Web landscape, and zoomed in on several exemplary solutions for publishing, querying, and reasoning.

We do not only aim to promote the development of *more* approaches, but in particular *different* ones. To that end, Semantic Web researchers and practitioners will have to reassess their expectations and evaluations of what properties a good "Semantic Web solution" should have. As an example, current research on triple stores focuses on storage scalability and query performance, i.e., how we can query large amounts of data with maximal speed. Several benchmarks to assess the performance of SPARQL endpoints, with varying numbers of triples, exist to verify this. As a result, we do have high-performance SPARQL endpoints. However, no benchmark to date examines the availability/cost perspective of triple stores. Consequently, these high-performance SPARQL endpoints suffer from low availability rates. This example indicates how today's state of the art is a product of our expectations and evaluations that aim to verify those. Yet we have to question the validity of our assumptions, and more specifically, to find whether the contexts in which they hold are still relevant for today's challenges.

The Semantic Web is also strongly influenced by the background of its early adopters, namely the artificial intelligence and database communities. Because of this, as strange as it may sound, the "Web" part of "Semantic Web" is not as present as one might expect from the name. We would like to see Semantic Web approaches evolve to be more "webby", i.e., utilizing the strengths of the global network infrastructure of the Web, as opposed to the strengths of database management systems. After all, the initial Semantic Web vision [2] relied strongly on intelligent *clients*, yet the majority of current work so far has only resulted in intelligent *servers*. The Linked Data principles [3] have surely given a new impulse to a more Web-oriented way of thinking, yet the situation seems to have stagnated somewhat after the initial years. If we want to escape the "local maximum" [12] in which we seem to be stuck, we should actively explore other options.

Of course, some negative lessons are also to be learned from the Web. As a consequence of the high-level interface offered by HTTP, developers of Web applications provide their own APIs. Even APIs that serve very similar tasks have

vastly different designs and are not interoperable (for instance, status updates on social networks such as Twitter, Facebook, Google+ and others). As such, a client designed for one API cannot work with the other, regardless of the fact that the tasks are highly similar. In the world of Linked Data, such disparity would be unmanageable. Part of the solution from the Semantic Web angle is (again) standardization, for instance, with the Linked Data Platform standard [15] for read/write Linked Data. However, this defines only one specific API and will not cover the needs of all use cases, again illustrating the tension between genericness and expressivity. Dynamically discoverable APIs, for instance, driven by hypermedia and described using a hypermedia vocabulary [14], can help to change this if clients are able to discover which of the supported features they know how to interact with.

This brings us at the final point that, ultimately, we *will* have to deal with the heterogeneity of the Web. To what extent can we after all demand that existing publishers conform to a set of rules and conventions we as a sub-community created and need? It is hard to imagine that often-visited websites, such as Facebook, Amazon, eBay, and others, will suddenly offer a SPARQL endpoint and correct RDFa annotations for all of their data. Maybe instead of fighting diversity, we should be open to embrace it. Instead of trying to fit everything in the same interface, we should find ways to deal with the abundance of HTML and JSON interfaces on the Web—because if you think about it, each and every website is in fact a unique API. Solutions that help clients make sense of non-RDF data, such as API2LOD[3] or RML [5] essentially work the other way round and might perhaps be an alternative to triplifying everything.

It is thus likely that the assumptions on which we have been building the Semantic Web so far need to be tested again. We should be prepared to open the community for different viewpoints and solution types, as no size will ever fit all.

Acknowledgments

The described research activities were funded by Ghent University, the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research Flanders (FWO Flanders), and the European Union.

References

1. Berners-Lee, T., Cailliau, R.: WorldWideWeb: Proposal for a hypertext project. Tech. rep., CERN (1990), http://www.w3.org/Proposal
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–43 (May 2001), http://www.scientificamerican.com/article.cfm?id=the-semantic-web
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – the story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (Mar 2009)

[3] http://api2lod.appspot.com/

4. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.Y.: SPARQL web-querying infrastructure: Ready for action? In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) Proceedings of the 12th International Semantic Web Conference. Lecture Notes in Computer Science, vol. 8219, pp. 277–293. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-41338-4_18
5. Dimou, A., Vander Sande, M., Colpaert, P., De Vocht, L., Verborgh, R., Mannens, E., Van de Walle, R.: Extraction and semantic annotation of workshop proceedings in HTML using RML. In: Semantic Publishing Challenge of the 11th Extended Semantic Web Conference (May 2014)
6. Guéret, C., Cudré-mauroux, P.: The Entity Registry System: Publishing and Consuming Linked Data in Poorly Connected Environments (Apr 2014), http://ercim-news.ercim.eu/en97/ri/the-entity-registry-system-publishing-and-consuming-linked-data-in-poorly-connected-environments
7. Guéret, C., Groth, P., van Harmelen, F., Schlobach, S.: Finding the achilles heel of the web of data: using network analysis for link-recommendation. In: Proceedings of the International Semantic Web Conference 2010, LNCS, vol. 6496, pp. 289–304. Springer Berlin Heidelberg (2010)
8. Guéret, C., Groth, P., Oren, E., Schlobach, S.: eRDF: A scalable framework for querying the Web of Data. Tech. rep., Vrije Universiteit Amsterdam, Amsterdam (2011), http://dl.dropbox.com/u/2137510/erdf_technicalreport.pdf
9. Guéret, C., Groth, P., Schlobach, S.: eRDF: Live Discovery for the Web of Data (2009)
10. Harris, S., Seaborne, A.: SPARQL 1.1 query language. Recommendation, W3C (Mar 2013), http://www.w3.org/TR/sparql11-query/
11. Hartig, O.: An overview on execution strategies for Linked Data queries. Datenbank-Spektrum 13(2), 89–99 (2013)
12. Hogan, A., Gutierrez, C.: Paths towards the sustainable consumption of semantic data on the web. In: Gottlob, G., Pérez, J. (eds.) AMW. CEUR Workshop Proceedings, vol. 1189. CEUR-WS.org (2014)
13. Huang, Z., Van Harmelen, F., Ten Teije, A.: Reasoning with inconsistent ontologies. In: IJCAI. vol. 5, pp. 254–259 (2005)
14. Lanthaler, M., Gütl, C.: Hydra: A vocabulary for hypermedia-driven Web APIs. In: Proceedings of the 6th Workshop on Linked Data on the Web (May 2013), http://ceur-ws.org/Vol-996/papers/ldow2013-paper-03.pdf
15. Speicher, S., Arwe, J., Malhotra, A.: Linked Data Platform 1.0. Working draft, W3C (Mar 2014), http://www.w3.org/TR/2014/WD-ldp-20140311/
16. Verborgh, R., Hartig, O., De Meester, B., Haesendonck, G., De Vocht, L., Vander Sande, M., Cyganiak, R., Colpaert, P., Mannens, E., Van de Walle, R.: Querying datasets on the Web with high availability. In: Proceedings of the 13th International Semantic Web Conference (Oct 2014)