



OKKAM – Enabling a Web Of Entities
Grant Agreement No. 215032

D8.2: Web Wide Harvesting cluster

Document Number	D8.2
Document Title	Web Wide Harvesting cluster
Version	0.3
Status	Final
Work Package	WP8
Deliverable Type	Report/Prototype/Demonstrator/Other
Contractual Date of Delivery	31/12/2008
Actual Date of Delivery	
Responsible Unit	
Contributors	National
Keyword List	
Dissemination level	PU/PP/RE/CO

Change History

Version	Date	Status	Author (Company)	Description
0.1	11/01/2008	Draft	Giovanni Tummarello (DERI)	First version available for review
0.2	05/01/2009	Draft	Richard Cyganiak	Review
0.3	27/5/2009	Final	Giovanni Tummarello	More polishing

Executive Summary

In this document we present the works that have been conducted on harvesting and preprocessing web data.

For the purpose of the Entity Centric Semantic Search Engine (ECSSE), we have concentrated our efforts on the collection and indexing of the Web of Data: the RDF world and the world of web pages annotated with Microformats or RDFa (also see D8.1).

This sort of data collection is relatively unexplored. For this reason we have developed several techniques to speed up both the collection of large datasets and the later processing, e.g. the reasoning processes which are fundamental to have a meaningful precision and recall in the later query process.

Furthermore, in this document we present the normative way by which OKKAM identifiers should be embedded in HTML pages.

Table of Contents

1. INTRODUCTION: ADVANCED WEB DATA HARVESTING AND PROCESSING TECHNIQUES.....	6
2. OKKAM ENTITIES AND MICROFORMATS TO RDF.....	7
2.1. CONVERTING MICROFORMATS TO RDF INSIDE SINDICE AND ECSSE	7
2.2. GENERAL HTML EXTRACTION.....	7
2.2.1. XFN.....	7
2.2.2. hCard.....	8
2.2.3. Geo.....	9
2.2.4. Adr	9
2.2.5. hCalendar	10
2.2.6. hReview.....	10
2.2.7. rel-license	11
2.2.8. hListing	11
2.2.9. hResume.....	12
2.2.10.....	13
2.3. EMBEDDING OKKAM IDENTIFIERS IN HTML	13
2.4. A SIMPLE OKKAM ONTOLOGY.....	14
2.4.1. Class: Okkam Entity (okkam:OkkamEntity)	14
2.4.2. Property: Okkam ID (okkam:okkamID).....	14
2.5. DESCRIPTION OF AN OKKAM ENTITY IN RDF	15
2.6. OKKAM ENTITIES IN HTML VIA RDFA	15
2.6.1. Example: Full page Markup:.....	15
2.6.2. Example: Multiple entities, invisible okkam markup, visible manifestation:	16
2.6.3. Example: Multiple entities, visible manifestation and markup	17
2.6.4. Example: Using Okkam URIs	18
2.6.5. Example: Advanced Entity markup.....	19
2.7. SUGGESTED PROPERTIES FOR VISIBLE MANIFESTATION OF OKKAM ENTITIES.....	20
3. EFFICIENT HARVESTING AND PROCESSING OF LARGE DATASETS WITH SEMANTIC SITEMAPS.....	22
3.1. THE SITEMAP PROTOCOL AND ROBOTS .TXT.....	23
3.2. THE STATE OF RDF PUBLISHING.....	23
3.2.1. Access methods to RDF data	23
3.2.2. Current Limitations.....	25
3.2.3. Related Work.....	25
3.3. THE SEMANTIC SITEMAPS EXTENSION	26
3.3.1. Datasets	26
3.3.2. Adding datasets descriptions to the Sitemap protocol	27
3.3.3. Other elements	28
3.3.4. Defining the DESCRIBE operator	28
3.3.5. Sitemaps and Authority.....	29
3.4. EVALUATION.....	30
3.4.1. Processing of Large RDF Dumps	31
3.5. ADOPTION	32
3.6. SUMMARY	33
4. LARGE SCALE PREPROCESSING OF HARVESTED DATA: REASONING AT WEB SCALE.....	34
4.1. CONTEXTS ON THE SEMANTIC WEB	34
4.2. IMPORT CLOSURE OF SEMANTIC DOCUMENTS.....	35
4.3. DEDUCTIVE CLOSURE OF SEMANTIC DOCUMENTS	36
4.4. CONTEXT DEPENDENT REASONING PROCEDURE.....	37
4.5. ONTOLOGY BASE CONCEPTS.....	37
4.6. ONTOLOGY BASE UPDATE STRATEGY.....	38
4.7. ONTOLOGY BASE QUERYING STRATEGY.....	39

4.8.	PROTOTYPE IMPLEMENTATION.....	39
4.8.1.	<i>Experimental Setup</i>	40
4.8.2.	<i>Preliminary Results</i>	40
4.9.	RELATED WORK.....	41
4.10.	SUMMARY.....	41
5.	HARVESTED DATA STATISTICS	42
5.1.	SUMMARY STATISTICS	42
5.2.	DISTRIBUTION OF DOCUMENT SIZES	42
5.3.	TOP DOMAINS	43
5.4.	OBJECTS OF RDF TRIPLES	44
6.	ACKNOWLEDGEMENTS.....	46
7.	APPENDIX 1: THE OKKAM ONTOLOGY IN RDF/XML	47
8.	REFERENCES.....	48

1. Introduction: Advanced Web Data Harvesting and Processing Techniques

In this document we present the works that have been conducted on harvesting and preprocessing web data.

For the purpose of the Entity Centric Semantic Search Engine (ECSSE), we have concentrated our efforts on the collection and indexing of the Web of Data: the RDF world and the world of web pages annotated with Microformats and or RDFa (also see D8.1).

This sort of data collection is relatively unexplored. For this reason we have developed several innovative techniques to speed up both the collection of large datasets and the later processing, e.g. the reasoning processes which are fundamental to have a meaningful precision and recall in the later query process.

Also, presented early in the chapter, in this document we define a format to embed Okkam identifiers in Web pages, an important step for data publishers to have their data automatically aggregated and reused by Okkam aware applications.

The document is so divided:

In **chapter 2**, we present a specification to embed Okkam Identifiers into documents on the Web, such as HTML pages. Thanks to this specification, the Sindice infrastructure can collect and index Okkam identifiers associated with entities represented in Web pages and therefore provide its services to implement the Entity Centric Search Engine. The chapter also presents our mappings that harmonize data written using microformats to RDF.

In **chapter 3**, we illustrate a technique we have developed for a great increase in efficiency of harvesting and processing of large web datasets (Semantic Sitemaps), as compared to raw data crawling. A report of implementation is also given.

In **chapter 4**, we present a technique for reasoning over large amounts of structured Web Data. The technique is then implemented using Hadoop.

We conclude in **chapter 5** with the statistics of the Semantic Data currently collected inside the Sindice infrastructure.

2. Okkam Entities and Microformats to RDF

The core dataset behind the Entity Centric Semantic Search Engine, is composed by tens of millions of structured data sources indexed by our peculiar Semantic Indexing technologies (see D8.1 for more details).

In this chapter we present the specifications which allow us to harvest and uniformly index:

- RDF
- Microformats
- Okkam Entities

Given that RDF is the native format of the Sindice index, two aspects need to be addressed; how to convert Microformats to RDF (section 2.1) and how to encode Okkam identifiers in HTML pages so that they can be harvested as RDF (section 2.3).

2.1. Converting Microformats to RDF inside Sindice and ECSSE

Microformats parsed out of HTML pages are converted into an RDF representation. In this way, one needs not to bother about the original data format but can simply query for the right RDF properties.

Note that this means that in general queries can be made in a way that is agnostic to the original semantic representation format, simply asking for the right RDF mapped Microformat class. It is possible, however, to restrict a query to get only documents that were originally obtained from a microformat-enabled HTML page, or a page containing RDFa.

2.2. General HTML extraction

An HTML page can contain any number of microformats. If more than one is detected, the resulting RDF graph simply contains the RDF conversion of all the microformats, plus the RDFa found in it.

If at least one microformat is detected, then the HTML `<title>` is also extracted and added to the page URI with `dc:title`, and subsequent extraction pass are run over it. See [Extraction strategy].

Each format is tagged with the **format** metadata field. Each microformat has a specific name, which will be used here. If any microformat has been found at all, MICROFORMAT will also be added as a special catch-all format.

2.2.1. XFN

XFN¹ (XHTML Friends Network) allows markup of hyperlinks as typed links that represent a human relationship between the authors or owners of two web pages. A `rel` attribute is added to the

¹ <http://gmpg.org/xfn/>

HTML hyperlink, e.g. `rel="contact"`, `rel="sweetheart"`, `rel="co-worker"`. A special case is `rel="me"` which connects different pages owned or created by the same person. This means that each link basically just transforms into a triple. In RDF:

An XFN me link from page.html to other.html in RDF

```
[
  a foaf:Person;
  foaf:isPrimaryTopicOf <page.html>;
  foaf:isPrimaryTopicOf <other.html>;
] .
```

An XFN friend link from page.html to other.html in RDF

```
[
  a foaf:Person;
  foaf:isPrimaryTopicOf <page.html>;
  xfn:friend [
    a foaf:Person;
    foaf:isPrimaryTopicOf <other.html>;
  ];
] .
```

The `xfn:` namespace expands to <http://gmpg.org/xfn/11#> (which, technically is namespace squatting).

We expand the XFN domain by adding direct triples, in the form

```
<uri.html> xfnrdf:friend <other.html>
<uri.html> xfnrdf:me <other.html>
<uri.html> xfnrdf:sweetheart <other.html>
```

Where **xfnrdf** is a vocabulary (namespace <http://vocab.sindice.com/xfn>) described in a separate document². This is actually done via the same code, so the mapping is 1:1.

Example searches that may/should return microformats (term, advanced):

```
Joe format:XFN
* <http://sindice.com/exfn/0.1/met-hyperlink> *
class:Person
```

2.2.2. hCard

We rely on the hcard profile³, which basically uses the vCard ontology.

Given something like:

```
<div class="vcard">
  <a class="fn org url" href="http://www.commerce.net/">CommerceNet</a>
  <div class="adr">
    <span class="type">Work</span>:
    <div class="street-address">169 University Avenue</div>
```

² <http://vocab.sindice.com/xfn/guide.html>

³ <http://www.w3.org/2006/03/hcard>

```

    <span class="locality">Palo Alto</span> ,
    <abbr class="region" title="California">CA</abbr>
    <span class="postal-code">94301</span>
    <div class="country-name">USA</div>
</div>
<div class="tel">
  <span class="type">Work</span> +1-650-289-4040
</div>
<div class="tel">
  <span class="type">Fax</span> +1-650-289-4041
</div>
</div>

```

We get

```

@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix v:      <http://www.w3.org/2006/vcard/ns#> .

```

```

[]    rdf:type v:VCard ;
      v:adr    [ rdf:type v:Address ;
                  v:countryName "USA" ;
                  v:locality "Palo Alto" ;
                  v:postalCode "94301" ;
                  v:region "CA" ;
                  v:streetAddress "169 University Avenue"
                ] ;
      v:fn     "CommerceNet" ;
      v:org    [ rdf:type v:Organization ;
                  v:organization-name "CommerceNet"
                ] ;
      v:tel    <tel:+1-650-289-4041> , <tel:+1-650-289-4040> ;
      v:url    <http://www.commerce.net/> .

```

Example queries:

```

format:HCARD
class:VCard class:Organization
* <http://www.w3.org/2006/vcard/ns#given-name> "Joe"

```

2.2.3. Geo

Geo uses the same ontology as VCard but just defines some of the types in that ontology. So for example you can ask for

```

class:Location
format:GEO
* <http://www.w3.org/2006/vcard/ns#latitude> "51.5217"

```

2.2.4. Adr

As above, Adr just comprises the Address class of the VCard ontology. So you can ask for

```

class:Address
format:ADR
* <http://www.w3.org/2006/vcard/ns#country-name> "Germany"

```

2.2.5. hCalendar

Based on the RDF Calendar spec⁴. Given:

```
<div class="vevent">
  <a class="url" href="http://www.web2con.com/">http://www.web2con.com/</a>
  <span class="summary">Web 2.0 Conference</span>:
  <abbr class="dtstart" title="2007-10-05">October 5</abbr>-
  <abbr class="dtend" title="2007-10-20">19</abbr>,
  at the <span class="location">Argent Hotel, San Francisco, CA</span>
</div>
```

We get:

```
@prefix r:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix c:      <http://www.w3.org/2002/12/cal/icaltzd#> .
<>   r:type    c:Vcalendar ;
      c:component
      [ r:type    c:Vevent ;
        c:dtend  "2007-10-20"^^<http://www.w3.org/2001/XMLSchema#date> ;
        c:dtstart "2007-10-05"^^<http://www.w3.org/2001/XMLSchema#date> ;
        c:location "Argent Hotel, San Francisco, CA" ;
        c:summary  "Web 2.0 Conference" ;
        c:url      <http://www.web2con.com/>
      ] .
```

Usually, calendar components (**Vevent**, **Vtodo**, **Vjournal**, **Vfreebusy**) occur within a surrounding **Vcalendar** entity. This surrounding Vcalendar block is optional in hCard. If it is not present, an implicit Vcalendar scoped to the entire document will be assumed and added to the RDF.

