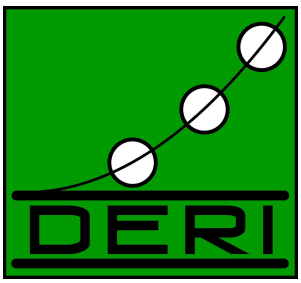
DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE



Weaving the Pedantic Web

Aidan Hogan Andreas Harth
Alexandre Passant Stefan Decker
Axel Polleres

DERI Technical Report 2009-07-28
July 2009

(Edit: 30th March 2010)
Please Cite As:
Aidan Hogan, Andreas Harth, Alexandre Passant,
Stefan Decker, Axel Polleres.
"Weaving the Pedantic Web".
Linked Data on the Web (LDOW 2010), WWW2010
Workshop, Raleigh, North Carolina, USA, 27 April,
2010.

DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE

DERI Galway
IDA Business Park
Galway, Ireland
www.deri.ie

DERI Technical Report
DERI Technical Report 2009-07-28, March 2010

Weaving the Pedantic Web

Aidan Hogan [1], Andreas Harth[1], Alexandre Passant[1],
Stefan Decker[1], Axel Polleres[1]

Abstract. In this paper we provide detailed discussion regarding common mistakes and issues relating to publishing RDF data on the Web. In particular, we discuss the cause, prevalence and possible solutions for the highlighted issues relating to accessibility, dereferenceability, protocol, usage of core vocabularies, datatypes, inconsistencies and *ontology hijacking*. Throughout, we provide statistics and examples from the Web to highlight the prevalence and severity of various observed anomalies in RDF Web data. Continuing, we introduce an online system which can be used by publishers to validate their RDF data with respect to our commonly observed mistakes.

[1]Digital Enterprise Research Institute (DERI), National University of Ireland, Galway,
IDA Business Park, Lower Dangan, Galway, Ireland. E-mail: firstname.lastname@deri.org.

Acknowledgements: The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2), European FP6 project inContext (IST-034718) and an IRCSET Postgraduate Research Scholarship.

1 Introduction

There has been continuous growth in both the scale and diversity of data published on the Web. The Semantic Web movement aims to bring structure to this data by providing a stack of technologies, the core of which is the Resource Description Framework (RDF) for publishing data in a machine-readable format: there now exist millions of RDF data sources on the Web contributing billions of statements. The Semantic Web technology stack includes means to supplement instance data being published in RDF with ontologies described in RDF Schema (RDFS) [6] and the Web Ontology Language (OWL) [19] providing machines a more sapient understanding of the data. However, such an advancement in knowledge representation on the Web has inevitably led to many teething problems in RDF publishing. As we show in this paper, problems are not only present in low-volume publishing, such as handcrafted RDF documents, but are also worryingly present in higher-volume RDF exporters and popular Web ontologies.

With regards to creating data using Semantic Web standards, currently there is no clear way of validating that the document is error-free and represents what is intended: oftentimes, publishers have to rely upon their experience and know-how, or upon community-driven feedback, with varying success. Indeed, in this paper we show that there exists a lot of noise, of many forms, within RDF data published on the Web and that this noise inhibits Semantic Web applications which process and interpret such data. Although applications can employ heuristics to provide workarounds for common mistakes, lossless interpretation of erroneous data is often impossible; also, the implementation of such workarounds provides a major obstacle for development of various software agents and hinders re-use of existing APIs implemented for various Web standards. Indeed, we also show that the majority of noise in RDF Web data is caused by minor unintentional errors in exporters which could be easily fixed by their owners.

We should pause for a moment to consider Alice, a hypothetical new user of a Semantic Web application which provides live browsing of RDF Web data. She loads some interesting data about herself and is immediately impressed by the integrated view of her data from publication, blog, social network and workplace exporters; however, for every second resource she explores, the application cannot locate or parse any relevant data. She tries to load her papers into a calendar view, but one quarter of them are missing as the date-times are illegal values. She wants more information relating to properties and classes used to describe herself, but some do not exist; discouraged, she clicks on a friend of hers but finds that he has 986 names and email addresses (she knew him as Bob). Getting more disappointed, she now finds out that her professor is also a document and begins to notice that all resources she explores are instances of nine strange properties. The final straw: she then searches for Linked Data, to find out what this is all about, only to find that it is a *person*; she is now clearly as confused as the machine that came to this apocryphal conclusion[1]. We will provide evidence in this paper that will show how, and why, Alice could have encountered such events when browsing RDF data currently on the Web.

Continuing, in this paper, we firstly hope to provide valuable insights into what measures Semantic Web applications must employ to be tolerant with respect to defective Web data; secondly, we hope to make RDF publishers more conscious of following best-practices and avoid the common pitfalls associated with their task; thirdly, we identify the causes for common mistakes and propose

[1] This is caused by two factors: (i) the rather generically named property `foaf:img` having domain `foaf:Person` and (ii) a document from the DBpedia exporter (`http://dbpedia.org/data/Linked_Data.xml`) inappropriately using said property on an instance representing Linked Data.

solutions for avoiding repetition in future; ultimately, we provide a validation system which we hope will eventually provide Alice with a more encouraging Semantic Web browsing experience.

To summarise, Section 2 presents the dataset used for our analysis; then, Section 3 identifies, categorises, provides examples and statistics for, and discusses the severity of numerous common issues relating to RDF Web data. Section 4 presents related work and Section 5 concludes.

2 Dataset

Throughout this paper, we will present insights into the prevalence of RDF-publishing issues using a representative Web dataset[2]; in Table 1 we present the diverse set of namespaces used throughout this paper. We retrieved this dataset from the Web in April 2009 by means of a Web-crawl using MultiCrawler [10]; beginning with Tim Berners-Lee's FOAF file[3], we performed a seven-hop breadth first crawl for RDF/XML files using the value `application/rdf+xml` for the HTTP `Accept` request-header [9]. After each hop, we extracted all URIs from the crawled data as input for the next hop. Finally, we restricted the crawl according to pay-level-domains; we enforced a maximum of 5,000 crawled documents from each domain so as to ensure a diverse and representative Web dataset covering a wide spectrum of both high- and low-volume data-publishers. The crawl consisted of access to 149,057 URIs, and acquired 54,836 (36.8%) valid RDF/XML documents. The crawled data contains 12,534,481 RDF statements mentioning 1,598,521 URIs (URIs in the last hop and excluded by the domain limit were not crawled) including 5,850 classes and 9,507 properties.

