Managing Co-reference on the Semantic Web

Hugh Glaser, Afraz Jaffri, Ian C. Millard
School of Electronics and Computer Science
University of Southampton
Southampton, Hampshire, UK
{hg, aoj04r, icm}@ecs.soton.ac.uk

ABSTRACT

Co-reference resolution, or the determination of 'equivalent' URIs referring to the same concept or entity, is a significant hurdle to overcome in the realisation of large scale Semantic Web applications. However, it has only recently gained the attention of research communities in the Semantic Web context, and while activities are now underway in identifying co-referent or conflated URIs, little consideration has been given to tools and techniques for storing, manipulating, and reusing co-reference information.

This paper provides an overview of the specification, implementation, interactions and experiences in using the *Co-reference Resolution Service* (CRS) to facilitate rigorous management of URI co-reference data, and enable interoperation between multiple Linked Open Data sources. Comparisons are made throughout the paper contrasting the differences in the way the CRS manages multiple URIs for the same resource with the emerging practice of using `owl:sameAs` to identify duplicate URIs. The advantages and benefits that have been gained from deploying the CRS on a site with multiple Linked Data repositories are also highlighted.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: Information storage and retrieval—*Online Information Services – data sharing, web based services*

Keywords

Co-reference, Linked Data, Semantic Web

1. INTRODUCTION

The Semantic Web vision fundamentally requires the creation of a 'Web of Data' containing large quantities of readily accessible, interlinked, and machine readable information, analogous to the existing World Wide Web content currently available for human consumption.

In recent years an increasing number of semantic datasets are emerging, fuelled primarily by the efforts of the Linking Open Data community, forming the beginnings of such a resource. By following the guidelines set out in the Linked Data Tutorial [3], or through the use of tools such as D2R [5], existing datasets are being made available online, including notable encyclopaedic, geographical, music, film and academic publication related resources.

However, in many cases there is minimal interlinking between these datasets, as often they are existing resources which have recently been exposed in a semantic representation. As a result, the 'web' remains fragmented and difficult to navigate.

As more datasets appear there is significant potential for overlap to occur, with a given resource being described in two or more repositories. These representations are more than likely to use different identifiers, stemming from each source, unless one or other of the datasets is relatively new and has been constructed with knowledge of the other. Furthermore, the information in different repositories could be expressed against different ontologies.

The problems involved with identifying these 'duplicate' descriptions, either within a single dataset or across multiple data sources, are encapsulated in the area of co-reference resolution [9]. Research has been and continues to be carried out in this field, developing systematic analysis and heuristic based approaches to identifying co-references in or between datasets, however techniques for managing, publishing and using co-reference information are lacking.

This is not a problem that will disappear as the Semantic Web gains momentum, and it is naïve to suppose or rely on the fact that a 'standard' set of identifiers will eventually emerge over time; organisations, companies and government agencies are unlikely to be willing to adopt identifiers over which they have no authority or control.

The remainder of this paper describes the Co-reference Resolution Service (CRS, formerly known as the Consistent Reference Service), which we have developed to meet the needs of managing co-reference data, both within our own project and the Semantic Web as a whole.

2. THE NEED FOR A CO-REFERENCE RESOLUTION SERVICE

Part of the ReSIST project with which the authors are involved aims to provide a synthesised view of resources related to Resilient and Dependable Computing research, utilising a Semantic Web based approach [6]. Data has been acquired from multiple sources, detailing academic publications, researchers, institutions and projects, and converted into RDF as required. This totals approximately 100M triples, from 20+ different sources, ranging from large publication repositories to small chunks of information submitted by project partners. Data from each different source has been kept separate, and published as Linked Data on

LDOW2009, April 20, 2009, Madrid, Spain.

subdomains of `rkbexplorer.com`.

Clearly there is likely to be overlap and duplicity of information between these repositories, particularly with people and publications. We have deployed a number of algorithms to identify co-referent identifiers in and between our datasets, however this is of little use without some way of applying these results to a semantic application or further analysis tools.