So calendars can be found with queries like:

```
class:Vcalendar
Eric Clapton class:Vevent
Live show format:HCALENDAR
opera ontology:icaltzd
* <http://www.w3.org/2002/12/cal/icaltzd#dtstart> "20080527T1900+0100"
```

2.2.6. hReview

hReview is still in draft, but seems interesting and some people use it. The reference specification is RDF Review⁵. Typical RDF output:

```
@prefix dc:      <http://purl.org/dc/elements/1.1/> .
@prefix review:  <http://www.purl.org/stuff/rev#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

[ ]   rdf:type  review:Review ;
      dc:title  "A wonderful night on acid at MoS!" ;
      review:rating "5" .
```

⁴ <http://www.w3.org/TR/rdfcal/>

⁵ <http://dannyayers.com/xmlns/rev/>

Queries for finding reviews:

```
sigur ros class:Review
ontology:rev ipod
format:HREVIEW
```

Notice that the RDF Review ontology actually has some deployment in its normal RDF form.

2.2.7. rel-license

Extracts the license links (simple **rel="license"**) and transforms them into

```
<this.html> dcterms:license <someLicense.html>
```

Multiple licenses are extracted correctly as expected.

Simple query:

```
jazz format:LICENSE
* <http://purl.org/dc/terms/license> "http://creativecommons.org/licenses/publicdomain/"
```

2.2.8. hListing

hListing is unstable, and we extract it tentatively, but as of now we are still working on a proper ontology for it.

The current parser accommodates this by being extremely liberal. As the format stabilizes we can restrict the rules.

The vocabulary used is namespaced as <http://sindice.com/hlisting/0.1/>. For a listing at kelkoo.com it gets data like this:

```
@prefix hl:    <http://sindice.com/hlisting/0.1/> .

[]    a hl:Listing ;
      hl:action hl:offer;
      hl:description "...";
      hl:item [ a hl:item ;
                hl:itemName "Benq MP622 - DLP Projector - 2700 ANSI lumens - XGA..." ;
                hl:itemPhoto
"http://img.kelkoo.com/uk/medium/675/496/00117250662929509422269096808645163496675.jpg" ;
                hl:itemUrl "http://bob.example.com/"
              ] ;
      hl:lister
      [ a hl:Lister ;
        hl:listerLogo
          "http://bob.example.com/data/merchantlogos/4621623/pcworld.gif" ;
        hl:listerName "PC World Business" ;
        hl:listerOrg "PC World Business" ;
        hl:listerUrl "http://bob.example.com/m-4621623-pc-world-business.html"
      ] ;
      hl:price "£480.17".
```

2.2.9. hResume

In HTML an hResume is a special combination of hCards and hCalendars. Thus we extract both the calendar and the vCards.

We also transform the resume using the DOAC vocabulary. As an example of a page on linkedin, notice the overlapping informations:

```
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://www.linkedin.com/in/grenzi>
  <http://purl.org/dc/elements/1.1/title>
    "Gabriele Renzi - LinkedIn" .

[]  rdf:type <http://xmlns.com/foaf/0.1/Person> ;
    <http://ramonantonio.net/doac/0.1/#affiliation>
      [ rdf:type <http://www.w3.org/2006/vcard/ns#VCard> ;
        <http://www.w3.org/2006/vcard/ns#fn>
          "stacktrace member" ;
        <http://www.w3.org/2006/vcard/ns#logo>
          <http://media.linkedin.com/media/p/1/000/00a/29d/0844af4.png> ;
        <http://www.w3.org/2006/vcard/ns#org>
          [ rdf:type <http://www.w3.org/2006/vcard/ns#Organization> ;
            <http://www.w3.org/2006/vcard/ns#organization-name>
              "stacktrace member"
            ] ;
        <http://xmlns.com/foaf/0.1/topic>
          [ <http://xmlns.com/foaf/0.1/name>
              "stacktrace member"
            ]
          ] ;
    <http://ramonantonio.net/doac/0.1/#affiliation>
      [ rdf:type <http://www.w3.org/2006/vcard/ns#VCard> ;
        <http://www.w3.org/2006/vcard/ns#fn>
          "Professionisti @ Ruby member" ;
        <http://www.w3.org/2006/vcard/ns#logo>
          <http://media.linkedin.com/media/p/3/000/00a/361/1a69bc0.png> ;
        <http://www.w3.org/2006/vcard/ns#org>
          [ rdf:type <http://www.w3.org/2006/vcard/ns#Organization> ;
            <http://www.w3.org/2006/vcard/ns#organization-name>
              "Professionisti @ Ruby member"
            ] ;
        <http://xmlns.com/foaf/0.1/topic>
          [ <http://xmlns.com/foaf/0.1/name>
              "Professionisti @ Ruby member"
            ]
          ] ;
    <http://ramonantonio.net/doac/0.1/#affiliation>
      [ rdf:type <http://www.w3.org/2006/vcard/ns#VCard> ;
        <http://www.w3.org/2006/vcard/ns#fn>
          "Forbes.com Personal Technology Forum member" ;
        <http://www.w3.org/2006/vcard/ns#logo>
          <http://media.linkedin.com/media/p/1/000/000/005/0a596c4.gif> ;
        <http://www.w3.org/2006/vcard/ns#org>
          [ rdf:type <http://www.w3.org/2006/vcard/ns#Organization> ;
            <http://www.w3.org/2006/vcard/ns#organization-name>
              "Forbes.com Personal Technology Forum member"
            ] ;
        <http://xmlns.com/foaf/0.1/topic>
          [ <http://xmlns.com/foaf/0.1/name>
              "Forbes.com Personal Technology Forum member"
            ]
          ]
      ] ;
  ] ;
```

```

<http://ramonantonio.net/doac/0.1/#summary>
  "Coding Monkey, Geek, Professional Student" ;
<http://xmlns.com/foaf/0.1/isPrimaryTopicOf>
  [ rdf:type <http://www.w3.org/2006/vcard/ns#VCard> ;
    <http://www.w3.org/2006/vcard/ns#adr>
      [ rdf:type <http://www.w3.org/2006/vcard/ns#Address> ;
        <http://www.w3.org/2006/vcard/ns#locality>
          "Rome Area, Italy"
        ] ;
    <http://www.w3.org/2006/vcard/ns#fn>
      "Gabriele Renzi" ;
    <http://www.w3.org/2006/vcard/ns#n>
      [ rdf:type <http://www.w3.org/2006/vcard/ns#Name> ;
        <http://www.w3.org/2006/vcard/ns#family-name>
          "Renzi" ;
        <http://www.w3.org/2006/vcard/ns#given-name>
          "Gabriele"
        ] ;
    <http://www.w3.org/2006/vcard/ns#title>
      "Coding Monkey, Geek, Professional Student" ;
    <http://www.w3.org/2006/vcard/ns#url>
      <http://www.riffraff.info> , <http://riffraff.blogspot.com> ;
    <http://xmlns.com/foaf/0.1/topic>
      [ <http://xmlns.com/foaf/0.1/name>
        "Gabriele Renzi"
      ]
    ] .

```

```

[] rdf:type <http://xmlns.com/foaf/0.1/Person> ;
<http://xmlns.com/foaf/0.1/isPrimaryTopicOf>
  <http://www.linkedin.com/in/grenzi> , <http://www.riffraff.info> ;
<http://xmlns.com/foaf/0.1/weblog>
  <http://www.linkedin.com/in/grenzi> , <http://www.riffraff.info> .

```

So searches like this are possible:

```

* http://ramonantonio.net/doac/0.1/#organization 'something'
format:HRESUME

```

2.3. Embedding OKKAM identifiers in HTML

In the OKKAM Entity Naming System vision, we foresee that participating agents (e.g. a companies or individual) will interact directly with the OKKAM system to obtain or negotiate an ID for internal entities. When this interaction is complete, it is useful for the agent to make sure that the okkam ID is associated with entity manifestations, e.g. with a web page or a fragment of a page that is intended to be associated univocally with the entity. This in practice calls for a markup technique to be able to embed the okkam ID for the entity into the page HTML or in a way which is associated to Web access. In this document we will illustrate a method based on RDFa.

RDFa provides a set of HTML attributes to augment visual data with machine-readable hints. Technically, RDFa allows the embedding of RDF information inside an XHTML document so that values that are displayed to the users are at the same time used as literals inside the embedded RDF.

Making so that the same data is both visualized and made available for machine consumption is very important toward ensuring that RDF data is relevant and correct (also known as [principle of visibility](#) in the Microformat domain). While this principle is not per se enforced or respected by RDFa, the examples we provide in this chapter will instead do so.

Microformats add machine-readable hints to HTML pages by overloading existing HTML elements and attributes. This results in various accessibility issues and potential semantic clashes. Given that RDFa extends XHTML with new attributes, it does not suffer from those problems and therefore is a technically superior choice. This comes at a slight cost: Validation of documents containing RDFa markup is currently only possible if the document is XHTML 1.1, however such documents work perfectly fine in both browsers and search engines, even though validators that are not aware of RDFa will reject them. At the same time, using RDFa allows the use of arbitrary ontologies to express data, a concept that has always been welcome on the W3C Semantic Web community.

In the next section we will first define an ontology for embedding Okkam identifiers and then show examples on how this ontology can be used within HTML documents

2.4. A simple Okkam Ontology

It is a fundamental principle to reuse existing ontologies where possible to describe entities. RDF itself is meant for generic entity (resource) description so we do not see the reason to specify anything more than a class name that will help an Agent to recognize that the URI used to describe an entity is in fact to be intended as an Okkam ID. Please notice that this could also be achieved by looking at the URI itself (e.g. in case it begins with <http://okkam.org>) but the use of a class ensures that in the future multiple Okkam providers, possibly using different URI prefixes, will be allowed to mint Okkam IDs.

The namespace “okkam” is assumed to be bound to: <http://okkam.org/terms#>

An RDF/XML version of this ontology can be found in Appendix 1.

2.4.1. Class: Okkam Entity (`okkam:OkkamEntity`)

Full URI: <http://okkam.org/terms#OkkamEntity>

Status: unstable

Comment: Any entity that can be identified using the OKKAM entity naming system. Note that there is no requirement that the entity be identified using an okkam.org URI. However, it must have exactly one `okkam:okkamID` property.

2.4.2. Property: Okkam ID (`okkam:okkamID`)

Full URI: <http://okkam.org/terms#okkamID>

Status: unstable

Comment: The `okkam ID` of the entity, given as a plain literal. This is an Inverse Functional Property and has maximum and minimum cardinalities of 1, that is, if an entity is of type `OkkamEntity` then it must have exactly one `okkamID` property.

Having this property implies being an `okkam:okkamID`, the cardinality is, forcefully, 1. Every value of this property is a `rdfs:Literal`. The property is a sub-property of [`dct:identifier`](#).

2.5. Description of an Okkam entity in RDF

In RDF terms, an Okkam entity is a marked up RDF node which must fulfill at least one of the following conditions:

1. It has an `okkam:okkamID` property set with a value returned by the Okkam infrastructure
2. It is of type `okkam:OkkamEntity` AND its identifier is a resolvable URI which, when resolved, returns RDF data that has the `okkam:okkamID` property set. This could, for example, simply be the URI of the identifier on the OKKAM system.

When both a resolvable URI and a direct value for the `okkam:okkamID` property is given, then the `okkam:okkamID` value obtained by resolving the URI must match the directly given value.

Valid Okkam entity (untyped, with `okkam:okkamID`, identified with URI):

```
<http://joeblog.org/index.html#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<foaf:Person> .
<http://joeblog.org/index.html#me> <http://okkam.org/terms#okkamID> "7539713471" .
```

Valid Okkam entity (untyped, with `okkam:okkamID`, blank node):

```
_:n0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <foaf:Person> .
_:n0 <http://okkam.org/terms#okkamID> "7539713471" .
```

Valid Okkam entity (typed, with resolvable URI):

```
<http://okkam.org/ID/123123123123> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
<http://okkam.org/ID/123123123123> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<foaf:Person>
```

This is valid since Okkam ID links are in fact valid linked data links, that is, <http://okkam.org/ID/123123123123> can be configured to return, when resolved, the `okkam:okkamID` triple that is needed to complete the entity:

```
<http://okkam.org/ID/123123123123> <http://okkam.org/terms#okkamID> "123123123123"
```

Invalid Okkam entity (typed, *without* resolvable URI):

```
<http://joeblog.org/index.html> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
<http://joeblog.org/index.html> <http://okkam.org/terms#okkamID> "7539713471" .
<http://joeblog.org/index.html> <http://purl.org/dc/elements/1.1/creator> "Joe Blog" .
```

(The entity's URI here is of a web page, and resolving it will not return RDF containing an `okkam:okkamID` value.)