We also performed reasoning over the dataset, using the Scalable Authoritative OWL Reasoner (SAOR) [13]. For more information, including the full list of rules based on a large fragment of RDF(S)/OWL semantics, we refer the reader to [13]. Please note that in this paper, we explicitly state when statistics are derived from the reasoned dataset; otherwise, the statistics are derived from the directly asserted statements.

3 RDF Web Data Issues

In this section, we enumerate and discuss issues present in RDF data published on the Web. In order to do so in a structured fashion, we loosely follow the Semantic Web layer cake[4], working from the bottom up: we begin in Section 3.1 with issues relating to how data is found and accessed; in Section 3.2 we discuss parsing and syntax issues; in Section 3.3 we analyse esoteric usage of the core RDF(S)/OWL vocabularies; in Section 3.4 we look at reasoning issues, including inconsistency; and finally, in Section 3.5 we discuss issues relating to non-authoritative redefinitions of remote classes and properties.

3.1 URI/HTTP: accessibility and derefencability

First off, we must have some means of publishing RDF data on the Web conformant to standard HTTP practices as defined in RFC 2616 [9]. In this section, we identify the most pertinent issues

[2]Throughout this paper we provide many examples of errors directly using URLs of RDF Web documents; for future reference we also provide snapshots of these documents, with the errors as discussed, at `http://aidanhogan.com/alerts/snapshots/`.

[3]`http://www.w3.org/People/Berners-Lee/card`

[4]cf. `http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html`

affx:	http://www.affymetrix.com/community/publications/affymetrix/tmsplice#
bibtex:	http://purl.oclc.org/NET/nknouf/ns/bibtex#
cycann:	http://sw.cyc.com/CycAnnotations_v1#
foaf:	http://xmlns.com/foaf/0.1/
ical:	http://www.w3.org/2002/12/cal/ical#
linkedct:	http://data.linkedct.org/resource/linkedct/
owl:	http://www.w3.org/2002/07/owl#
politico:	http://www.rdfabout.com/rdf/schema/politico/
qdos:	http://foaf.qdos.com/lastfm/schema/
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
rss:	http://purl.org/rss/1.0/
scot:	http://scot-project.org/scot/ns#
sioc:	http://rdfs.org/sioc/ns#
sioct:	http://rdfs.org/sioc/types#
slink	http://www.semanlink.net/2001/00/semanlink-schema#
swrc:	http://swrc.ontoware.org/ontology#
wn:	http://www.w3.org/2006/03/wn/wn20/schema/
wordmap:	http://purl.org/net/ns/wordmap#
xsd:	http://www.w3.org/2001/XMLSchema#

Table 1: Namespaces and prefixes used

404	403	500	503	401
6,483 (4.3%)	632 (0.4%)	581 (0.4%)	112 (<0.1%)	63 (<0.1%)

Table 2: Count of top five 4xx/5xx reponse codes returned (% of total URIs)

relating to the HTTP standard, and also provide some brief insights into uptake of Linked Data principles on the Web. We commence by analysing how many URIs resolve to an information resource.

3.1.1 Document not retrievable

Briefly, here we mean that a URI found through crawling does not resolve to a Web document; such an attempt will usually result in the return of a response code from the HTTP 4xx (client error) or 5xx (server error) classes [9]. During our crawl, 68.2% of URIs crawled returned a response code of 200 OK and 26.5% returned a 3xx redirect code. However, 5.3% of URIs returned a 4xx/5xx response code: Table 2 summarises the error codes returned.

For the most part, simply nothing exists at that location and a 404 Not Found code is returned. 403 Forbidden indicates that the server cannot service the request (e.g., if directory listings have been turned off). The 500 Internal Server Error responses stem from requests for dynamic content[5]. All but 10 of the 503 Service Unavailable responses were from the w3.org and bio2rdf.org domain, and all are now again online. The 401 Unauthorized URIs required authentication.

[5]e.g. 46 response code 500 URIs (Internal Server Error) stem from SPARQL queries issued to the endpoint at http://www4.wiwiss.fu-berlin.de/dblp/sparql

	text/html	application/rdf+xml	image/jpeg	text/plain	text/xml
All	70,122 (47%)	51,532 (34.6%)	6,532 (4.4%)	6,206 (4.2%)	5,715 (3.8%)
RC=200	35,423 (34.8%)	46,136 (45.4%)	6,532 (6.4%)	830 (0.8%)	5,713 (5.6%)

Table 3: Top five content-types returned, as a percentage of (i) all URIs and (ii) only URIs which return response code 200

application/rdf+xml	text/xml	application/xml	text/plain	text/html
45,565 (83.1%)	5,229 (9.5%)	3,221 (5.9%)	535 (1%)	202 (0.4%)

Table 4: Top five content-types returned for valid RDF/XML documents

3.1.2 Faulty content negotiation

Although a URI has resolved to a Web document, the content may not abide by the HTTP content-negotiation request issued by the downloading agent. In our scenario, we specifically refer to non-RDF/XML content being returned for the specified accept `application/rdf+xml` media type; for the moment this is the only RDF-specific registered media type [20] and although unofficial media types are used for other formats (e.g., N3 uses `text/n3`, `text/rdf+n3`, `application/n3`, `application/x-n3`) and non-RDF-specific media types are used for other formats (e.g., RDFa uses `application/xhtml+xml`), we exclude discussion of these. Table 3 presents the top five content types returned for the URIs crawled using `Accept: application/rdf+xml`.

The most commonly returned content-type is in fact `text/html` (47%) with `application/rdf+xml` only second (34.6%). However, including only the 101,709 URIs which returned a valid response code of `200` (and thus excluding redirects and error codes), 45.4% of URIs report a content-type `application/rdf+xml`, as opposed to 34.8% `text/html`. In almost all instances of a non-RDF content-type, the URI is simply a document without any supporting RDF metadata; thus the problem is not related to content-negotiation errors, but an absence of suitable RDF data to return at that location. Commonly in RDF data, information resource URIs are used to identify themselves (or more problematically to identify related resources); for example, the value of the `foaf:homepage` property is commonly the URI of a HTML document. Also for example, FOAF recommends use of the `tel:` URI scheme for the `foaf:phone` property, which is not dereferenceable.

3.1.3 `Content-Type`/actual format mismatch

The HTTP defined `Content-Type` entity-header field is intended to encode the media type of the content served to the requesting agent. However, oftentimes these do not match. During our crawl, we found that 5,967 documents returning the `application/rdf+xml` content-type were not valid RDF/XML documents. Of these, 5,396 (90.4%) were documents that did not contain any content (e.g., were redirects, etc.). Conversely, Table 4 lists the top five content-types returned for the 54,836 documents found to be valid RDF/XML.