The most prevalent way of dealing with 'duplicate' URIs that are deemed to be the same is to use the `owl:sameAs` predicate to link between them. The semantics of `owl:sameAs` dictate that all the URIs linked with this predicate have the same identity [2], implying that the subject and object must be the same resource. The major disadvantage with this approach is that the two URIs become indistinguishable, even though they may refer to different entities according to the context in which they are used.

Named graphs may be used in some cases to overcome this problem, but this approach has significant drawbacks. In addition to being outside of the RDF model, prior understanding of the graphs and their partition is required. Furthermore, if RDF descriptions are combined, cached, or passed between different services, then named graphs can easily be lost.

Generally, co-reference resolution techniques are not as certain as one might hope, somewhat undermining the strong semantics behind `owl:sameAs`. Once again, we must consider the notion of equivalence within a given context: with the exception of very elementary examples, one can only be sure that two URIs are equivalent within the confines of a specific application, whereas `owl:sameAs` asserts that two references are always the same.

It is the authors' belief that more often than not the use of `owl:sameAs` is inappropriate and is being applied incorrectly, and rather that `owl:sameAs` should only be used when the two concepts being represented are utterly indistinguishable.

The approach taken within the ReSIST project has been to separate out knowledge regarding co-reference and equivalence from the main datasets, in a manner similar to that in which early hypertext systems were developed by storing content and link-bases as distinct components. By treating such information as a first class entity and storing it in a separate system, the Co-reference Resolution Service, a number of benefits can be realised.

Firstly, a number of CRSes can be used to represent different co-reference contexts; applications can then use one or more CRSes as appropriate. For example, in undertaking citation analysis, a paper with the same title and text that appeared both as a journal article and technical report should be considered as two separate papers, whereas in many other applications it may be thought of as the same resource appearing in two different publication formats. A different CRS instance could be created to represent each viewpoint, whereas an application accessing a linked data site with embedded `owl:sameAs` links has no opportunity to choose an equivalence context.

Secondly, in recognising co-reference data as important knowledge in its own right, and by storing it separately and manipulating it through custom services, more powerful management techniques can be applied, including history, rollback and annotation capabilities.

In relation to the ongoing issues over URI identity, both the RDF predicate based approach [8] and the URI declarations approach [4] to identity management can be seamlessly integrated into the CRS framework. Indeed, any approach to URI identity management will be easier to implement and control in a world where knowledge of URI synonyms and URI definition are kept separate.

Merge uri_a and uri_b ...

$$bundle_1 = \{\underline{uri_a}\} \quad (1)$$
$$bundle_2 = \{\underline{uri_b}\} \quad (2)$$
$$bundle_3 = bundle_1 \cup bundle_2 \quad (3)$$
$$bundle_3 = \{\underline{uri_a}, uri_b\} \quad (4)$$
$$\cancel{bundle_1}, \cancel{bundle_2} \quad (5)$$

Merge uri_a and uri_c ...

$$bundle_4 = \{\underline{uri_c}\} \quad (6)$$
$$bundle_5 = bundle_3 \cup bundle_4 \quad (7)$$
$$bundle_5 = \{\underline{uri_a}, uri_b, uri_c\} \quad (8)$$
$$\cancel{bundle_3}, \cancel{bundle_4} \quad (9)$$

Merge uri_m and uri_n ...

$$bundle_6 = \{\underline{uri_m}\} \quad (10)$$
$$bundle_7 = \{\underline{uri_n}\} \quad (11)$$
$$bundle_8 = bundle_6 \cup bundle_7 \quad (12)$$
$$bundle_8 = \{\underline{uri_m}, uri_n\} \quad (13)$$
$$\cancel{bundle_6}, \cancel{bundle_7} \quad (14)$$

Merge uri_n and uri_b ...

$$bundle_9 = bundle_8 \cup bundle_5 \quad (15)$$
$$bundle_9 = \{uri_a, uri_b, uri_c, \underline{uri_m}, uri_n\} \quad (16)$$
$$\cancel{bundle_8}, \cancel{bundle_5} \quad (17)$$

Figure 1: Examples of bundle formation

Consider the case where the URI synonyms for the same resource are included as `owl:sameAs` links in the definition of the resource that is being described. Alterations to these URIs will cause an alteration in the definition of the URI. Separating the URI co-reference links into a bundle in a separate knowledge base allows the duplicate URIs to be changed without affecting the definition of the resource for the original URI.