2.6. Okkam entities in HTML via RDFa

Withstanding the principles of the previous section, RDFa markup of OKKAM entities in HTML pages may come in different forms as the *manifestation of the entities* on a web page. Examples:

2.6.1. Example: Full page Markup:

A person named Joe Blog, puts a web page describing himself on the web. He wants this web page to have a unique and persistent ID in the Okkam system, so he embeds the okkam ID of the page in

the page's XHTML code. Joe does not want the Okkam identifier to be visible on the page. In this case, the most appropriate location is in a <meta> header in the page's <head> section.

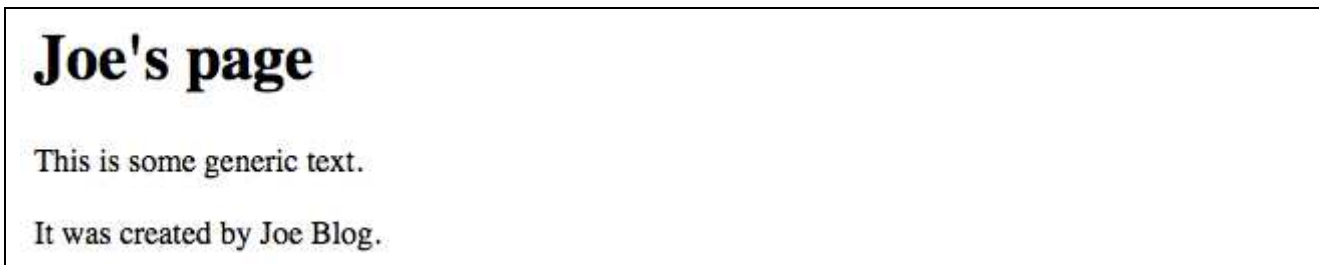
XHTML code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
"http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:okkam="http://okkam.org/terms#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
>
  <head>
    <title>Joe's page</title>
    <meta property="okkam:okkamID" content="7539713471" />
  </head>
  <body>
    <h1>Joe's page</h1>
    <div about="">
      <p>This is some generic text.</p>
      <p>It was created by <span property="dc:creator">Joe Blog</span>.</p>
    </div>
  </body>
</html>
```

N3 output:

```
<http://joeblog.org/index.html> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
<http://joeblog.org/index.html> <http://okkam.org/terms#okkamID> "7539713471" .
<http://joeblog.org/index.html> <http://purl.org/dc/elements/1.1/creator> "Joe Blog" .
```

Rendering:



2.6.2. Example: Multiple entities, invisible Okkam markup, visible manifestation:

This example has a page listing two persons. Each of the persons has a unique OKKAM identifier, which is invisibly embedded in the page, but also has a visible manifestation (E.g. the person's name). It is important that the visible manifestation be linked with the entity node, e.g. via literal property such as rdfs:label, so to establish a connection between the visual element and the entity itself. For a list of useful properties that can be used to markup visible entity manifestations to the RDF node see section 2.7.

XHTML code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
"http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:okkam="http://okkam.org/terms#"
  xmlns:rdfs="http://www.w3.org/2003/06/sw-vocab-status/ns#"
>
```

```

<head>
  <title>test</title>
</head>
<body>
  <h1>Two Unrelated Guys</h1>
  <ul>
    <li typeof="okkam:OkkamEntity" property="okkam:okkamID" content="86753429879">
      <span property="rdfs:label">Joe Blog</span>
    </li>
    <li typeof="okkam:OkkamEntity" property="okkam:okkamID" content="23423477234">
      <span property="rdfs:label">Jenny Penny</span>
    </li>
  </ul>
</body>
</html>

```

N3 output:

```

_:n0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
_:n0 <http://okkam.org/terms#okkamID> "86753429879" .
_:n0 <http://www.w3.org/2003/06/sw-vocab-status/ns#label> "Joe Blog" .

_:n1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
_:n1 <http://okkam.org/terms#okkamID> "23423477234" .
_:n1 <http://www.w3.org/2003/06/sw-vocab-status/ns#label> "Jenny Penny" .

```

Rendering:

Two Unrelated persons

- Joe Blog
- Jenny Penny

2.6.3. Example: Multiple entities, visible manifestation and markup

A page which lists two persons, each of the persons has a unique Okkam identifier, which is also displayed on the page.

XHTML code:

```

<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:okkam="http://okkam.org/terms#"
  xmlns:rdfs="http://www.w3.org/2003/06/sw-vocab-status/ns#"
>
<head>
  <title>test</title>
</head>
<body>
  <h1>Two Unrelated Guys</h1>
  <ul>
    <li typeof="okkam:OkkamEntity">
      <span property="rdfs:label">Joe Blog</span>, with this OkkamID: <span
property="okkam:okkamID">86753429879</span>
    </li>
    <li typeof="okkam:OkkamEntity">

```

```

        <span property="rdfs:label">Jenny Penny</span>, with this OkkamID: <span
property="okkam:okkamID">98765653475</span>
    </li>
</ul>
</body>
</html>

```

N3 output:

```

_:n0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
_:n0 <http://www.w3.org/2003/06/sw-vocab-status/ns#label> "Joe Blog" .
_:n0 <http://okkam.org/terms#okkamID> "86753429879" .

_:n1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
_:n1 <http://www.w3.org/2003/06/sw-vocab-status/ns#label> "Jenny Penny" .
_:n1 <http://okkam.org/terms#okkamID> "98765653475" .

```

Rendering:

Two Unrelated Guys

- Joe Blog, with this OkkamID: 86753429879
- Jenny Penny, with this OkkamID: 98765653475

2.6.4. Example: Using Okkam URIs

This example has a page which lists two persons. Each person is linked to the Okkam system, allowing easy lookup of additional information. Please note that this in fact does not use Okkam identifiers, but instead uses full Okkam URIs for the entities.

According to the linked data principles, you can then resolve the URIs to obtain the `okkam:okkamID` properties.

XHTML code:

```

<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:okkam="http://okkam.org/terms#"
  xmlns:rdfs="http://www.w3.org/2003/06/sw-vocab-status/ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
>
<head>
  <title>test</title>
</head>
<body>
  <h1>Two Unrelated Guys</h1>
  <ul>
    <li>
      <a typeof="okkam:OkkamEntity" href="http://okkam.org/ID/86753429879">
        <span property="rdfs:label">Joe Blog</span>
      </a>
    </li>
    <li typeof="okkam:OkkamEntity" about="http://okkam.org/ID/98746359387">
      <a href="http://okkam.org/ID/98746359387">
        <span property="rdfs:label">Jenny Penny</span>
      </a>
    </li>
  </ul>

```

```
</ul>
</body>
</html>
```

N3 output:

```
<http://okkam.org/ID/86753429879> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
<http://okkam.org/ID/86753429879> <http://www.w3.org/2003/06/sw-vocab-status/ns#label>
"Joe Blog" .
```

```
<http://okkam.org/ID/98746359387> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
<http://okkam.org/ID/98746359387> <http://www.w3.org/2003/06/sw-vocab-status/ns#label>
"Jenny Penny" .
```

Rendering:

Two Unrelated Guys

- [Joe Blog](#)
- [Jenny Penny](#)

2.6.5. Example: Advanced Entity markup

This example has a page with collaboratively created text. The text has two authors, and they both additionally know each other. This example shows how using RDFa one can express more complex relationships among entities.

XHTML code:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:okkam="http://okkam.org/terms#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdfs="http://www.w3.org/2003/06/sw-vocab-status/ns#"
>
  <head>
    <title>Collaborative text</title>
  </head>
  <body>
    <h1>Collaborative text</h1>
    <div about="" property="okkam:okkamID" content="7539713471"
  typeof="okkam:OkkamEntity">
      <p>This text has been created by multiple authors.</p>
      <div rel="dc:creator"> Authors:
        <ul>
          <li typeof="foaf:Person" about="http://okkam.org/ID/86753429879">
            <a property="okkam:okkamID" content="86753429879"
  href="http://okkam.org/ID/86753429879">
              <span property="rdfs:label">Joe Blog</span>
            </a>
            <span rel="foaf:knows"
  resource="http://okkam.org/ID/98746359387"/>
          </li>
          <li typeof="foaf:Person" about="http://okkam.org/ID/98746359387">
            <a property="okkam:okkamID" content="98746359387"
  href="http://okkam.org/ID/98746359387">
              <span property="rdfs:label">Jenny Penny</span>
            </a>
```

```

                <span rel="foaf:knows"
resource="http://okkam.org/ID/86753429879" />
                </li>
            </ul>
        </div>
    </div>
</body>
</html>

```

N3 output:

```

<http://joeblog.org/index.html> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://okkam.org/terms#OkkamEntity> .
<http://joeblog.org/index.html> <http://okkam.org/terms#okkamID> "7539713471" .
<http://joeblog.org/index.html> <http://purl.org/dc/elements/1.1/creator>
<http://okkam.org/ID/86753429879> .
<http://joeblog.org/index.html> <http://purl.org/dc/elements/1.1/creator>
<http://okkam.org/ID/98746359387> .

<http://okkam.org/ID/86753429879> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .
<http://okkam.org/ID/86753429879> <http://okkam.org/terms#okkamID> "86753429879" .
<http://okkam.org/ID/86753429879> <http://www.w3.org/2003/06/sw-vocab-status/ns#label>
"Joe Blog" .
<http://okkam.org/ID/86753429879> <http://xmlns.com/foaf/0.1/knows>
<http://okkam.org/ID/98746359387> .

<http://okkam.org/ID/98746359387> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .
<http://okkam.org/ID/98746359387> <http://okkam.org/terms#okkamID> "98746359387" .
<http://okkam.org/ID/98746359387> <http://www.w3.org/2003/06/sw-vocab-status/ns#label>
"Jenny Penny" .
<http://okkam.org/ID/98746359387> <http://xmlns.com/foaf/0.1/knows>
<http://okkam.org/ID/86753429879> .

```

Rendering:

Collaborative text

This text has been created by multiple authors.

Authors:

- [Joe Blog](#)
- [Jenny Penny](#)

2.7. Suggested properties for visible manifestation of Okkam Entities

The method for embedding Okkam IDs in HTML pages described here builds on RDFa. This allows very flexible and extensible descriptions of the Okkam entities to be embedded in the pages. In theory, any RDF property from any RDF vocabulary can be used to further describe and annotate entities. The following table contains a small selection of properties that are especially useful when marking up visible manifestations of entities in the HTML. Doing so is important to provide visual feeds about entities by browser plugins and similar.

In general, it can be said that the Dublin Core Metadata Terms⁶ are a good choice for providing information about entities.

rdfs:label	To mark the main label of the entity
dc:abstract	To mark a part of descriptive text about the resource
dc:description	Description may include but is not limited to: an abstract, a table of contents, a graphical representation, or a free-text account of the resource. More general than dc:abstract.
dc:created	Date of creation of the resource

⁶ <http://dublincore.org/documents/dcmi-terms/>

3. Efficient Harvesting and Processing of large datasets with Semantic Sitemaps

Data on the Semantic Web can be made available and consumed in many different ways. For example, an online database might be published as one single RDF dump. Alternatively, the recently proposed Linked Data paradigm is based on using resolvable URIs as identifiers to offer access to individual descriptions of resources within a database by simply resolving the address of the resources itself (Berners-Lee, 2006). Other databases might offer access to its data via a SPARQL endpoint that allows clients to submit queries using the SPARQL RDF query language and protocol.

If several of these options are offered simultaneously for the same database, the choice of access method can have significant effects on the amount of networking and computing resources consumed on the client and the server side.

For example, a Semantic Web search engine that wants to index an entire database might prefer to download the single dump file, instead of crawling the data piecemeal by fetching individual Linked Data URIs. A client interested in the definition of only a few DBpedia (Auer, Bizer, Kobilarov, Lehmann, Cyganiak, & Ives, 2007) resources would, on the other hand, be well-advised to simply resolve their URIs. But if the client wants to execute queries over the resources, it would be better to use the available SPARQL service.

Such choices can have serious implications: For example, on February the 2nd 2007, Geonames⁷ was hit by what appeared to be the first distributed denial of service attack against a Semantic Web site⁸.

What happened, however, was not a malicious attack but rather a Semantic Web crawler bringing down the site by rapid-firing requests to Geonames' servers up to the point where the site's infrastructure could not keep up. One could simply judge this as a case of inconsiderate crawler behaviour but, it highlights an important issue: What crawling rate should a crawler have on a Semantic Web site and would this be compatible with the size of the datasets? Considering that results are generated from semantic queries, it might be sensible to limit the query rate to one document per second, for example. It has to be noted however that crawling Geonames' 6.4M RDF documents would take 2.5 months under this condition, so that periodic recrawls to detect changes would take the same unsatisfactory amount of time and it would be impossible to have data in a fresh state.

Conceptually, the Geonames incident could have been avoided: Geonames offers a complete RDF dump of their entire database so this could have been bulk imported instead of crawling. But how is an automated crawler able to know about this dump and to know that the dump contains the entire Geonames database?

This chapter, adapted from Cyganiak et al 2008, describes a methodology with the primary goal to allow publishers to provide exactly such information and thus enable the smart selection of data access methods by clients and crawlers alike.

⁷ <http://geonames.org/>

⁸ <http://geonames.wordpress.com/2007/02/03/friendly-fire-semantic-web-crawler-ddos/>

The methodology is based on extending the existing Sitemap Protocol (section 2) by introducing several new XML tags for announcing the presence of RDF data and to deal with specific RDF publishing needs (section 3).

3.1. The Sitemap Protocol and robots.txt

The Sitemap Protocol⁹ defines a straightforward XML file for automatic agents (e.g. crawlers) that holds a list of URLs they should index. This is possible through tags that describe the location of each crawlable resource along with meta-information such as, for example, the expected rate of change for each individual URL or the date when this was last modified. An example of a sitemap is shown in the following listing:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/</loc>
    <lastmod>2005-01-01</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.8</priority>
  </url>
</urlset>
```

Once a sitemap has been created, it must be saved in a file on the server. The protocol defines a way to extend the robot.txt file, so that a robot can find the location of the sitemap file on a given site.