In summary, 16.9% of valid RDF/XML documents are served with a content-type other than `application/rdf+xml`; we note here that a software agent requiring RDF/XML should not reject content on the basis of the reported MIME-type and, similarly, all RDF/XML documents should return the official MIME-type.

unescaped '&'	misplaced ';'	closing tag mismatch	unescaped '<'	invalid QName
243	85	82	79	78

Table 5: Count of the five most common XML parse issues

3.1.4 Linked Data accessibility

Since our crawl was based on access to the URIs used in RDF data, from the statistics already presented in this section we can make some interesting observations relating to Linked Data principles. Tim Berners-Lee identified four tenets for Linked Data publishing which are as follows: (i) use URIs as names for things; (ii) use HTTP URIs so that those names can be looked up; (iii) provide useful information when a look-up on that URI is made; and (iv) include links to other URIs [3].

With regards to providing information about a resource upon a HTTP lookup of its URI – called dereferencing – emphasis is placed on providing information in the format requested (particularly RDF/XML and HTML) and disambiguating identification of information resources (document URIs) from non-information resources (entities described in those documents). Two suitable URI naming schemes are provided. The first uses a '#' fragment identifier construct which is appended onto a document URI along with a name for the entity; the resulting URI uniquely identifies the "non-information" resource and should dereference (by removal of the fragment) to the document describing that resource. The other scheme allows use of any valid HTTP URI to identify a "non-information" resource, whereby that URI provides a `303 See Other` redirect to the describing document.

From our dataset, 4.8% of the URIs used to identify resources in our RDF dataset could not be used to access any document. 14.6% of the URIs found offered a `303` redirect to another location as recommended; however, another 8% used a `302` redirect and a further 3.9% used a `301` redirect. Although software agents could provide support for other redirect schemes, the `303` code should be used to unambiguously assert that the URI identifies a "non-information" resource. Further, of the URIs which provided content, 54.6% reported a content-type other than `application/rdf+xml` and so do not provide an RDF/XML description of the resource.

3.2 XML, RDF/XML, datatypes: syntax

At the outset of the Semantic Web movement, publishers opted to employ the existing XML standard to encode RDF; RDF/XML is still the most popular means of publishing RDF today. In this paper we focus solely on RDF/XML documents resident on the Web and in this section, we focus on documents which are invalid XML, or are valid XML but invalid RDF/XML.

3.2.1 Invalid XML

At the highest level, an RDF/XML document may not be parsable due to XML syntax errors. Of the 46,136 documents which return response code `200` and content-type `application/rdf+xml`, 485 (1%) were invalid XML documents. Table 5 illustrates the top five type of XML syntax errors found in such documents (please note that one document may have more than one type of error).

The most common problem is that of unescaped special characters. The 'misplaced ;' errors

unqualified element	multiple children for predicate node	rdf:resource used as predicate	illegal value for rdf:ID	illegal value for rdf:parseType
24	18	9	8	7

Table 6: Count of the five most common RDF specific parse issues

stem from a FOAF[6] exporter within the trust.mindswap.org domain[7]. Invalid QName issues arise from a FOAF exporter on the semanticweb.org domain[8] where a blank xmlns: is used. We conclude that such exporters do not use standard XML APIs for creating their content; in any case, such problems are quite rare as methods of validating XML documents are well known and all errors should be easily resolvable in the exporter.

3.2.2 Valid XML but invalid RDF/XML

A document may be valid XML but invalid RDF: publishers of RDF/XML documents sometimes neglect to ensure valid 'striping' [2] which results in an XML document with no clearly defined triple strata. In addition, a document may misuse the RDF/XML syntactic shortcuts; viz.: the elements RDF, Description and the attributes ID, about, parseType, resource, li, nodeID, datatype.

Of the 46,136 documents which return response code 200 and content-type application/rdf+xml, 86 (0.2%) were valid XML but invalid RDF/XML documents. Table 6 lists the five most common RDF specific parse issues.

The most common issue relates to unqualified element names (missing namespace); the next two issues stem from incorrect striping leading to property or attribute constructs used where not allowed; the latter two issues relate to misuse of RDF syntactic terms. Again, such issues are relatively rare, presumably due to use of RDF/XML APIs for producing data and the popularity of the W3C RDF/XML validation service[9].

3.3 RDF(S)/OWL: esoteric use

Continuing, we look at *esoteric* usage of the core RDF(S)/OWL vocabularies on the Web. By esoteric, we refer to usage of classes and properties which, although not disallowed by the official specifications or indicative of inconsistency, is generally inadvisable and often indeliberate.

3.3.1 Atypical use of collections, containers and reification

There is a set of URI names which are reserved by the RDF specification for special interpretation in a set of triples; although the RDF specification does not formally restrict usage of these reserved names, misuse is often inadvertent.

We firstly discuss the RDF collection vocabulary, which consists of four constructs: {List, first, rest, nil}. Indeed, few examples of atypical collection usage exist on the Web. This is attributable to widespread usage of the rdf:parseType="Collection" RDF/XML shortcut for specifying collections; this shortcut shields users from the underlying complexity of collections on

[6] Friend of a Friend: http://xmlns.com/foaf/0.1/
[7] cf. http://trust.mindswap.org/cgi-bin/FilmTrust/foaf.cgi?user=aaronsw
[8] cf. http://semanticweb.org/wiki/Special:ExportRDF/Axel_Polleres
[9] http://www.w3.org/RDF/Validator/

foaf:member_name	foaf:tagLine	foaf:image	cycann:label	qdos:neighbour
148,251	148,250	140,791	123,058	100,339

Table 7: Count of the top five properties used without a definition

sioc:UserGroup	rss:item	linkedct:link	politico:Term	bibtex:inproceedings
21,395	19,259	17,356	14,490	11,975

Table 8: Count of the top five classes used without a definition

the triple level and generally ensures typical collection use. The only atypical collection usage we found in our Web-crawl was one document which specified resources of type `List` without `first` or `rest` properties attached[10].

A related issue is that of atypical container usage, which is concerned with the following constructs: `Alt`, `Bag`, `Seq`, `_1...n` and the syntactic keyword `li`. Again, atypical container usage is uncommon on the Web: we found one domain (viz. `semanticweb.org`[11]) which, in 229 documents, exports RDF containers without choosing a type of `Alt`, `Bag` or `Seq`.

Finally, there may exist atypical usage of the reification constructs: `Statement`, `subject`, `predicate`, `object`. However, in our dataset we only found one such example[12] wherein `predicate` is assigned a blank node value and used alone without `subject` or `object`.