3. CRS IMPLEMENTATION

The CRS provides what is essentially a very simple service – maintaining sets of equivalent URIs – however it has taken several iterations to arrive at the current version 3, which is maintaining co-reference data for each of the `rkbexplorer.com` repositories and enabling the complex cross-repository interoperation required by the RKBExplorer application [7].

What may appear to be a trivially straight-forward service actually delivers a refined yet powerful set of capabilities, which is the result of much thought, deliberation and experience through implementing and using the service to manage real-world data and support complex applications.

The core CRS functionality is implemented in a PHP class, enabling easy integration to a wide variety of web-based applications and middleware libraries, and backed by

a mySQL database to facilitate acceptable performance when used with large datasets.

Equivalent URIs are conceptually stored in a 'bundle' – a set of identifiers referring to resources which are considered to be the same in a given context. A URI can exist in at most one bundle within a CRS instance. One URI in each bundle is nominated to be a canonical identifier, or canon, for that bundle, representing a 'preferred' URI for the set of duplicates. An application that wishes to use data from multiple sources as if they were a single resource can process results by looking up URIs in a CRS and replacing them with their canons on the fly, reducing the multiplicity of identifiers to a single definitive URI. Bundles additionally have sequential numeric identifiers, however these are only used internally and are not exposed.

Bundles are formed by atomic operations only, by means of merging pairs of URIs together. In merging uri_a and uri_b the CRS first checks to see that each URI is already known and exists within a bundle. If not, a 'singleton' bundle is created for new URIs as required. Now to perform the merge, a new third bundle is created consisting of the union of the bundles that contain the URIs which are being asserted as equivalent. The two constituent bundles which were merged are then marked as inactive, as shown in Figure 1.

A number of schemes can be employed to elect the canon for this newly merged bundle, from random allocation, selection by a ordering according to a list of preferred URI domains, or simply by assuming the canon from the bundle in the left hand side of the pair of merged URIs, as in the example above.

In order to handle large datasets, the CRS uses a mySQL database for back end storage. To facilitate fast access when querying the CRS, data is internalised in indexed tables of hashed URIs, according to the schema in Figure 2. This enables simple queries to be formulated which permit extremely fast lookups to find the canon of a given URI, or finding all URIs in a given bundle; the two fundamental query operations and most used features of the CRS.

Each operation performed by the CRS can additionally be logged in a history table, including the facility to record a comment as to why an action was carried out. As a result, if at a later date it is discovered that two URIs were incorrectly deemed to be equivalent, then operations can be 'undone' or rolled back to rectify the situation.

Finally, functionality is provided to 'deprecate' URIs within a dataset, by setting a flag in the *uris* table. A number of sources from which we acquired publications data contained particularly poor quality information with regards to person identifiers, often conflating different individuals who share common names under the same URI. As a result, we were forced to generate a new URI for every author name on every publication, and then perform our own co-reference analysis to collapse equivalent URIs where appropriate [7].

However, this process led to bundles containing many tens or low hundreds equivalent URIs, each from within the same 'local' dataset. These duplicates are of our own creation, provide little additional value, and in fact cause significant overheads if each variant has to be checked by an application. It was decided therefore that once a phase of this 'cold start' co-reference analysis had been completed, the underlying RDF data in the associated knowledge base should be modified to remove unnecessary duplicates by consulting the CRS and re-writing each 'local' URI with the canon

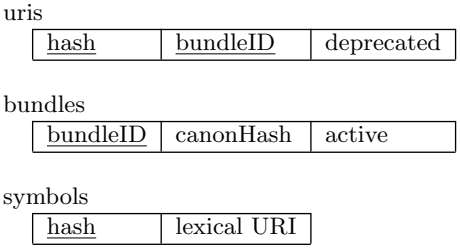

Figure 2: CRS database schema

returned by the CRS for that URI. In removing the unnecessary duplicates, we reduce the number of query iterations that are required to retrieve all possible facts from an equivalence closure. Those duplicates removed from the underlying dataset are flagged as deprecated within the CRS, which continues to give results when asked to give equivalents for both normal and deprecated URIs. However, deprecated URIs are not returned in equivalence sets, hence if the CRS is queried for equivalents of a deprecated URI, only the non-deprecated members of the bundle are returned. All URIs remain in their bundles, maintaining the history and bundle formation structures; deprecated ones are simply filtered out when results are returned. Checks have been put in place to ensure that canons cannot be deprecated, and while it would be perfectly feasible to change the canon for a given bundle to an alternative member of that bundle, we have not found need to implement such functionality.