3.2. The State of RDF Publishing

Even though there is no universal agreement on how information should be best published on the Semantic Web, we can characterize some of the options that are in widespread use today; list some of the limitations imposed by those options; and look at existing proposals to address these limitations.

3.2.1. Access methods to RDF data

Several different access methods to RDF data are in widespread use today. They vary widely along several dimensions. Two dimensions warrant closer attention, namely discoverability and cost of access.

The RDF Web and Linked Data. The traditional World Wide Web can be characterized as the HTML Web. It consists of a large number of HTML documents made available via the HTTP protocol and connected by hyperlinks. By analogy, the RDF Web consists of the RDF documents that are available via HTTP. Each RDF document can be parsed into an RDF graph. Often, different documents share URIs, and therefore merging the documents results in a connected graph. Unlike HTML hyperlinks, statements in these RDF graphs usually do not connect the URIs of documents, but instead they connect the URIs of resources described in those documents.

RDF uses URIs to identify resources of interest, but does not prescribe any particular way of choosing URIs to name resources. Experience shows that it is a good idea to choose URIs so that a Web lookup (an HTTP GET operation) on the URI results in fetching the RDF document that describes them. This effect is typically achieved by either appending a fragment identifier (e.g.

⁹ <http://www.sitemaps.org/protocol.php>

#me) to the document URI, or by employing HTTP's 303 See Other status code to redirect from the resource's URI to the actual document.

Datasets on the RDF Web are typically served so that resolving the URI of a resource will return only the RDF statements closely describing the resource, rather than resolving to a file containing the entire dataset. This is usually achieved by using a properly configured server which creates such descriptions on demand.

The cost of accessing this publishing method is usually relatively low. Descriptions of a single resource are usually much smaller than the description of an entire knowledge base. Similarly, the computational complexity for generating resource descriptions is limited, albeit non negligible.

RDF dumps. Like documents on the RDF Web, RDF dumps are serializations of an RDF graph made available on the Web. But they usually contain descriptions of a large number of resources in a single file, the resource identifiers usually do not resolve to the dump itself, and the file is often compressed. Many RDF datasets are published in this way on the Web and they cannot be easily browsed, crawled or linked into. It should be noticed, however, that:

- dumps are obviously useful to provide the entire knowledge behind a dataset at a single location (e.g. the Geonames dump as a single file to process it directly)
- RDF datasets that are not crawlable or do not contain resolvable resources may exist for perfectly valid technical reasons.
- Producing RDF dumps is technically less challenging than operating a SPARQL endpoint or serving Linked Data which makes RDF dumps a popular option.

Again, the cost of accessing this publishing method is computationally very low, as dumps are usually precomputed and can be easily cached. Since dumps represent the entire knowledge base, they can be however expensive in terms of network traffic and memory requirements for processing them.

SPARQL endpoints. SPARQL endpoints can provide descriptions of the resources described in a knowledge base and can further answer relational queries according to the specification of the SPARQL query language.

While this is the most flexible option for accessing RDF data, the cost of this publishing method is high:

- For simple tasks such as obtaining a single resource description, it is more involved than a simply resolving a URI. A query needs to be written according to the correct SPARQL syntax and semantics, the query needs to be encoded, and the results need to be parsed into a useful form.
- It leaves a Semantic Web database open to potential denials of service due to queries with excessive execution complexity.

Multiple access methods. It is a common scheme for publishers to provide large RDF datasets through more than one access method. This might be partially explained by the early state of the Semantic Web, where there is no agreement on the best access method; but it also reflects the fact that the methods' different characteristics enable fundamentally different applications of the same data. Our proposal embraces and formalizes exactly this approach.

3.2.2. Current Limitations

Here we list some limitations imposed by current RDF publishing practices which we address with the Semantic Sitemaps proposal.

Crawling performance. Crawling large Linked Datasets takes a long time and is potentially very expensive in terms of computing resources of the remote server.

Disconnected datasets. Not all datasets form a fully connected RDF graph. Crawling such datasets can therefore result in an incomplete reproduction of the dataset on the client side.

Scattered RDF files. To find scattered, poorly linked RDF documents, an exhaustive crawl of the HTML Web is necessary. This is likely not cost-effective for clients primarily interested in RDF, leading to missed documents.

Cataloging SPARQL endpoints. Even a full HTML Web crawl will not reveal SPARQL endpoints because support for the SPARQL protocol is not advertised in any way when requests are made to a specific SPARQL endpoint URI.

Discovering a SPARQL endpoint for a given resource. If all we have is a URI then we can resolve it to reveal some data about it. But how can we discover a potentially existing SPARQL endpoint for asking more complex queries about the resource?

Provenance. Provenance is a built-in feature of the Web, thanks to the grounding of HTTP URIs in the DNS. But control over parts of a domain is often delegated to other authorities. This delegation is not visible to the outside world.

Identifying RDF dumps. An HTML Web crawl will reveal links to many compressed archive files. Some of them may contain RDF data when uncompressed, but most of them will most likely contain other kinds of files. Downloading and uncompressing all of those dumps is likely not cost-effective.

Closed-world queries about self-contained data. RDF semantics is based on the open-world assumption. When consuming RDF documents from the Web, we can never be sure to have complete knowledge and therefore cannot with certainty give a negative answer to questions like “Does Example, Inc. have an employee named Eric?”

3.2.3. Related Work

Most of the problems listed above have emerged just recently due to an increase in the amount and diversity of data available on the Semantic Web. The emergence of generic protocols for accessing RDF, especially the SPARQL protocol and Linked Data, enables the development of generic clients such as Tabulator (Berners-Lee, 2006) letting the user explore RDF documents anywhere on the Web. Equipped with a SPARQL API and a working knowledge of the query language, developers can interrogate any SPARQL endpoint available on the Web. As a result of these standardized protocols and tools, talking to data sources has become much easier. By contrast, efficient discovery and indexing of Semantic Web resources has become an important bottleneck that needs to be addressed quickly. A number of protocols and ideas have been evaluated and considered to address this and will be discussed below.

It is clear that the problem of indexing datasets is tightly linked to the problem of indexing the Deep Web. Our Semantic Sitemaps proposal “piggy backs” on the Sitemap protocol, a successful existing approach to indexing the Deep Web...

Semantic Web Clients such as Disco¹⁰ and Tabulator (Berners-Lee, 2006) are also somewhat related since they could ideally make use of this specification, e.g., to locate SPARQL endpoints. Use cases to serve search engines such as Sindice (Tummarello, Oren, & Delbru, 2007), SWSE (Harth, Hogan, Delbru, Umbrich, O’Riain, & Decker, 2007) and Swoogle (Ding, et al., 2004) have been of primary importance in the development of these specifications.

There many examples of services which index or provide reference to collections of RDF documents using diverse methodologies. PingTheSemanticWeb¹¹, for example, works by direct notifications from data producers of every file put online or updated. SchemaWeb.info¹² offers a large repository of RDF documents submitted by users. Even Google provides a good amount of RDF files when asked specifically with a `filetype:rdf` query.

Related proposals include POWDER¹³, a method for stating assertions about sets of resources. A POWDER declaration can assert that all resources starting with a certain URI prefix are `dc:published` by a specific authority. In fact, a GRDDL transform could be used to turn a Semantic Sitemap into a POWDER declaration.

Finally, the Named Graph proposal provides a vocabulary for assertions and quotations with functionalities which can be used in conjunctions with these specifications (Carroll, Bizer, Hayes, & Stickler, 2005).

3.3. The Semantic Sitemaps Extension

We will start by clarifying the notion of a dataset, then list the key pieces of information that can be provided in a Semantic Sitemap, and finally look at two specific issues: obtaining individual resource descriptions from a large dump; and the topic of authority on the Semantic Web.

3.3.1. Datasets

A dataset represents a knowledge base which is a very useful notion for the following reasons:

- Different access methods, such as those mentioned above, can and should indeed be simply thought of as only “access methods” to the same knowledge base.
- The knowledge base is more than just the sum of its parts: by itself, it is of informational value whether a piece of information belongs to a dataset or not and queries using aggregates can make this explicit. For example, the question “Is Berlin the largest city mentioned in DBpedia?” cannot be answered by only getting the description of the resource itself (in case of Linked data though resolving their Berlin URI). On the contrary, such queries can be answered only the entire content of the dataset is available.

The Semantic Sitemap extension has the concept of dataset at its core: Datasets are well defined entities which can have one or more access methods. It is well defined what properties apply to a certain access method and what properties apply to a given dataset. Therefore properties that apply to that dataset will be directly related to all the data that can be obtained, independently from the access method.

¹⁰ <http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/>

¹¹ <http://pingthesemanticweb.com/>

¹² <http://schemaweb.info>

¹³ <http://www.w3.org/2007/powder/>

This statement of existence of an underlying dataset implies that in case of multiple offered access methods, then they will provide information consistent with each other. The specification mandate that this in fact must be the case, with the exceptions only limited to:

- operational issues such as delays in the publication of dumps.
- information that pertains only to a certain access method, such as `rdfs:seeAlso` statements that link together the documents of a linked data deployment.

A publisher can host multiple datasets on the same site and can describe them independently using different sections of the Semantic Sitemap. While there is nothing that prevents information overlap or contradictions between different datasets, it is expected that this is not the case.