3.3.2 Use of undefined classes/properties

Oftentimes, a term is used in the predicate position of a triple, which is not formally defined as being a property: Table 7 enumerates the five most common found in our dataset.[13] We can conclude that publishers seem to invent terms within a related namespace for their exporters (`foaf:member_name`/`foaf:tagLine`). Also, publishers may misspell genuine terms (`foaf:image` should be `foaf:img`). The presented FOAF errors all stem from the `livejournal.com` domain[14]; however many more exist across many domains. The `cycann:label` issue results from mistaken use of the RDF/XML shortcut `rdf:about` (which creates the term using the base URI of the document pared at the last slash) instead of `rdf:ID` (which creates the term by appending a hash and unique name to the whole document URI) to identify the property in the term's dereferenced document. Similarly, the term `qdos:neighbour` is misspelt `qdos:neighbours` in its dereferenced document.

There are also many instances of classes which have not been defined, with the top five enumerated in Table 8. Neither of the first three classes nor the last class are defined in the dereferenced documents; for example, many of the `sioc:UserGroup` instances come from the `apassant.net` domain[15]. The class `politico:Term` is generically described in the dereferenced document, but is neither implicitly nor explicitly typed as a class.

Again, most of the errors outlined are due to spelling or syntactic mistakes resolvable through minor fixes to the respective ontologies or exporters. Where terms have been knowingly invented,

[10] `http://scripts.mit.edu/~kennylu/myself.rdf`

[11] cf. `http://iswc2006.semanticweb.org/submissions/Harth2006dq_Harth_Andreas`

[12] `http://web.mit.edu/dsheets/www/foaf.rdf`

[13] Please note that, in this paper, we include prefix mappings for less well known namespaces as footnotes under the respective table for the reader's convenience.

[14] cf. `http://danbri.livejournal.com/data/foaf`

[15] cf. `http://apassant.net/home/2007/12/flickrdf/data/people/36887937@N00`; indeed the authors herein are prone to making simple errors in their publishing and would benefit from the online validation system presented in Section 5.

rdfs:range	foaf:Image	rdfs:Class	wot:PubKey	foaf:OnlineAccount
8,012	639	94	18	15

Table 9: Top five "classes" used in the predicate position of a triple

foaf:knows	foaf:name	foaf:sha1	swrc:author	foaf:based_near
4	4	2	1	1

Table 10: Top five properties found in the object position of an `rdf:type` triple

we suggest that the term be recommended as an addition to the respective ontology, or defined in a seperate namespace.

3.3.3 Misplaced classes/properties

Here, we mean that a URI defined as a class appears in the predicate position of a triple or, conversely, that a URI defined as a property appears in the object position of an `rdf:type` triple. Table 9 shows the top five classes used in the predicate position of a triple. One source[16] defines `rdfs:range` as being of `rdf:type rdfs:Class` (this is non-standard use of RDFS: cf. [7, 15, 13]); hence the 8,012 occurrences are valid use of the property. Most occurrences of `foaf:Image` used as a property stem from the `sembase.at` domain[17]; here the `foaf:depiction` property would be more suitable. Use of `rdfs:Class` as a predicate (also non-standard use) comes from the `ajft.org` and `rdfweb.org` domains[18] where `rdfs:Class` is seemingly mistaken as `rdf:type`. The class `wot:PubKey` is mistakingly used instead of `wot:hasKey`[19]. Misuse of `foaf:OnlineAccount` stems from one document[20] wherein the RDF/XML shortcut `rdf:parseType="Resource"` is used inappropriately, causing interpretation of `foaf:OnlineAccount` elements as predicates.

After reasoning, more such errors were discovered, particularly in the `affymetrix.com` domain[21] which describes genes and uses `rdfs:subClassOf` to assert subsumption relations between properties (amongst many other issues); this resulted in properties – which, combined, were used in 37,454 triples – being typed as classes.

Conversely, the usage of properties in the class position is much less common; Table 10 lists the results, with most errors stemming from one document[22]. Again, all such errors by the publisher should be easily fixed once they are made aware.

3.3.4 Misuse of `owl:DatatypeProperty`/`owl:ObjectProperty`

The defined class `owl:DatatypeProperty` describes properties which relate some resource to a literal value; similarly, the OWL class `owl:ObjectProperty` describes properties which relate some resource to a URI or blank-node value. Although misuse of such properties is not strictly inconsistent within OWL, it is indicative of errors in the data.

[16] `http://www.w3.org/2000/10/swap/infoset/infoset-diagram.rdf`
[17] cf. `http://wiki.sembase.at/index.php/Special:ExportRDF/Dieter_Fensel`
[18] `cf.http://swordfish.rdfweb.org/discovery/2004/01/www2004/files/1101776794087.rdf`
[19] cf. `http://www.snell-pym.org.uk/alaric/alaric-foaf.rdf`
[20] cf. `http://tommorris.org/foaf`
[21] cf. `http://affymetrix.com/community/publications/affymetrix/tmsplice/all_genes.1.rdf`
[22] `http://www.marconeumann.org/foaf.rdf`

swrc:journal	swrc:series	ical:location	foaf:name	foaf:msnChatID
19,853	14,963	4	4	3

Table 11: Top five datatype-properties with non-literal objects

affx:startsAt	affx:stopsAt	affx:cdsType	affx:frame	affx:commonToAll
6,234	6,234	5,193	4,882	4,814

Table 12: Top five object-properties with literal objects

There were 34,835 datatype-properties with non-literal objects (in 1,194 documents). Table 11 lists the top five; the only significant errors stem from `l3d.de`[23] which exports RDF from the Digital Bibliography & Library Project (DBLP) – they define two datatype-properties in the `swrc:` namespace but only use the properties with non-literal objects.

Analogously, there were 41,752 instances of object-properties (in 4,438 documents) with literal values. Table 12 lists the top five, which all come again from the `affymetrix.com` domain. However, outside of this table, there were many other properties with significant misuse including miscellaneous properties from the `opencyc.org` domain (6,161), `foaf:page` (3,160), `foaf:based_near` (1,078), `ical:organizer` (456), amongst others; the errors were spread over 92 hosts. In fact, at the time of writing, the property `foaf:myersBriggs` (in the FOAF specification itself) is incorrectly defined as an `owl:ObjectProperty` with `rdfs:range rdfs:Literal` and had 35 literal values in our dataset.

All errors can be solved by removing the datatype-/object-property constraint. Where such constraints are erroneously specified, they can simply be reversed by the ontology maintainers. However, in many cases such constraints are purposefully defined to ensure consistency in usage of the term; in this case, publishers should abide by such usage.