4. USING A SINGLE CRS

As stated in the previous section, the core functionality of the CRS is implemented in a PHP class. This can be used directly, incorporated within an application, or wrapped in simple scripts to expose functionality via HTTP interfaces. In either case, the back end database does not have to reside on the same machine as the code executing the CRS class, given appropriate mySQL permissions and firewall access, enabling multiple applications to access the same co-reference information directly via PHP. However this obviously may incur additional overheads.

The CRS class provides a function to 'merge' two URIs, i.e. assert that they are equivalent, along with a number of other useful functions to facilitate querying of the underlying knowledge. One can request equivalent URIs for a given input URI, which returns the set of non-deprecated URIs from the bundle in which the requested URI resides. If no information is known about the requested URI, a set is simply returned containing only that URI. Similarly, the canonical URI can be requested for any given input URI.

There is no level of access control built in to the core functionality, other than authenticating to the mySQL database. As a result, we have chosen not to give public access to the `rkbexplorer.com` CRSes via the CRS class, rather to provide a number of web interfaces which permit read-only querying of the co-reference knowledge.

For each `rkbexplorer.com` sub-domain, the URI naming scheme uses the following pattern for non-information resources: `http://<repository>.rkbexplorer.com/id/xyz`. When a non-information resource is dereferenced with `Accept: application/rdf+xml` an RDF representation is

returned as expected within linked data best practice. However, in this document, there is an additional link via the `coref:coreferenceData` predicate, indicating that there is co-reference data available at the URI `http://<repository>.rkbexplorer.com/crs/xyz` and allowing CRS aware applications to discover related CRSes. This URI produces a representation of the bundle for the `/id/xyz` URI in either HTML or RDF, based on content negotiation, such as that in Figure 3. Alternatively, to facilitate use by a wider number of systems, a request can be made which returns a document in ntriples format describing the canon to be `owl:sameAs` all other duplicates in the bundle.

Applications may also wish to query the CRS in a more general sense, which is provided by the interface accessible via `/crs/export/?term=<uri>&format=<format>`

The core CRS implementation can handle arbitrary URIs from any number of sources, however the HTTP interfaces described above and used with `rkbexplorer.com` sub-domains have the implication that at least one of the URIs in any pair of equivalence assertions comes from the sub-domain for which that CRS is representative. As a result, each `rkbexplorer.com` CRS maintains co-references between URIs on that sub-domain, in addition to links to equivalent URIs in other `rkbexplorer.com` sub-domains or external Linked Data sources.

Finally, each CRS instance, or database, is assumed to contain knowledge according to a single co-reference context only. Unfortunately ontologies have not yet been defined for encapsulating the contextual aspects of co-reference analysis or the use of co-reference information; hence an application must currently either have prior knowledge of a set of CRSes it may consult, or accept data 'carte-blanche' from any CRS it discovers.

5. USING MULTIPLE CRSES

It is conceivable that linked data providers may wish to publish co-reference information about their dataset, representing equivalences both between local URIs and linking to external URIs in other sources. Typically we envisage that providers could host one (or more) CRSes per dataset, as demonstrated with `rkbexplorer.com`. When investigating co-reference for a given URI, application developers may choose to treat a CRS which exists on the same domain as the URI in question as a first point of call, or as more 'authoritative' than other CRSes published elsewhere, however this is not a prescribed semantic.

We have seen how the separation of co-reference data into CRSes allows for additional services to be provided that could not easily be achieved with `owl:sameAs` approaches. Another of these is the use of multiple CRSes to efficiently deduce a global equivalence closure for finding duplicates for a given URI. Finding all equivalences is simply a matter of following the `coref:coreferenceData` links to the bundle for that URI and recursively repeating the process for each URI in that bundle.