3.3.2. Adding datasets descriptions to the Sitemap protocol

The Semantic Sitemap extension allows the description of a dataset via the tag `<sc:dataset>`, to be used at the same level as `<url>` tags in a regular sitemap. Access options for the datasets are given by additional tags such as `<sc:dataDump>`, `<sc:sparqlEndpoint>` and `<sc:linkedDataPrefix>`. If a sitemap contains several dataset definitions, they are treated independently. The following example shows a sitemap file applying the extension.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:sc="http://sw.deri.org/2007/07/sitemapextension">
  <sc:dataset>
    <sc:datasetLabel>
      Example Corp. Product Catalog
    </sc:datasetLabel>
    <sc:datasetURI>
      http://example.com/catalog.rdf#catalog</sc:datasetURI>
    <sc:linkedDataPrefix sc:slicing="subject-object">
      http://example.com/products/</sc:linkedDataPrefix>
    <sc:sampleURI>
      http://example.com/products/widgets/X42</sc:sampleURI>
    <sc:sampleURI>
      http://example.com/products/categories/all</sc:sampleURI>
    <sc:sparqlEndpoint sc:slicing="subject-object">
      http://example.com/sparql</sc:sparqlEndpoint>
    <sc:dataDump>
      http://example.com/data/catalogdump.rdf.gz</sc:dataDump>
    <sc:dataDump>
      http://example.org/data/catalog_archive.rdf.gz</sc:dataDump>
    <changefreq>weekly</changefreq>
  </sc:dataset>
</urlset>
```

The dataset is labelled as the *Example Corp. Product Catalog* and identified by `http://example.com/catalog.rdf#catalog`. Hence it is reasonable to expect further RDF annotations about the dataset `http://example.com/catalog.rdf`.

The “things” described in the dataset all have identifiers starting with `http://example.com/products/`, and their descriptions are served as Linked Data. A dump of the entire dataset is available, split into two parts and the publisher states that dataset updates can be expected weekly.

RDF dataset dumps can be provided in formats such as RDF/XML, N-Triples and N-Quads (same as N-Triples with a fourth element specifying the URI of the RDF document containing the triple; the same triple might be contained in several different documents). Optionally, dump files may be compressed in GZIP, ZIP, or BZIP2 format.

3.3.3. Other elements

Other interesting elements in the specifications include:

`<sc:sparqlGraphName>`. If this optional tag is present, then it specifies the URI of a named graph within the SPARQL endpoint. This named graph is assumed to contain the data of this dataset. This tag must be used only if `<sc:sparqlEndpointLocation>` is also present, and there must be at most one `<sc:sparqlGraphName>` per dataset. If the data is distributed over multiple named graphs in the endpoint, then the publisher should either use a value of `*` for this tag, or create separate datasets for each named graph. If the tag is omitted, the dataset is assumed to be available via the endpoint's default graph.

`<sc:datasetURI>`. An optional URI that identifies the current dataset. Resolving this URI may yield further information, possibly in RDF, about the dataset, but this is not required.

`<sc:datasetLabel>`. An optional label that provides the name of the dataset.

`<sc:sampleURI>`. This tag can be used to point to a URI within the dataset which can be considered a representative sample. This is useful for Semantic Web clients to provide starting points for human exploration of the dataset. There can be any number of sample URIs.

3.3.4. Defining the DESCRIBE operator

Publishing an RDF dataset as Linked Data involves creating many smaller RDF documents, each served as a response when accessing the URI itself via HTTP.

This is usually the result of a SPARQL DESCRIBE query performed giving as argument the resource's identifier. But the SPARQL specification does not specify how DESCRIBE queries are actually answered: it is up to the data provider to choose the amount of data to return as a description.

It is important that the Semantic Sitemap provides a way to specify how this description process is most likely to happen. Knowing this enables a process that we call "slicing" of the dataset, i.e. turning a single large data dump into many individual RDF models served online as description of the resource identifiers. This process is fundamental for creating large indexes of online RDF documents descriptions, as illustrated in section 5.1.

For this, the `<sc:linkedDataPrefix>` and `<sc:sparqlEndpointLocation>` tags can have an optional `sc:slicing` attribute taking a value from the list of slicing methods below and which mean that the description of a resource *X* includes:

- `subject`: all triples whose subject is *X*.
- `subject-object`: all triples whose subject or object is *X*.
- `CBD`: the Concise Bounded Description (Stickler, 2005) of *X*.
- `SCBD`: the Symmetric Concise Bounded Description (Stickler, 2005) of *X*.
- `MSG`: all the Minimal Self-Contained Graphs (Tummarello, Morbidoni, Puliti, & Piazza, 2005) involving *X*.

Publishers that want to use a slicing method that is not in the list should pick the value that most closely matches their desired method, or they may omit the `sc:slicing` attribute. If the slicing method is very different from any in the list, it is recommended to publish a dump in N-Quads format.

3.3.5. Sitemaps and Authority

While there is no official definition of authoritative information specifically for the Semantic Web, many currently agree on extending the standard definition given in *Architecture of the World Wide Web, Volume One* (Jacobs & Walsh, 2004), which links authority to the ownership of the domain name in which the URIs are minted.

For example, a piece of information coming from the owner of the domain `example.com` would be considered authoritative about the URI `http://example.com/resource/JohnDoe`. Following this definition, any information obtained by resolving the URI of information resources served as Linked Data can be considered authoritative, as it comes from the same domain where the URI is hosted.

However, no mechanism is currently defined to get information on such an authority: The only possibility for this is via DNS domain records, which are outside the actual Web architecture.

For this reason the Semantic Sitemap extension proposes an `<sc:authority>` element, which is used at the top level of a sitemap file to specify a URI identifying the person, organization or other entity responsible for the sitemap's URI space. This is only useful if the URI is resolvable and yields additional information about the authority. The authority URI has to be within the sitemap's URI space, which makes the authority responsible for providing their own description.

The semantics of `<sc:authority>` is such that, given an authority URI a , for any document d within the sitemap's URI space, there is an implied RDF statement d `dc:publisher` a . For example, if a sitemap file in the root directory of `http://example.com/` declares an `sc:authority` of `http://example.com/foaf.rdf#me`, then the entity denoted by that URI is considered to be the publisher of all documents whose URI starts with `http://example.com/`. It has to be kept in mind that publication does not necessarily imply endorsement; it merely means that a is responsible for making d available. To express a propositional attitude, like assertion or quotation, additional means are necessary, such as the Semantic Web Publishing Vocabulary (Carroll, Bizer, Hayes, & Stickler, 2005).

Delegation of authority. The boundaries of DNS domains do not always match the social boundaries of URI ownership. An authority sometimes delegates responsibility for chunks of URI space to another social entity. The Semantic Sitemap specification accounts for this pattern. If another sitemap file is placed into a sitemap's URI space, then the second sitemap's URI space is considered to be not under the authority of the first. For example, publishing a sitemap file at `http://example.com/~alice/sitemap.xml` delegates the subspace rooted at `http://example.com/~alice/`.

Furthermore, `robots.txt` must link to a sitemap index file that in turn links to both of the sitemap files. This ensures that all visitors get identical pictures of the site's URI space. The approach is limited: It only allows the delegation of subspaces whose URIs begins with the same URI as the secondary sitemap file.

This feature proves to be very useful in properly assigning authority to Semantic Web information published in popular URI spaces like `perl.org`.

Joining URI spaces. Besides partitioning of URI spaces, it is sometimes necessary to join URI spaces by delegating responsibility of additional URIs to an existing sitemap. This is particularly useful when a single dataset spans over multiple domains, for example, if the SPARQL endpoint is located on a different server. The joining of URI spaces is achieved by placing a sitemap file into each of the spaces and having the `<sc:subSitemap>` element point to the other's URI. The sub-sitemap also needs a reciprocating `<sc:parentSitemap>` element. This prevents fraudulent appropriation of URI spaces.

Authoritative descriptions from SPARQL endpoints and dumps. An RDF description of a URI u is considered authoritative if resolving u yields an RDF document containing that description. An authoritative description is known to originate directly from the party owning the URI u , and thus can be considered to be definitive with regard to the meaning of u .

Providing authoritative information thus requires the publication of RDF in the "RDF Web" style. Descriptions originating from SPARQL endpoints or RDF dumps cannot be considered authoritative, because it cannot be assumed that information about u from a SPARQL endpoint at URI e is indeed authorized by the owner of u . The presence of a Semantic Sitemap, however, changes this. If e and u and d are in the same sitemap's URI space, then they originate from the same authority, and therefore we can treat information from the SPARQL endpoint as authoritative with respect to u even if u is not resolvable. This has two benefits. First, it allows RDF publishers to provide definitive descriptions of their URIs even if they choose not to publish RDF Web-style documents. Second, it allows RDF consumers to discover authoritative descriptions of URIs that are not resolvable, by asking the appropriate SPARQL endpoint or by extracting from the RDF dump.

3.4. Evaluation

We now revisit the challenges identified in section 3.2.2 and will show how the Semantic Sitemaps proposal can address each of them.

Crawling performance. Crawling large Linked Data deployments takes a long time and is potentially very expensive in terms of computing resources of the remote server. An RDF publisher can make a dump of the merged Linked Data documents available and announce both the `<sc:linkedDataPrefix>` and `<sc:dataDump>` in a sitemap file. Clients can now discover and download the dump instead of crawling the site, thus dramatically reducing the required time.

Disconnected datasets. Not all datasets form a fully connected RDF graph. Crawling such datasets can result in an incomplete reproduction of the dataset on the client side. Sitemaps address this in two ways. Firstly, clients can again choose to download a dump if it is made available. Secondly, by listing at least one `sc:sampleURI` in each component of the graph, the publisher can help to ensure a complete crawl even if no dump is provided.

Scattered RDF files. To find scattered, poorly linked RDF documents, an exhaustive crawl of the HTML Web is necessary. This is likely not cost-effective for clients primarily interested in RDF, leading to missed documents. Site operators can provide an exhaustive list of those documents using `<sc:dataDump>` or `<sc:sampleURI>`.

Cataloging SPARQL endpoints. Even a full HTML Web crawl will not reveal SPARQL endpoints, because support for the SPARQL protocol is not advertised in any way when requests are made to a SPARQL endpoint URI. However, if a sitemap has been provided, the crawler can discover the services via `<sc:spqrqlEndpoint>`.

Discovering a SPARQL endpoint for a given resource. If all we have is a URI, we can resolve it to reveal some data about it. But to discover a potentially existing SPARQL endpoint for asking more

complex queries we can look for a sitemap on the URI's domain and inspect it for `<sc:sparqlEndpoint>` elements.

Provenance. Provenance is a built-in feature of the Web, thanks to the grounding of HTTP URIs in the DNS. But control over parts of a domain is often delegated to other authorities. Semantic Sitemaps allow site operators to make this delegation visible, by appropriate placement of multiple sitemap files, and also by employing `<sc:subSitemap>` and `<sc:parentSitemap>` elements.

Identifying RDF dumps. An HTML Web crawl will reveal links to many compressed archive files. Some of them may contain RDF data when uncompressed, but most of them will most likely contain other kinds of files. Downloading and uncompressing all of those dumps is likely not cost-effective. Semantic Sitemap files explicitly list the locations of RDF dumps in the `<sc:dataDump>` element and therefore allow crawlers to avoid the cost of inspecting dumps that turn out not to contain any RDF.

Closed-world queries about self-contained data. RDF semantics is based on the open-world assumption and hence, when consuming RDF documents from the Web, we can never be sure to have complete knowledge. So we cannot with certainty give a negative answer to the question “Does Example, Inc. have an employee named Eric?”. Datasets defined in Semantic Sitemaps are natural candidates for applying local closed-world semantics. Drawing on `<sc:datasetLabel>`, this allows us to give a stronger answer: “Example, Inc. has not given information about such an employee in the dataset labelled *Example, Inc. Employee List*.”

3.4.1. Processing of Large RDF Dumps

A major motivation for our proposal is its potential to reduce cost, both in raw time and computing resources, of indexing the largest RDF datasets currently available on the RDF Web. These datasets are usually also available as RDF dumps, and Semantic Sitemaps enable clients to download the dump and avoid crawling. Here we present the results of some experiments into quantifying the benefits of this approach and showing the general feasibility of processing very large RDF dumps. We employ the Hadoop framework¹⁴ for parallel data processing on a “mini” cluster of low cost machines (single core, 4 GB ram).

The issue with crawling large collections of RDF documents from a single host, such as the UniProt datasets available on <http://purl.uniprot.org>, is that a crawler has to space its request by a reasonable amount of time in order to avoid overloading the server infrastructure. Semantic Web servers are likely to be experimental and not optimized for high load. Therefore submitting more than one request per second seems to be unadvisable for RDF crawlers.

The UniProt dataset¹⁵ contains approximately 14M RDF documents. A full crawl at this rate would take more than five months, and a full re-crawl to detect updated documents would take another five months.

UniProt provides a dump (over 10 GB in size, compressed). The time for gathering the data can thus be reduced to downloading of the dump files, plus subsequent local data processing to slice the dumps into parts that are equivalent to the individual documents. Each part describes a single resource. The parts can be passed on to our indexing system grouped in .TAR.GZ files each containing 10000 individual RDF files.

¹⁴ <http://lucene.apache.org/hadoop/>

¹⁵ available from ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/rdf

The first step in our processing was to convert the dumps from the RDF/XML format to N-Triples. The line-based N-Triples format can be more easily processed with Hadoop's off-the-shelf classes. We observed processing rates of about 60k triples/s for this format conversion. We did not attempt to further optimize or parallelize this step.

For the actual slicing of the dumps we used Hadoop's MapReduce (Dean & Ghemawat, 2004) facility. In the mapping phase, each triple is sorted into buckets based on the URIs mentioned in the subject and object position. In the reduce phase, the triples in each bucket are written to a separate file.

To gather some proof-of-concept results, we first processed the file `uniref.nt` (270M triples) on a single machine, then two machines and then four. We then ran a test on the complete UniProt dataset (1.4B triples) on four machines.

#Machines	1	2	4	4
#Triples	272M	272M	272M	1.456M
Conversion	1:17:01	1:19:30	1:19:10	6:19:28
MapReduce	10:04:35	5:18:55	2:56:54	16:36:26

These preliminary results show that processing of a very large RDF dump to obtain slices equivalent to the documents that are being served as description of the resources using the Linked Data paradigm is feasible. The processing task can be parallelized, and the results indicate that, unlike with the crawling approach, the data retrieval step is unlikely to be a bottleneck.

3.5. Adoption

Currently, the Sitemap Specification is currently available in its 5th release¹⁶. The specification creation process involved a great deal of interaction with data producers and developers alike. Most of this interaction happened in specialized Mailing Lists and through technical workshop dissemination (Tummarello, 2007).

The process has so far been very interactive, with data producers proposing features and clarifications they considered important such as, for example split datasets for DBpedia, and details on how to use a sitemap on shared domains like `purl.org`.

Thanks to this direct interaction, adoption at Data Producer level has so far been very satisfactory. Most large datasets provide a Semantic Sitemap and in general we report that data producers have been very keen to add one when requested given the very low overhead and the perceived lack of negative consequences.