3.3.5 Members of deprecated classes/properties

The OWL classes `owl:DeprecatedClass` and `owl:DeprecatedProperty` are used to indicate classes or properties that are no longer recommended for use. In our dataset, we did not find any members of a deprecated class; however, we found 290 instances (in 115 documents) of four deprecated properties: `wordmap:subCategory` (260), `sioc:has_group` (15), `sioc:content_encoded` (10) and `sioc:description` (5).

3.4 Reasoning: noise and consistency

In this section, we will look at issues relating to reasoning on the Web; particularly we look at noise with respect to inverse-functional properties, datatypes and logical inconsistencies.

3.4.1 Bogus `owl:InverseFunctionalProperty` values

Resources are also commonly identified by values for properties unique to a resource. Such properties are termed "inverse-functional" and are identified in OWL as the class `owl:InverseFunctionalProperty`. For example, the FOAF ontology has defined a number of inverse-functional properties for identifying people; these include `foaf:homepage`, `foaf:mbox` (email), `foaf:mbox_sha1sum` (encoded email to prevent spamming), amongst others.

[23] cf. `http://dblp.l3s.de/d2r/data/publications/conf/aswc/HoganHP08`

Property	Value	Count
foaf:mbox_sha1sum	"08445a31a78661b5c746feff39a9db6e4e2cc5cf"	986
foaf:mbox_sha1sum	"da39a3ee5e6b4b0d3255bfef95601890afd80709"	167
foaf:homepage	<http://>	11
foaf:mbox_sha1sum	""	5
foaf:isPrimaryTopicOf	<http://>	2

Table 13: Count of the five most common void inverse-functional property values

However, FOAF exporters commonly do not respect the semantics of these inverse-functional properties and export 'void' values given partial user-input (we have previously documented this problem in [12]); Table 13 details the top five void values for inverse-functional properties which we found in our dataset.

The first two sha1sum values are for 'mailto:' and the empty string created by FOAF exporters who do not properly constrain user-input for email fields. Each group of resources with the same values should be inferred as equivalent. The problem is quite widespread, with 52 hosts contributing 1,169 bogus values in 1,041 documents. For example, 194 errors come from the bleeper.de domain[24], 189 from identi.ca[25], 166 from uni-karlsruhe.de[26], 163 from twit.tv[27] and 92 from tweet.ie[28].

According to the standard reflexive, symmetric and transitive semantics of owl:sameAs, if we take for example the 986 entries with the same null sha1 value, 986^2=972k owl:sameAs statements would be inferred. Further, assuming, for example, an average of eight triples mentioning each equivalent resource, 972k*8 = 7.8M statements would be inferred by substituting each equivalent identifier into each statement. In other words, such chains of equality cause a quadratic explosion of inferences; when one considers larger Web-crawls, the problem becomes quite critical. For publishers, the issue is easily resolved by, for example, validating user input and checking the uniqueness and validity of inverse-functional values. Clearly reasoning agents need to counter-act such errors by, for example, blacklisting such values.

3.4.2 Datatype issues

In RDF entailment [11], the only inconsistencies that can occur are XML clashes where a malformed (ill-typed)[29] XML literal is given the datatype rdf:XMLLiteral and is stated to be a literal through use of a surrogate blank node (please see [11] for more detail); however, malformed XML literals cannot occur in valid RDF/XML documents, and so a valid RDF/XML document is RDFS consistant (thus, we find no such examples after parsing).

According to D-interpretations [11] (a more generic datatype entailment regime), datatype-clashes can occur as above for malformed literals of any supported datatype; also, datatype-clashes can occur if a literal simultaneously occupies two disjoint datatype classes [11]. Here, we focus

[24] cf. http://bleeper.de/powerboy/foaf
[25] cf. http://identi.ca/whataboutbob/foaf
[26] cf. http://www.aifb.uni-karlsruhe.de/Personen/viewPersonFOAF/foaf_1876.rdf
[27] cf. http://army.twit.tv/takeit2/foaf
[28] cf. http://tweet.ie/seank/foaf
[29] Please note that although a 'malformed' datatype literal is also 'ill-typed', we use the former term to indicate errors relating to the syntax of a literal, and the latter to indicate errors purely relating to typing of literals.

xsd:dateTime	xsd:int	xsd:nonNegativeInteger	xsd:gYearMonth	xsd:gYear
4,042 (26.4%)	250 (2.1%)	232 (0.6%)	67 (100%)	27 (1.4%)

Table 14: Top five datatypes having malformed values (% of all values which are illegal)

on XSD-interpretation/clashes wherein the primitive XSD datatypes – and their respective sets of sub-datatypes – are considered pairwise disjoint [5]. Firstly, we will examine malformed literals (which, granted, are not inconsistent by themselves) and then examine datatype-clashes.

From the content of the crawl, we found 3,666,840 literals of which 170,351 (4.6%) were typed. Of these, the top five most popular datatypes were `xsd:string` (53,879), `xsd:nonNegativeInteger` (38,501), `xsd:integer` (15,826), `xsd:dateTime` (15,824), and `xsd:unsignedLong` (12,318); 732 literals used user-specified datatypes, most of which stem from the `dbpedia.org` domain and some which stem from incorrectly specified XSD namespaces. For the standard datatypes, we found 4,650 malformed literals (2.7% of all typed literals): Table 14 summarises the top five such datatypes.

The two most common errors for `xsd:dateTime` stem from (i) the `wasab.dk` domain[30] whereby time-zones are missing the required ':' separator; and (ii) the `soton.ac.uk` domain[31] whereby the mandatory seconds-field is not specified. For `xsd:int`, almost all errors stem from the `freebase.com` domain whereby boolean values `True` and `False` are found[32]. For `xsd:nonNegativeInteger`, all stem from the `deri.ie` domain[33] where non-numeric strings are incorrectly used. Finally, for `xsd:gYearMonth` and `xsd:gYear`, all illegal usage comes from the `dbpedia.org` domain[34] where full `xsd:dateTime` literals are used instead.

Clearly, malformed literals are quite common (e.g., 26.4% of `xsd:dateTime` literals are invalid). Although datatype aware agents could incorporate heuristics to shoulder common mistakes – for example, the omission of the seconds field from `xsd:dateTime` – not all such mistakes can be accounted for and a publisher should not rely on such leniency. In all examples, the errors can be resolved by simple syntactic fixes to the exporter, or removing or changing the datatype on the literal; one can conclude that maintainers of the exporters are simply not aware of such issues.

Aside from explicitly typed literals, the range of properties may also be constrained to be a certain datatype, mandating respectively typed values for that property; a datatype clash can then occur if the property is given an object (i) that is malformed, or (ii) that is a member of a disjoint datatype. Table 15 provides counts of datatype clashes for the top five such properties. The property `slink:creationDate` has the range `xsd:date` but all triples with `slink:creationDate` in the predicate position have plain-literal objects – all originating from the `semanlink.net` tagging system[35]; please note that plain literals without language tags are considered as `xsd:strings` [11] and so are disjoint with `xsd:date`. The property `scot:ownAFrequency` is given range `xsd:float` but only ever used in the domain `linkeddata.org`[36] with `xsd:integer` objects; `xsd:integer` is a sub-type of `xsd:decimal` and is disjoint with `xsd:float` [5]. `owl:cardinality` is often used with plain-literal

[30] cf. `http://www.wasab.dk/morten/2004/08/photos/1/index.rdf`
[31] cf. `http://rdf.ecs.soton.ac.uk/publication/10006`
[32] cf. `http://rdf.freebase.com/rdf/aviation/aircraft_ownership_count`
[33] cf. `http://www.deri.ie/fileadmin/scripts/foaf.php?id=320`
[34] cf. `http://dbpedia.org/data/1994_San_Marino_Grand_Prix.xml`
[35] cf. `http://www.semanlink.net/tag/rdf.rdf`
[36] cf. `http://community.linkeddata.org/dataspace/kidehen2/subscriptions/Kingsley_Feed_Collection/tag/rdf`

slink:creationDate	scot:ownAFrequency	owl:cardinality	ical:description	wn:tagCount
9212 (100%)	529 (100%)	464 (65.2%)	262 (21.8%)	204 (100%)

Table 15: Top five properties with datatype-clashes (% usage of property)

objects[37] contrary to the defined range xsd:nonNegativeInteger. The property ical:description – defined as having range xsd:string – is almost always instantiated with a plain-literal object (99.8%); however, only the 21.8% which use language tags constitute an inconsistency[38]. Finally, wn:tagCount has range xsd:nonNegativeInteger but is only used with plain literals in the w3.org domain[39].

Although the usage of properties often does not reflect the defined datatype range, in our dataset we found that the literal strings were almost always within the lexical space of the range datatype and that they were just poorly typed. We only found two properties which were given objects malformed according to the range datatype (before, we were concerned with malformed literals given an *explicit* datatype): viz. exif:exposureTime with range xsd:decimal (given 49 plain literals with malformed decimal values in one document[40]) and cfp:deadline with range xsd:dateTime (given 3 plain literals with malformed date-time values in 3 documents[41]). Thus, in all but the latter cases, liberal software agents could ignore mismatches between an object's datatype and that specified by the property's range, parsing the literal string into the value space of the range datatype; however, caution is required when considering non-standard datatypes: consider if a property ex:temp has the datatype ex:celcius as range and is used with an ex:fahrenheit value – clearly the value should not be parsed as ex:celcius although in it's lexical space.

In all cases, the root problem could be resolved by removing the range constraint on the property; in many cases such an approach may even be suitable: properties such as ical:description which are intended to have prose values should remove xsd:string constraints and thus allow use of language tags.[42] However, the majority of such datatype domain constraints are validly used to restrict possible values for the property and the onus is on data-publishers to thereby abide.

3.4.3 OWL inconsistencies

We now introduce various inconsistencies possible in OWL. To begin with, the class owl:Nothing is intended to represent the empty class, and, as such, should not contain any members. In our dataset, we found no directly asserted members of owl:Nothing. Also, an inconsistency can occur when owl:sameAs and owl:differentFrom overlap. However, we found no such examples in our crawl; in fact, we found no usage of owl:differentFrom in the predicate position of a triple. Similarly, although we found two instances of owl:AllDifferent/ owl:distinctMembers usage, none resulted in an inconsistency.

[37] 425 of the 464 examples of plain literal objects for owl:cardinality stem from http://bioinfo.icapture.ubc.ca/subversion/Cartik/Object-OWLDL2.owl

[38] cf. http://www.ivan-herman.net/professional/CV/W3CTalks.rdf

[39] cf. http://www.w3.org/2006/03/wn/wn20/instances/wordsense-act-verb-3.rdf

[40] http://kasei.us/pictures/2005/20050422-WCCS_Dinner/index.rdf

[41] cf. http://sw.deri.org/2005/08/conf/ssws2006.rdf – another example of errors admittedly generated by an author of this paper.

[42] Recent introduction of the new (albeit controversial) rdf:PlainLiteral class may offer a suitable range for "prose-text" properties.

foaf:Agent foaf:Document	foaf:Organization foaf:Person	foaf:Document foaf:Person	sioc:Container sioc:Item	sioc:Item sioc:User
502	328	232	194	35

Table 16: Top five instantiated pairs of disjoint classes

The OWL property `owl:disjointWith` is used to distinguish classes whose intersection is empty; in other words, a resource should not be a member of disjoint classes. Assertions of disjointness between classes in popular Web ontologies are used as an indicator of inconsistent information being provided. In our dataset, we found two instances that were asserted to be members of the disjoint classes `foaf:Document` and `foaf:Project`. Interestingly, both examples came from different domains whereby one domain[43] used a document URI[44] to identify a project, and another domain[45] defined the URI as being a document.

More commonly, memberships of disjoint classes are inferred through reasoning. After reasoning, there were 1,329 occurrences of unique disjoint membership pairs; Table 16 enumerates the top five.

The most prominent cause of such problems stem from two FOAF exporters for LastFM data: URIs are simultaneously defined as being of type `foaf:Person` in the `opiumfield.com` domain[46] and as values for `foaf:isPrimaryTopicOf` in the `dbtune.org` domain[47]; since `foaf:isPrimaryTopicOf` has range `foaf:Document`, both domains together instantiate members of `foaf:Document` and `foaf:Person`. Again, there are many other exporters and domains which contribute; for example, a SIOC Wikipedia exporter in the `sioc-project.org` domain[48] uses the same URI to identify a user (typed as `sioc:User`) and the user's Wikipedia profile page (typed as `sioct:WikiArticle`, an indirect subclass of `sioc:Item`).

Such problems may be quite difficult to solve. The obvious solution is to remove the disjointness constraints from the relevant ontologies; however, these constraints are intended to flag nonsensical or conflicting information and removing them clearly does not solve the root cause. The main cause for such inconsistencies stems from inappropriate re-use of URIs, possibly by different domains, for incompatible resources; agreement must be reached on what is an appropriate identifier for the contentious resource. Often, a term is used to simultaneously identify an information resource and the entity it primarily describes; in such cases the inverse-functional property `foaf:isPrimaryTopicOf` and the URI of the information resource can be used instead to uniquely identify the entity.