At consumer level and as discussed earlier, the Sindice Semantic Web indexing engine adopts the protocol (Tummarello, Oren, & Delbru, Sindice.com: Weaving the open linked data., 2007) and thanks to it has indexed, as today, more than 26 million RDF documents.

We can report that the SWSE Semantic Web Search Engine (Harth, Hogan, Delbru, Umbrich, O'Riain, & Decker, 2007) will also soon be serving data obtained thanks to dumps downloaded using this extension.

¹⁶ <http://sw.deri.org/2007/07/sitemapextension/>

While it is unlikely that the current specifications will change profoundly, we envision that future versions of the Semantic Sitemaps will address issues such as: Data published in formats such as RDFa and using GRDDL transformations, datasets available in multiple versions, enhance SPARQL end point descriptions, mirrored datasets and copyright and legal related information.

3.6. Summary

In this chapter we have described an extension to the original Sitemap Protocol to deal with large Semantic Web datasets. We have highlighted several challenges to the way Semantic Web data have been previously published and subsequently showed how these can be addressed by applying the proposed specifications, thus allowing both clients and servers alike to provide and important novel functionalities.

We have further verified the feasibility of an important use case: server side “dump splitting” to efficiently process and index millions of documents from the Semantic Web. We have shown that this task can be performed efficiently and in a scalable fashion even with modest hardware infrastructures.

4. Large scale preprocessing of harvested data: Reasoning at Web Scale

It would be practically impossible, and certainly incorrect, to apply reasoning over a single RDF model composed of all RDF documents found on the Semantic Web. Letting alone the sheer computational complexity, the problem here is clearly the integration of information from different contexts: each document is published in a particular context, and a naive integration of documents coming from different contexts can result in undesirable inferred assertions.

We therefore strive to ensure that the context is maximally preserved when aggregating documents in a Semantic Web indexing engine such as the Sindice infrastructure that we use as the basis for ECSSE. We proceed in a way that we consider to be a form of *divide, compose and conquer* approach to Web scale reasoning (also referred to as contextual reasoning (Bouquet, Ghidini, Giunchiglia, & Blanzieri, 2003)). The ABox and TBox¹⁷ of the Web of Data is partitioned into smaller “context boxes” (on a per-document basis). Then, based on dependencies between these “boxes”, we are able to combine multiples boxes together in an aggregated context that preserves us from inferring undesirable assertions.

4.1. Contexts on the Semantic Web

For an RDF model published on the web the URI at which the RDF document is retrievable provides a natural way to define its context. By resolving the URI, an agent is able to retrieve an RDF graph. It is common practice to name this graph with that context URI, i.e. associate the context URI with each statement of the graph. Naming an RDF graph has multiple benefits. It helps tracking the provenance of each statement. In addition, since named graphs are treated as first class objects, it enables the description and manipulation of a set of statements just as for any other resource. A formal description of the Named Graph framework can be found in (J. Carroll, 2005). For our purposes, it is sufficient to understand that by context we identify a set of statements of a (finite set of) file(s) on the Web.

Guha (Guha, 1992) proposed a context mechanism for the Semantic Web which provides a formal basis to specify the aggregation of contexts. Within his framework, a context denotes the scope of validity of a statement. This scope is defined by the symbol *ist* (“is true in context”), introduced by Guha in (Guha, 1992). The notation *ist(c, φ)* states that a proposition φ is true in the context *c*.

The notion of context presented here is based on Guha's context mechanism. Its aim is to enable the control of integration of ontologies on the granularity level of documents, and ultimately avoid the aggregation of ontologies that may result in undesirable inferred assertions.

Aggregate Context An *Aggregate Context* is a subclass of *Context*. Its content is composed by the content retrieved by resolving its URI, and by the contents lifted from other contexts. An aggregate context must contain the full specification of what it imports. In our case, each RDF document is considered an *Aggregate Context*, since an RDF document always contains the specification of what it imports through explicit or implicit import declarations, described next.

¹⁷ as which we refer to the assertional and structural knowledge in RDF(S) and OWL documents, slightly abusing Description logics terminology.

Lifting Rules Since contexts are first class objects, it becomes possible to define expressive formulae whose domains and ranges are contexts. An example of this are so called *Lifting Rules* that enable to *lift* axioms from one context to another. In Guha's context mechanism, a lifting rule is a property type whose domain is an *Aggregate Context* and whose range is a *Context*. A lifting rule can simply import all of the statements from a context to another, but can also be defined to select precisely which statements have to be imported.

4.2. Import Closure of Semantic Documents

On the Semantic Web, ontologies are meant to be published so to be easily reused by third parties. OWL provides the `owl:imports` primitive to indicate the inclusion of a target ontology inside an RDF document. Conceptually, importing an ontology brings the content of that ontology into another document.

The `owl:imports` primitive is transitive. That is, an import declaration states that, when reasoning with an ontology O , one should consider not only the axioms of O , but the entire import closure of O . The imports closure of an ontology O is the smallest set containing the axioms of O and of all the axioms of the ontologies that O imports. For example, if ontology O_A imports O_B , and O_B imports O_C , then O_A imports both O_B and O_C .

Implicit Import Declaration The declaration of `owl:imports` primitives is a not a common practice in Web documents, most published documents do not contain explicit `owl:imports` declarations. For example, among the 30 million of documents in the Sindice index, only 5.56 thousand are declaring at least one `owl:imports`.

Instead, documents generally refer to classes and properties of existing ontologies by their URIs. For example, most FOAF profile documents don't explicitly import the FOAF ontology, but instead just refer to classes and properties of the FOAF vocabulary. Following the W3C best practices (Miles, Baker, & Swick, 2006) and Linked Data principles (Berners-Lee, 2006), the URIs of the classes and properties defined in an ontology should be resolvable and provide the machine-processable content of the vocabulary.

In the presence of such dereferenceable class or property URIs, we perform what we call an *implicit import*. By dereferencing the URI, we attempt to retrieve a document containing the ontological description of the entity and to include its content inside the source document.

Also implicit import is considered transitive. For example, if a document refers to an entity E_A from an ontology O_A , and if O_A refers to an entity E_B in an ontology O_B , then the document imports two ontologies O_A and O_B .

Import Lifting Rules Guha's context mechanism defines the *importsFrom* lifting rule. It is the simplest form of lifting and corresponds to the inclusion of one context into another. The `owl:imports` primitive or the implicit import declaration is easily mapped to such a lifting rule.

Definition 1 (Guha, McCool, & Fikes, 2004). Let c_2 be a context with a set of propositions $\varphi(x)$, and c_1 a context with an `owl:imports` declaration referencing c_2 . Then the set of propositions $\varphi(x)$ is also true in the context c_1 :

$$lst(c_2, \varphi(x)) \wedge lst(c_1, importsFrom(c_1, c_2)) \rightarrow lst(c_1, \varphi(x))$$

A particular case is when import relations are cyclic. Importing an ontology into itself is considered a null action, so if ontology O_A imports O_B and O_B imports O_A , then the two ontologies are considered to be equivalent (Bechhofer, et al., 2004). Based on this OWL definition, we extend Guha's definition to allow cycles in a graph of *importsFrom*. We introduce a new symbol *eq*, and the notation $eq(c_1, c_2)$ states that c_1 is equivalent to c_2 , i.e. that the set of propositions true in c_1 is identical to the set of propositions true in c_2 .

Definition 2. Let c_1 and c_2 be two contexts. If c_1 contains the proposition *importsFrom*(c_1, c_2) and c_2 the proposition *importsFrom*(c_2, c_1), then the two contexts are considered equivalent:

$$ist(c_2, importsFrom(c_2, c_1)) \wedge ist(c_1, importsFrom(c_1, c_2)) \rightarrow eq(c_1, c_2)$$

4.3. Deductive Closure of Semantic Documents

In Sindice, we consider for indexing inferred statements from the deductive closure of each document we index. What we call here the “deductive closure” of a document is the set of assertions entailed in the aggregate context composed of the document itself and of its ontology import closure.

When reasoning with Web documents, we can not expect to deal with documents at a level of expressiveness of OWL-DL (d'Aquin, Baldassarre, Gridinoc, Angeletou, Sabou, & Motta, 2007), but would need to consider OWL Full. Since under such circumstances, we cannot strive for complete reasoning anyway, we therefore content with an incomplete but *finite* entailment regime based on a subset of RDFS and OWL, namely the *ter Horst* fragment (ter Horst, 2005) as implemented by off-the-shelf rule-based materialisation engines such as for instance OWLIM¹⁸. Such an incomplete deductive closure is sufficient with respect to increasing the precision and recall of the search engine, providing useful RDF(S) and OWL inferences such as class hierarchy, equalities, property characteristics such as inverse functional properties or annotation properties.

Indeed, a rule-based inference engine is currently used to compute full materialisation of the entailed statements with respect to this finite entailment regime. In fact is it in such a setting a requirement that deduction to be finite, which in terms of OWL Full can only be possible if the entailment regime is incomplete (as widely known the RDF container vocabulary alone is infinite already (Hayes, 2004)). This is deliberate and we consider a higher level of completeness hardly achievable on the Web of Data: In a setting where the target ontology to be imported may not be accessible at the time of the processing, for instance, we just ignore the imports primitives and are thus anyway incomplete from the start.

Deductive Closure of Aggregate Contexts One can see that the deductive closure of an aggregate context can lead to inferred statements that are not true in any of the source contexts alone.

Definition 3. Let c_1 and c_2 be two contexts with respectively two propositions φ_1 and φ_2 , $ist(c_1, \varphi_1)$ and $ist(c_2, \varphi_2)$, and $\varphi_1 \wedge \varphi_2 \models \varphi_3$, such that $\varphi_2 \not\models \varphi_3$, $\varphi_1 \not\models \varphi_3$, then we call

¹⁸

<http://www.ontotext.com/owlim/>

φ_3 a newly entailed proposition in the aggregate context $c_1 \wedge c_2$. We denote the set of all newly defined propositions $\Delta_{c_1 c_2}$, i.e.,

$$\Delta_{c_1 c_2} = \{ist(c_1, \varphi_1) \wedge ist(c_2, \varphi_2) \models ist(c_1 \wedge c_2, \varphi_3) \text{ and } \neg(ist(c_1, \varphi_3) \vee ist(c_2, \varphi_3))\}$$

For example, if a context c_1 contains an instance x of the class *Person*, and a context c_2 contains a proposition stating that *Person* is a subclass of *Human*, then the entailed conclusion that x is a human is only true in the aggregate context $c_1 \wedge c_2$:

$$ist(c_1, Person(x)) \wedge ist(c_2, subclass(Person, Human)) \rightarrow ist(c_1 \wedge c_2, Human(x))$$

Note that - by considering (in our case (Horn) rule-based) RDFS/OWL inferences only - aggregate contexts enjoy the following monotonicity property¹⁹: if aggregate context $c_1 \subseteq c_2$ then $ist(c_2, \emptyset)$ implies $ist(c_1, \emptyset)$, or respectively, for overlapping contexts, if $ist(c_1 \cap c_2, \emptyset)$ implies both $ist(c_1, \emptyset)$ and $ist(c_2, \emptyset)$. We will exploit this property in the following outlined reasoning procedure.

4.4. Context Dependent Reasoning Procedure

Our goal is to develop a distributed ABox reasoning procedure based on an persistent TBox, called an ontology base. The ontology base is in charge of storing any ontology discovered on the Web of Data with the import relations between them. The ontology base also stores inference results that has been performed in order to reuse such computation later.

When trying to apply the above theory, architectural design problems appear. In order to support the context mechanism presented in the previous section, the ontology base requires 1. an internal representation of the import relations and 2. lifting rules as built-in triggers that are activated in the presence of import declarations.

We first introduce the basic concepts used for the formalisation of the ontology base. We then discuss an optimised strategy to update the ontology base. We finally describe how to query the ontology base for performing ABox reasoning on a single document.

4.5. Ontology Base Concepts

The ontology base uses the notion of named graphs for modelling the contexts of ontologies and their import relations. The ontology base has a rules-based inference engine that can be applied to any context independently or to a combination of them. This particular reasoning model allows localising the inference of a reasoning task.

Ontology Entity An *Ontology Entity* is a property, instance of `rdf:Property`, or a class, instance of `rdfs:Class`. The ontology entity must be identified by an URI (we exclude entities that are identified by a blank node) and the URI must be resolvable and point to a document containing the entity definition.

¹⁹ We remark here that under the addition of possibly non-monotonic rules to the Semantic Web architecture, this context monotonicity only holds under certain circumstances [17]

Ontology Context Internally to our ontology base, an *Ontology Context* O is a named graph composed by the ontology statements that have been retrieved after dereferencing the entity URIs of the ontology. The content of this graph consists of the union of the description of each property and class associated to a Web namespace of an ontology. According to the Best practices (Miles, Baker, & Swick, 2006), properties and classes defined in an ontology context should have the same URI namespace.

Usually, this means that the RDF associated with the ontology context will simply contain the ontology as found on the Web. There are cases however, e.g. in the case of the DBpedia ontology, where each property and class has its own RDF document. The DBpedia ontology context is therefore composed by the union of all these RDF documents.

Ontology Network An ontology context can have import relationships with other ontology contexts. A directed link l_{AB} between two contexts, O_A and O_B , stands for O_A imports O_B . In this view, l_{AB} serves as a pointer to a frame of knowledge that is necessary for completing the semantics of O_A and is mapped to an *importsFrom* lifting rule.

Definition 4 (Ontology Network). An ontology network is a directed graph $G = (O, L)$ with O a set of vertices (ontology contexts) and L a set of directed edges (import relations).

Definition 5 (Import Closure). The import closure of an ontology context O_A is a subgraph $G_{O_A} = (O', L')$ such as:

$$G_{O_A} \subseteq G, O' \subseteq O, L' \subseteq L \mid \forall O \in O', \exists \text{path}(O_A, O)$$

where a path is a sequence of n vertices O_A, O_B, \dots, O_n such that from each of its vertices there is an edge to the next vertex.

For the purpose of this work, we consider the import closure to also contain the result of the inference that can be performed on the union of all the ontologies. For example, given two ontology contexts O_A and O_B , with O_A importing O_B , the import closure of O_A will consist of the context aggregating O_A, O_B and the deductive closure of the union of the two ontologies.

4.6. Ontology Base Update Strategy

Whenever a new ontology context is added to the ontology network, the import closure of the new context is materialised. The deductive closure is then performed on the ontology by lifting the axioms of the imported ontologies. The ontology network is then updated so that the stored inferred triples are never duplicated.

This is better explained by an example. In Fig. 1 a document D_1 is processed which imports explicitly 2 ontologies O_1 and O_2 . The import closure of D_1 is calculated and this is found to include also O_3 since O_3 is imported by O_2 . At this point, the deductive closure of O_1, O_2 and O_3 is performed both separately (stored in their respective nodes) and together by lifting their axioms into a virtual aggregate context. The statements resulting from the reasoning over O_1, O_2 and O_3 together but that are not found already in any of the source contexts (i.e. Δ_{O_1, O_2, O_3} , see Def. 3 for a detailed formalisation) are stored in the virtual aggregate context I_{123} .

At this point a new RDF document comes (D_2) which only imports O_1 and O_3 . The update strategy will: 1. calculate the inference results of O_1 and O_3 and store the new triples (Δ_{O_1, O_3}) in a new

virtual context I_{13} ; 2. subtract these triples from the content of the previous context I_{123} . I_{123} is renamed into I_{123-13} and at the same time connected to I_{13} by an import relation.