Continuing, we also performed similar checks for instances of classes which were defined as complements of each other using `owl:complementOf`; however, we found no `owl:complementOf` relations in our dataset.

Briefly, we also performed simple checks for unsatisfiable concepts whereby one class is (possibly indirectly) both a subclass of and disjoint with another class. For each class found, we performed reasoning on an arbitrary membership of that class and checked whether any of the inferred memberships were of disjoint classes; however, we found no such concepts on the Web.

[43] `http://www.schemaweb.info/foaf/vlindesay.xml`
[44] `http://www.semanticweb.info/`
[45] `http://journal.dajobe.org/journal/2003/07/semblogs/bloggers.rdf`
[46] cf., `http://rdf.opiumfield.com/lastfm/profile/danbri`
[47] cf., `http://dbtune.org/last-fm/danbri.rdf`
[48] cf. `http://ws.sioc-project.org/mediawiki/mediawiki.php?wiki=http://en.wikipedia.org/wiki/User:Andy_Dingley`

3.5 Non-authoritative contributions

In previous work, we have described our system for performing reasoning over RDF Web data called SAOR [13]; as previously mentioned, we use SAOR in this paper to provide statistics over reasoned data. In [13], we found it essential to provide "authoritative" restrictions in our reasoning to protect popular ontologies from unwanted third party contributions; essentially, we allow the document dereferenced by a class/property's URI to speak authoritatively for (control the semantics of) that term. We allow third party (non-authoritative) documents to extend these classes/properties, but not to influence reasoning on their members; we call the redefinition of external classes/properties "ontology hijacking". We refer the interested reader to [13] for a more in-depth treatment.

In our dataset, we found that 5,211 document engaged in some form of ontology hijacking. However, most such occurrences were due to third party sources 'echoing' the authoritative definition of a class or property in their local ontology. However, for example, we found 219 statements declaring `foaf:Image` – authoritatively defined to be a member of `rdfs:Class` – to be an `owl:ObjectProperty`; these were again from the `sembase.at` domain (again see Footnote 17). Another document[49] defines nine of its *properties* as being the domain of `rdf:type` (also non-standard usage).

Clearly, on the Web, people should not be constrained in what they express and where they express it; however, to do useful reasoning, developers must take contextual information into account and provide some means of insulating ontologies from wayward external contributions.

3.6 Summary

To conclude this section, we can now see that although our protagonist Alice is purely hypothetical, her adventures in Semantic Web wonderland are disappointingly less so; in our analysis, we have shown the types of issues in RDF data on the Web that have made her journey so disconcerting. We have determined that many such issues could be easily resolved, either through simple fixes to ontologies or RDF exporters; other issues – particularly relating to inconsistencies, identification of resources and use of dereferenceable URIs – may be more difficult to resolve.

4 Related Work

Earlier papers analysing problems in RDF Web data and the uptake of standards mainly focus on the categorisation and validation of documents with respect to the various OWL species. In [1], the authors performed validation – based on OWL-DL constraints – for a sample group of 201 OWL ontologies which were all found to be OWL Full for mainly trivial reasons; the authors then suggested means of patching the ontologies to be OWL-DL conformant. A similar but more extensive survey was conducted in [22] over 1,275 ontologies; the authors provided categorisation of the expressivity and species and discussion related to patching of the ontologies. At the moment, we do not offer species validation for RDFS/OWL and our scope is much broader with respect to validation.

In [16], the authors describe common user errors in modelling OWL-DL ontologies. In [18], the authors describe some error checking for OWL ontologies using integrity constraints based on the Unique Name Assumption (UNA) and the Closed World Assumption (CWA). Similarly, in [21], various errors and constraints are introduced for error checking; the primary contribution is

[49] `http://www.eiao.net/rdf/1.0`

the introduction of five 'incongruencies' (e.g., an individual not satisfying a cardinality constraint according to UNA/CWA) with cases, causes and methods of detection. However, all of these papers have a decidedly more OWL-centric focus than our work and provide no analysis or discussion of Web data.

In [8], the authors provided an in-depth analysis of the landscape of RDF Web data in a crawl of 300M triples. Also they identified some statistics about classes and properties (SWTs) in RDF data; e.g., they found that 2.2% of classes and properties had no definition and that 0.08% of terms had both class and property meta-usage. However, again our focus is much more broad in characterising errors in RDF Web data.

5 Discussion

In this paper, we have taken stock of the shortcomings of RDF publishing on the Web. We have presented, provided statistics and examples for, and discussed a plethora of different types of errors, hopefully raising awareness of such issues amongst data publishers and developers of agents who wish to access and interpret such data. Most of the errors we have identified are easily fixed, particularly for large-volume exporters; we therefore conclude that publishers have merely been unaware of the problems resident in their data. It is also strange to note that probably the most prominent Semantic Web document – the FOAF spec – had an error relating to the `foaf:myersBriggs` property.

The resolution of errors may sometimes require compromise between maintainers of ontologies and maintainers of exporters which populate the ontologies' terms, reflecting the current social and community driven nature of Web publishing. Ontology maintainers may account for popular demand in their ontologies: for example the FOAF spec recently removed WordNet classes from the ontology because they were not dereferenceable, SIOC has removed domain and range constraints on properties as they were deemed too restrictive for intended applications by users in the community.

Reflecting such community driven efforts, consideration is being given to more open ontology editing and creation. In VoCamp events[50], people from different backgrounds and with different perspectives meet to work on modelling lightweight ontologies for immediate use. In order to allow ontologies to evolve according to user needs, initiatives such as semantic wikis for ontology management [14] and services such as OpenVocab[51] allow users to more freely interact with the ontology terms they wish to use and share. Although such approaches may again suffer from human error and disagreement – and have many open issues such as versioning and editing privileges – such community-driven efforts could lead to a more extensive vocabulary of terms for use on the Web.

For now, we return to helping Alice. One short-term solution would be to provide a system for validating RDF data being published to the Web: several systems exist but do not cover the broad range of issues discussed in this paper. From a syntactic point of view, the first validator available was the W3C RDF Validator[52], being able to check the syntax of any RDF/XML document (however, not datatype syntax). The DAML validator[53] provides checking of a large number of issues; however the validator is out of date (does not support OWL), and, at the time of writing,

[50] `http://vocamp.org/wiki/Main_Page`
[51] `http://open.vocab.org/`
[52] `http://www.w3.org/RDF/Validator/`
[53] `http://www.daml.org/validator/`

does not work. With regards to the protocol issues, the online Vapour validator[54] [4] aims at validating the compliance of published RDF data (either vocabularies or instances data) according to Linked Data principles [3]. The online Pellet [17] validator[55] enables species validation as well as other criteria we identified such as checking ontology consistency and finding unsatisfiable concepts.