As a result of the upload procedure, if two or more ontologies are never activated together by an RDF document, their deductive closure will never be computed and stored.

A last optimisation is based on Def. 2. Whenever a cycle is detected into the ontology network, the ontology contexts present in the cycle are aggregated into one unique context.

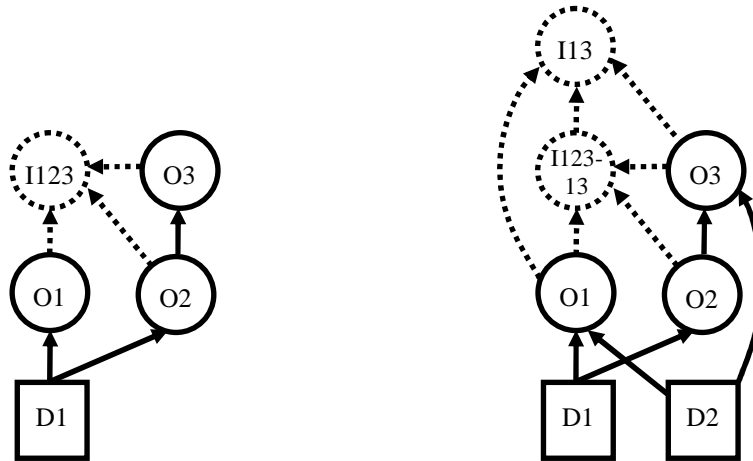


Fig. 1 On the left, the ontology network after processing a document D_1 . On the right, after processing D_2 the virtual context I_{123} has been split in 2 parts.

4.7. Ontology Base Querying Strategy

When a document imports an ontology O , we query the ontology base to lift the import closure $\mathcal{G}_O = (O', L')$ (see Def. 5) into the document.

If a document imports more than one ontology, we query the ontology base with the set of ontologies. The ontology base computes the import closure of each ontology and returns the union of the import closures including related Δ contexts.

For example in Fig. 1, the document D_1 imports the two ontologies O_1 and O_2 . This lifts the contexts $\{O_1, O_2, O_3, I_{13}, I_{123-13}\}$ into the document D_1 .

4.8. Prototype Implementation

In Sindice, reasoning is performed for every document at indexing time. This is possible with the help of a distributed architecture based on Hadoop²⁰. Each Hadoop node has its own ontology base which supports the linked ontology and inference model described in the previous section. A single ontology base is shared across several Hadoop worker jobs. This architecture can be seen as a distributed ABox reasoning with a shared persistent TBox.

²⁰

Hadoop: <http://hadoop.apache.org/core/>

Each Hadoop worker job acts as a reasoning agent processing a document at a time. It first analyses the document in order to discover the explicit `owl:imports` declarations or the implicit ones by resolving each class and property URIs. Then, the reasoning agent lifts the content of the referred ontologies inside the documents by querying the ontology base with the URIs of the ontologies. The ontology base responds by providing the set of ontological assertions as described in 3.3. As none of the triple-stores available today support context aware reasoning, our implementation required considerable low level modifications to the Aduna Sesame framework²¹.

4.8.1. Experimental Setup

The prototype has been deployed on a small Hadoop cluster of 3 nodes. Each node is a 4 cores 2.33GHz with 8GB and executes 4 Hadoop jobs. Thus, in total, the cluster enables the execution of 12 Hadoop jobs in parallel.

Each node has its own ontology base, shared among 4 Hadoop jobs. We assume in the next that all the ontologies required for the experiment has been inserted beforehand into the ontology base.

4.8.2. Preliminary Results

Using the experimental setup described in the previous section, we are able to process²² an average of 40 documents per second when building the 30 million documents index currently available in the Sindice beta1 index.

The processing speed varies depending on the type of document. On documents with a simple and concise representation, such as the ones found in the Geonames dataset, the prototype is able to process up to 80 documents per second. It is to be noticed however that these results come from a prototypical implementation, still to be subject to many possible technical optimizations.

The deductive closure of each document that has been indexed can be checked online on the Sindice web site²³. For example, for the search result "tim berners lee"²⁴, the hyperlink "Cached"²⁵ will display the cached version of the document and its deductive closure. The same page also displays, when clicking on the hyperlink "Ontologies", the list of ontologies that have been recursively imported when performing the reasoning.

It is interesting to make a few observation based on the large corpus of documents processed. We observe that on a snapshot of the index containing 6 million of documents, the original size of the corpus was 18GB whereas the total size after inference was 46GB, thus a ratio of 1.5. As a result of inference, we can then observe that we are indexing twice as many statements as we would by just indexing explicit semantics. Also we find that the number of ontologies, defined as documents which define new classes or properties, in our knowledge base is around 95.000, most of which are fragments coming from projects such as OpenCyc, Yago and DBpedia. The ratio of ontologies to semantic documents is nevertheless low, currently 1 to 335.

²¹ OpenRDF Sesame: <http://www.openrdf.org/>

²² including document pre-processing and post-processing (metadata extraction, indexing, etc.)

²³ Sindice.com: <http://www.sindice.com>

²⁴ <http://sindice.com/search?q=tim%20berners%20lee&qt=term>

²⁵ <http://sindice.com/search/page?url=http%3A%2F%2Fwww.w3.org%2Fpeople%2FBerners-Lee%2Fcard>

4.9. Related work

The notion of context has been extensively studied since early 1990s, starting with a first formalisation by Guha (Guha, Contexts: a formalization and some applications, PhD Thesis, 1992) followed by McCarthy (McCarthy, 1993) and by Giunchiglia (Giunchiglia, 1993). For more information about the domain, (Serafini & Bouquet, 2004) presents a survey and comparison of the main formalizations of context. There is recent works that adapt context formalisations to the Semantic Web such as (Bouquet, Giunchiglia, van Harmelen, Sera, & Stuckenschmidt, 2004), (Guha, McCool, & Fikes, 2004).

The previous references provide a strong theoretical background and framework for developing a context-dependent reasoning mechanism for the Semantic Web. Up to our knowledge, only (Stoermer, Bouquet, Palmisano, & Redavid, 2007) propose a concrete application of management of contextualized RDF knowledge bases. But no one has described a methodology for efficient large scale reasoning that deals with contextuality of web documents.

Assumption-based Truth Maintenance Systems (de Kleer, 1986) are somehow comparable to our approach since such system keeps track of dependency between facts in multiple views (contexts) in order to maintain consistency in a knowledge base. The difference with our approach lies in that we do not try to maintain consistency, instead we just maintain the provenance of each inferred statements.

A side remark in (Ding, Tao, & McGuinness, 2008) suggests that the authors follow a very similar strategy to ours in determining the ontological closure of a document, but no details on efficient contextual confinement and reuse of inferences - which are the main contribution of our work - are discussed.

4.10. Summary

We discussed a context dependent methodology for large scale web reasoning.

We first defined the problem conceptually and then illustrated how to create an ontology repository which can provide the materialization of reasoning statements while still keeping tracks of the original statements provenance. Our implementation currently supports a subset of RDFS and OWL. We find this level of inference support to be in line with our target objective to support the RDF community of practice, e.g. the Linked Data Community, which usually relies only on RDFS and OWL features covered by the OWL ter Horst fragment (ter Horst, 2005).

Reasoning of the Web of Data enables us to be more effective in term of precision and recall for Semantic Web information retrieval. As an example, it would not be possible to answer a query on FOAF social networking files asking for entities which have label "giovanni" unless reasoning is performed to infer `rdfs:labels` from the property `foaf:name`.

The context mechanism allows us to avoid the deduction of undesirable assertions in documents, a common risk when working with the Web of Data. However, this context mechanism does not restrict the freedom of expression of data publisher. Data publisher are still allowed to reuse and extend ontologies in any manner, but the consequences of their modifications will be confined in their own context, and will not alter the intended semantics of the other documents.

The technique presented in this chapter is mainly focussed on the TBox level. Import relations between ontologies provide a good support for lifting rules. On ABox level, this becomes more difficult since it is not clear which ABox relations should be consider as lifting rules. ABox relations, such as `owl:sameAs` declarations will be investigate in a future work.

5. Harvested Data Statistics

This section presents a statistical analysis of the data that is contained in the Sindice index. The main goal behind the computation of these statistics was to put a framework for analysis of the Sindice index in place. This has been achieved, and further work can focus on computing specific statistics to answer scientific and engineering questions.

Statistics are computed by running distributed Hadoop jobs over the *page repository*, which is an HBase table that provides random access to the original RDF data that makes up the index.

For each data item, Sindice stores not just the explicit RDF triples obtained from the source document, but also inferred triples. We therefore distinguish between *explicit content* and *implicit* or *inferred content* in all sections below.

The statistics shown in this chapter represent the state of the Sindice index as of September 2008. The index has since grown from 34M to 46M documents. We plan to re-calculate and extend the statistics in early 2009.

5.1. Summary Statistics

This section provides some high-level overall statistics about the data contained in the Sindice index.

Total number of records: **34,584,007**

Explicit content

Records without explicit-content: 140,072

Records where explicit-content cannot be parsed as N-Triples: 37,941

Successfully parsed records: 34,405,994

Total explicit triples in the page repository: 428,400,582

Average document size: 12.45 triples

Implicit content

Number of records: 34,584,007

Records without implicit-content: 3,110,228

Records where implicit-content cannot be parsed as N-Triples: 800

Successfully parsed records: 31,472,979

Total inferred triples in the page repository: 380,219,154

Average document size: 12.08 triples

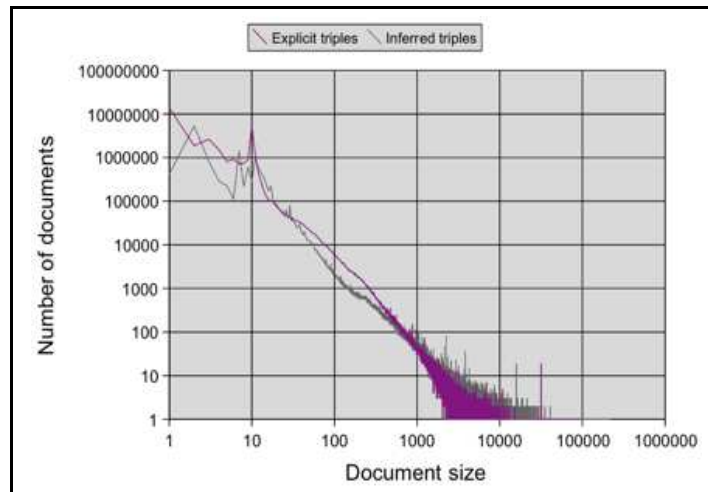
16.95M out of the 31.47M successfully parsed records contain zero inferred triples, that is, no inferences.

The average document size is calculated as: total triples divided by total documents.

Records without explicit-content, and records where either content cannot be parsed, indicate corrupt data in the index. This was usually caused by software glitches whose results have not yet been flushed from the index.

5.2. Distribution of Document Sizes

The following chart shows the distribution of document sizes in the index. This is a log-log axis plot. The purple line represents explicit content, the grey line represents implicit content.



Some noteworthy features:

- Power law distribution
- Spikes for small document sizes are caused by large number of recurring patterns, e.g. some Microformats converted to RDF always result in the same number of triples
- We have no good explanation for the noticeable low number of inferred content in the 30-300 triple range

5.3. Top Domains

The table below shows the results of grouping all documents in the index by their DNS domain name, and showing the domains that have the most documents. In general, the second-level domain (such as `wordpress.com`) is used for the grouping. For certain top-level domains, such as `.uk`, the third-level domain is used because the second-level domains are from the fixed set `.co.uk`, `.ac.uk` and so on.

The table shows that the Sindice index is dominated by large datasets that are imported from RDF dumps via Semantic Sitemaps. Crawled documents constitute a smaller fraction of the index.

Documents	Domain	Comment
15304201	<code>dbpedia.org</code>	Wikipedia conversion into RDF
13273503	<code>geonames.org</code>	Places described in RDF
4874319	<code>rkbexplorer.com</code>	Scientists, publications, funding etc in RDF
570074	<code>joanneum.at</code>	RDF conversion of Eurostat
181931	<code>uniprot.org</code>	RDF descriptions of proteins
98528	<code>livejournal.com</code>	User profiles in FOAF
12270	<code>wordpress.com</code>	Blog authors and commenters in hCard
11394	<code>bio2rdf.org</code>	Genes, proteins etc described in RDF
5423	<code>soton.ac.uk</code>	University department described in RDF
5219	<code>digg.com</code>	Mostly microformat-enabled user profiles
4778	<code>semanticweb.org</code>	Semantic Web community information
4645	<code>apassant.net</code>	FOAF conversion of Flickr user profiles
4160	<code>revyu.com</code>	Reviews in RDF
4126	<code>l3s.de</code>	RDF conversion of DBLP
3652	<code>dowhatimean.net</code>	SIOC-enabled weblog

3420	technorati.com	Creative Commons licenses
2870	fu-berlin.de	Various RDF datasets
2725	blogsme.com	FOAF and XFN for user profiles
2608	quotationsbook.com	Creative Commons licenses
2587	blogspot.com	Tag microformat and Creative Commons licenses
2542	typepad.com	XFN and Creative Commons licenses
2470	openlinksw.com	Various RDF datasets
2089	kwark.org	FOAF of user profiles
2073	johnbreslin.com	SIOC-enabled weblog
1977	vox.com	hCards of user profiles
1945	bibsonomy.org	Social bookmarks in RDF
1846	w3.org	Various
1672	semanlink.net	Social bookmarks in RDF
1053	gnolia.com	hCards for user profiles
1005	zitgist.com	YAGO in RDF
1000	netbib.de	Creative Commons licenses

5.4. Objects of RDF Triples

The following set of statistics considers all documents in the Sindice index as a set of RDF triples, and performs certain counts over the third (object) component of the triples. This was done to get some insight into real-world usage of RDF features like datatypes and language tags, and to fine-tune the performance of the index with regard to its storage of URIs vs. literals.

Only the explicit content was considered.

Object node type	Triples
URI	336,756,002
Literal	86,312,720
Blank node	5,331,860

Datatypes: Out of 86M literals, only 128k are datatyped, the rest are plain.

Language tags: Out of the plain literals, 15M have a language tag, 71M have no language tag. Almost all language tags are @en (English), only about 50k are for other languages.

The following table breaks the 128k typed literals down by datatype. Only those datatypes occurring in more than 100 triples are shown.

Datatype	Triples
http://www.w3.org/2001/XMLSchema#string	63214
http://www.w3.org/2001/XMLSchema#float	32622
http://www.w3.org/2001/XMLSchema#dateTime	8023
http://www.w3.org/2001/XMLSchema#anyURI	3788
http://www.w3.org/2001/XMLSchema#date	3639
http://www.w3.org/2001/XMLSchema#int	3360
http://www.w3.org/2001/XMLSchema#integer	2770
http://www.w3.org/2001/XMLSchema#nonNegativeInteger	2379
http://www.w3.org/2001/XMLSchema#double	2110
http://www.w3.org/2001/XMLSchema#gYear	1593
xs:string	869
http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral	863

http://www.w3.org/2001/XMLSchema#boolean	779
http://www.w3.org/2000/10/XMLSchema#string	752
http://www.w3.org/2001/XMLSchema#decimal	318
http://www.w3.org/2001/XMLSchema#unsignedLong	200
http://www.w3.org/2001/XMLSchema#Integer	172
http://www.openmobilealliance.org/tech/profiles/UAPROF/xmlschema-20030226#Literal	163
http://www.w3.org/2002/12/cal/icaltzd#dateTime	126

Some noteworthy features of this table:

- The top item is technically unnecessary, as a plain literal is semantically equivalent to `xsd:string`
- The frequent use of the fourth item, `xsd:anyURI`, is somewhat questionable, as RDF provides a native non-literal way of referring to URIs
- Two items are authoring errors (“`xs:string`” as a URI, and the incorrect `/2002/10/` namespace)
- The only two non-XSD items appear to be attempts at working around perceived shortcomings of the RDF/XSD type system

6. Acknowledgements

Document authors are: Giovanni Tummarello, Renaud Delbru, Richard Cyganiak, Michele Catasta, Oana Ureche, Benjamin Heitmann

Many people have provided valuable feedback and comments about Semantic Sitemaps including Chris Bizer (Free University Berlin), Andreas Harth (DERI Galway), Aidan Hogan (DERI Galway), Leandro Lopez (independent), Stefano Mazzocchi (SIMILE - MIT), Christian Morbidoni (SEMEDIA - Universita' Politecnica delle Marche), Michele Nucci (SEMEDIA - Universita' Politecnica delle Marche), Eyal Oren (DERI Galway), Leo Sauermann (DFKI)

7. Appendix 1: The Okkam Ontology in RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:label="http://purl.org/net/vocab/2004/03/label#"
  xmlns:status="http://www.w3.org/2003/06/sw-vocab-status/ns#"
  xmlns:okkam="http://okkam.org/terms#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
>

  <owl:Class rdf:about="#OkkamEntity">
    <rdfs:label xml:lang="en">Okkam Entity</rdfs:label>
    <label:plural xml:lang="en">Okkam Entities</label:plural>
    <rdfs:comment xml:lang="en">Any entity that can be identified using the
      OKKAM entity naming system. In general one could say this is equivalent
      to resource but there are certain entity types that will work better on
      the OKKAM system</rdfs:comment>
    <status:term_status>unstable</status:term_status>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#okkamID"/>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:minCardinality>
        <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
    </owl:Class>

    <owl:InverseFunctionalProperty rdf:about="#okkamID">
      <rdfs:label xml:lang="en">Okkam ID</rdfs:label>
      <label:plural xml:lang="en">Okkam IDs</label:plural>
      <rdfs:comment xml:lang="en">The Okkam ID of the entity, given as a
        literal</rdfs:comment>
      <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/terms/identifier"/>
      <status:term_status>unstable</status:term_status>
      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
      <rdfs:domain rdf:resource="#OkkamEntity"/>
    </owl:InverseFunctionalProperty >
  </rdf:RDF>
```

8. References

- Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In K. Aberer, K. S. Choi, N. Noy, D. Allemang, K. I. Lee, L. Nixon, et al., *The Semantic Web. Volume 4825 of LNCS* (pp. 722-735). Springer.
- Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., L. M. D., Patel-Schneider, P. F., et al. (2004). *OWL Web Ontology Language Reference. W3C Recommendation*.
- Berners-Lee, T. (2006). *Linked data*. W3C Design Issues.
- Berners-Lee, T. (2006). Tabulator: Exploring and Analyzing linked data on the Semantic. In: *Proceedings of the The 3rd International Semantic Web User Interaction Workshop*.
- Bouquet, P., Ghidini, C., Giunchiglia, F., & Blanzieri, E. (2003). Theories and uses of context in knowledge representation and reasoning. *Journal of Pragmatics* , 455-484.
- Bouquet, P., Giunchiglia, F., van Harmelen, F., Sera, L., & Stuckenschmidt, H. (2004). Contextualizing ontologies. *Journal of Web Semantics I(4)* , 325-343.
- Carroll, J., Bizer, C., Hayes, P., & Stickler, P. (2005). Named Graphs. *Journal of Web Semantics 3(4)* , 247-267.
- Cyganiak, R., Stenzhorn H, Delbru R, Decker, S, Tummarello G, Semantic Sitemaps: Efficient and Flexible Access to Datasets on the Semantic Web, in Proceedings of the Proceedings of the 5th European Semantic Web Conference 2008.
- d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., & Motta, E. (2007). Characterizing Knowledge on the Semantic Web with Watson. *EON*, (pp. 1-10).
- de Kleer, J. (1986). An Assumption-Based TMS. In *Artif. Intell. 28(2)* (pp. 127-162).
- Dean, J., & Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. *OSDI'04: Proceedings of the 6th conference on Symposium on Operating Systems Design and Implementation*, (pp. 137-150).
- Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., et al. (2004). Swoogle: a search and metadata engine for the semantic web. *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and and knowledge management*, (pp. 652-659). New York.
- Ding, L., Tao, J., & McGuinness, D. L. (2008). An initial investigation on evaluating semantic web instance data. *WWW '08: Proceeding of the 17th international conference on World Wide Web*, (pp. 1179-1180). New York.
- Giunchiglia, F. (1993). Contextual reasoning. In *Epistemologia, special issue on I Linguaggi e le Macchine* (pp. 345-364).
- Guha, R. V. (1992). *Contexts: a formalization and some applications, PhD Thesis*. Stanford.
- Guha, R. V., McCool, R., & Fikes, R. (2004). Contexts for the Semantic Web. *International Semantic Web Conference*, (pp. 32-46).
- Guha, R. V., McCool, R., & Fikes, R. (2004). Contexts for the Semantic Web. *International Semantic Web Conference*, (pp. 32-46).
- Harth, A., Hogan, A., Delbru, R., Umbrich, J., O'Riain, S., & Decker, S. (2007). Swse: Answers before links! *SemanticWeb Challenge 2007, 6th International Semantic Web Conference*.

- Hayes, P. (2004). *RDF Semantics. W3C Recommendation, World Wide Web Consortium.*
- J. Carroll, C. B. (2005). Named graphs, provenance and trust. *WWW '05: Proceedings of the 14th international conference on World Wide Web*, (pp. 613-622). New York.
- Jacobs, I., & Walsh, N. (2004). *Architecture of the World Wide Web, Volume One - W3C Recommendation*. Retrieved September 25, 2006, from <http://www.w3.org/TR/webarch/>
- McCarthy, J. (1993). Notes On Formalizing Context. *Proceedings of IJCAI-93*, (pp. 555-560).
- Miles, A., Baker, T., & Swick, R. (2006). *Best Practice Recipes for Publishing RDF Vocabularies. Technical Report.*
- Serafini, L., & Bouquet, P. (2004). Comparing formal theories of context in AI. In *Artificial Intelligence* (pp. 155(1-2):41).
- Stickler, P. (2005). *CBD - Concise Bounded Description*. Retrieved September 25, 2006, from <http://www.w3.org/Submission/CBD/>.
- Stoermer, H., Bouquet, P., Palmisano, I., & Redavid, D. (2007). A Context-Based Architecture for RDF Knowledge Bases: Approach, Implementation and Preliminary Results. *RR*, (pp. 209-218).
- ter Horst, H. J. (2005). Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2-3) , 79-115.
- Tummarello, G. (2007). A sitemap extension to enable efficient interaction with large quantity of linked data. *W3C Workshop on RDF Access to Relational Databases*.
- Tummarello, G., Morbidoni, C., Puliti, P., & Piazza, F. (2005). Signing individual fragments of an rdf graph. *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, (pp. 1020–1021). New York.
- Tummarello, G., Oren, E., & Delbru, R. (2007). Sindice.com: Weaving the open linked data. In K. Aberer, K. S. Choi, N. Noy, D. Allemang, K. I. Lee, L. J. Nixon, et al., *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007)* (pp. 547-560). Berlin, Heidelberg: Springer Verlag.