There are also a number of command-line validators. The Validating RDF Parser (VRP)[56] operates on specified RDF Schema constraints, with some support for datatypes. The Eyeball[57] project provides command-line validation of RDF data for common problems including use of undefined properties and classes, poorly formed namespaces, problematic prefixes, some simple literal validation (not including validation of lexical values) and other optional heuristics.

However, none of them solve the plethora of issues we have encountered. Thus, we present "RDF Alerts": `http://swse.deri.org/RDFAlerts/`. We have already shown that most noise in RDF Web data can be easily resolved and so the system is intended to be a pragmatic tool which RDF publishers and exporters can use to detect and investigate otherwise hidden issues with their data – in a timely fashion. In Figure 1 we detail the architecture of our system. We use Squid[58] as a caching proxy to speed up access to Web resources; we use the YARS2 endpoint[59] to query the broader RDF Web for equivalent resources and possible inconsistencies; again, we use SAOR to provide OWL reasoning. Given a URI input by a user, we provide validation for all of the issues enumerated in this paper and intend to extend the tool according to community feedback.

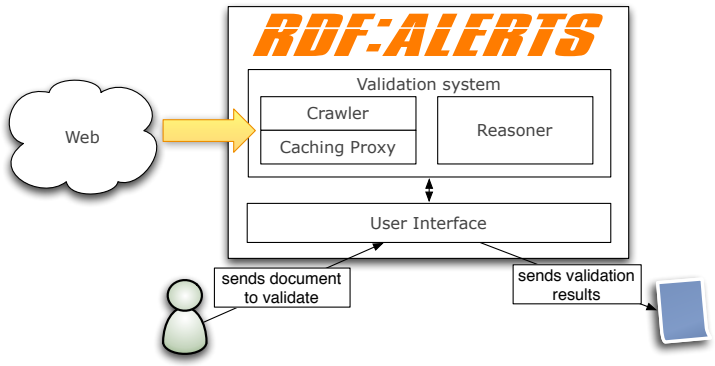


Figure 1: RDF Alerts architecture

To finally conclude, we would like to replace the present hypothetical Alice with a possible future Alice who is again browsing the Semantic Web – however this time using an application which has been tempered for noisy data, where the documents have been validated, consistent identifiers used, and resources described using a rich vocabulary of community-edited terms. We hope that such an Alice might be amazed – this time for the right reasons.

[54] `http://validator.linkeddata.org`
[55] `http://www.mindswap.org/2003/pellet/demo.shtml`
[56] `http://139.91.183.30:9090/RDF/`
[57] `http://jena.sourceforge.net/Eyeball/`
[58] `http://www.squid-cache.org/`
[59] `http://swse.deri.org/yars2`

References

[1] S. Bechhofer and R. Volz. Patching syntax in OWL ontologies. In *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 668–682. Springer, November 2004.

[2] D. Beckett and B. McBride. RDF/XML syntax specification (revised). W3C Recommendation, Feb. 2004. `http://www.w3.org/TR/rdf-syntax-grammar/`.

[3] T. Berners-Lee. Linked Data. Design issues for the World Wide Web, World Wide Web Consortium, 2006. `http://www.w3.org/DesignIssues/LinkedData.html`.

[4] D. Berrueta, S. Fernndez, and I. Frade. Cooking HTTP content negotiation with Vapour. In *Proceedings of 4th Workshop on Scripting for the Semantic Web (SFSW2008)*, June 2008.

[5] P. V. Biron and A. Malhotra. XML Schema part 2: Datatypes second edition. W3C Recommendation, Oct. 2004. `http://www.w3.org/TR/xmlschema-2/`.

[6] D. Brickley and R. Guha. RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, Feb. 2004. `http://www.w3.org/TR/rdf-schema/`.

[7] J. d. Bruijn and S. Heymans. Logical foundations of (e)RDF(S): Complexity and reasoning. In *6th International Semantic Web Conference*, number 4825 in LNCS, pages 86–99, Busan, Korea, Nov 2007.

[8] L. Ding and T. Finin. Characterizing the Semantic Web on the Web. In *Proceedings of the 5th International Semantic Web Conference*, November 2006.

[9] R. Fielding, J. Gettys, J. Mogul, H. F. Nielsen, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. RFC 2616, 1999. `ftp://ftp.isi.edu/in-notes/rfc2616.txt`.

[10] A. Harth, J. Umbrich, and S. Decker. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *5th International Semantic Web Conference*, pages 258–271, 2006.

[11] P. Hayes. RDF semantics. W3C Recommendation, Feb. 2004. `http://www.w3.org/TR/rdf-mt/`.

[12] A. Hogan, A. Harth, and S. Decker. Performing object consolidation on the semantic web data graph. In *1st I3 Workshop: Identity, Identifiers, Identification Workshop*, 2007.

[13] A. Hogan, A. Harth, and A. Polleres. Scalable Authoritative OWL Reasoning for the Web. *Int. J. Semantic Web Inf. Syst.*, 5(2), 2009.

[14] M. Krötzsch, S. Schaffert, and D. Vrandecic. Reasoning in semantic wikis. In *Reasoning Web*, pages 310–329, 2007.

[15] S. Muñoz, J. Pérez, and C. Gutiérrez. Minimal deductive systems for RDF. In *ESWC*, pages 53–67, 2007.

[16] A. L. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *EKAW*, pages 63–81, 2004.

[17] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[18] E. Sirin, M. Smith, and E. Wallace. Opening, closing worlds - on integrity constraints. In *OWLED*, 2008.

[19] M. K. Smith, C. Welty, and D. L. McGuinness. OWL Web Ontology Language Guide. W3C Recommendation, Feb. 2004. `http://www.w3.org/TR/owl-guide/`.

[20] A. Swartz. application/rdf+xml media type registration. RFC 3870, 2004. `http://www.ietf.org/rfc/rfc3870.txt`.

[21] J. Tao, L. Ding, and D. L. McGuinness. Instance data evaluation for semantic web-based knowledge management systems. In *HICSS*, pages 1–10, 2009.

[22] T. D. Wang, B. Parsia, and J. A. Hendler. A survey of the web ontology landscape. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, pages 682–694, Athens, GA, USA, Nov. 2006.