There are various methods that can speed up this process, such as only looking at one URI from each CRS repository, or following only the `coref:canon` predicates in order to build up a unified view of equivalent URIs. It is also possible for an application to maintain a list of known CRSes applicable to a given context, and to query each one in parallel to discover any equivalences it knows about, or to naïvely query the `<repository>.rkbexplorer.com/crs/` CRS whenever a `<repository>.rkbexplorer.com/id/` URI is encountered.

In comparison, Semantic Web applications that rely on `owl:sameAs` to represent all co-references must always recursively load and potentially compute inference over the data of each URI that is deemed equivalent to the current URI in order to compute a global equivalence closure. This may bring significant performance overheads, imposing unnecessary loading and processing of large chunks of data. Furthermore, there are no opportunities to limit or control the expansion of the equivalence set, whereas the CRS architecture allows for following as many, or as few duplicate URIs as required with no significant barrier on performance.

We have provided CRS instances for each of our `rkbexplorer.com` sub-domains, and performed significant co-reference analysis both internally and across these datasets. A visualisation of the cross-repository linkage is presented in Figure 4, and experimental voiD descriptions [1] are provided at `http://<repository>.rkbexplorer.com/id/void` detailing these linkages and CRS content in a semantically annotated manner.

This set of CRSes informs our faceted browser application, RKBExplorer, enabling data from the various repositories to be incorporated as required. Although rigorous performance and load testing has not been carried out on the CRS implementation, managing millions of URIs in tens of millions of bundles has presented no problems. Indeed, fetching the global equivalence closure is an insignificant step when compared to other processing and analysis phases within the application. It is anticipated that individual CRSes will scale well beyond the current usage, and even more so when multiple CRSes are employed.

The global equivalence closure described above has been implemented within the RKBExplorer application, and additionally exposed through an HTTP interface at `http://www.rkbexplorer.com/sameAs/`. This service will consult all necessary CRSes to determine the overall set of equivalents for a given URI, while additionally picking a canon from a preferential order of domains. Again, to enable easy integration of CRS knowledge in non CRS aware applications, the service can simply be queried with content negotiation or the additional parameter `&format=n3` to retrieve a document listing the equivalence relationships using the `owl:sameAs` predicate.

6. CONCLUSIONS

This paper has briefly outlined the problems of co-reference resolution within Open Linked Data repositories and on the Semantic Web as a whole. The problems of using `owl:sameAs` have been discussed, and the needs of more capable management techniques presented. We detail the rationale, capabilities and implementation of the CRS architecture, and describe its use in real-world applications.

Co-reference within the Semantic Web is a growing, yet largely unappreciated problem. It has been suggested that it is a matter that will resolve as the Semantic Web evolves, with careful social engineering and planning, however due to the reasons discussed previously we do not believe this to be the case.

It is our conclusion that the most effective means for combating the issue is to make co-reference awareness an architectural feature of future semantic applications. Existing use of `owl:sameAs` is not sufficient, and in many cases incorrect. We believe the use of the bundle framework provides a

```
<rdf:RDF xmlns:coref="http://www.rkbexplorer.com/ontologies/coref#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <coref:Bundle>
    <coref:canon rdf:resource="http://southampton.rkbexplorer.com/id/person-00021"/>
    <coref:duplicate rdf:resource="http://acm.rkbexplorer.com/id/person-102898" />
    <coref:duplicate rdf:resource="http://citeseer.rkbexplorer.com/id/resource-CSP109002" />
    <coref:duplicate rdf:resource="http://dblp.rkbexplorer.com/id/people-27aedbcb" />
    <coref:duplicate rdf:resource="http://eprints.rkbexplorer.com/id/kfupm/person-27aed0c1" />
    <coref:duplicate rdf:resource="http://southampton.rkbexplorer.com/id/person-00021" />
    <coref:duplicate rdf:resource="http://wiki.rkbexplorer.com/id/hugh_glaser" />
    <coref:lastUpdated>2009-01-16 11:11:40</coref:lastUpdated>
  </coref:Bundle>

</rdf:RDF>
```

Figure 3: Example RDF description of equivalent URIs in a bundle

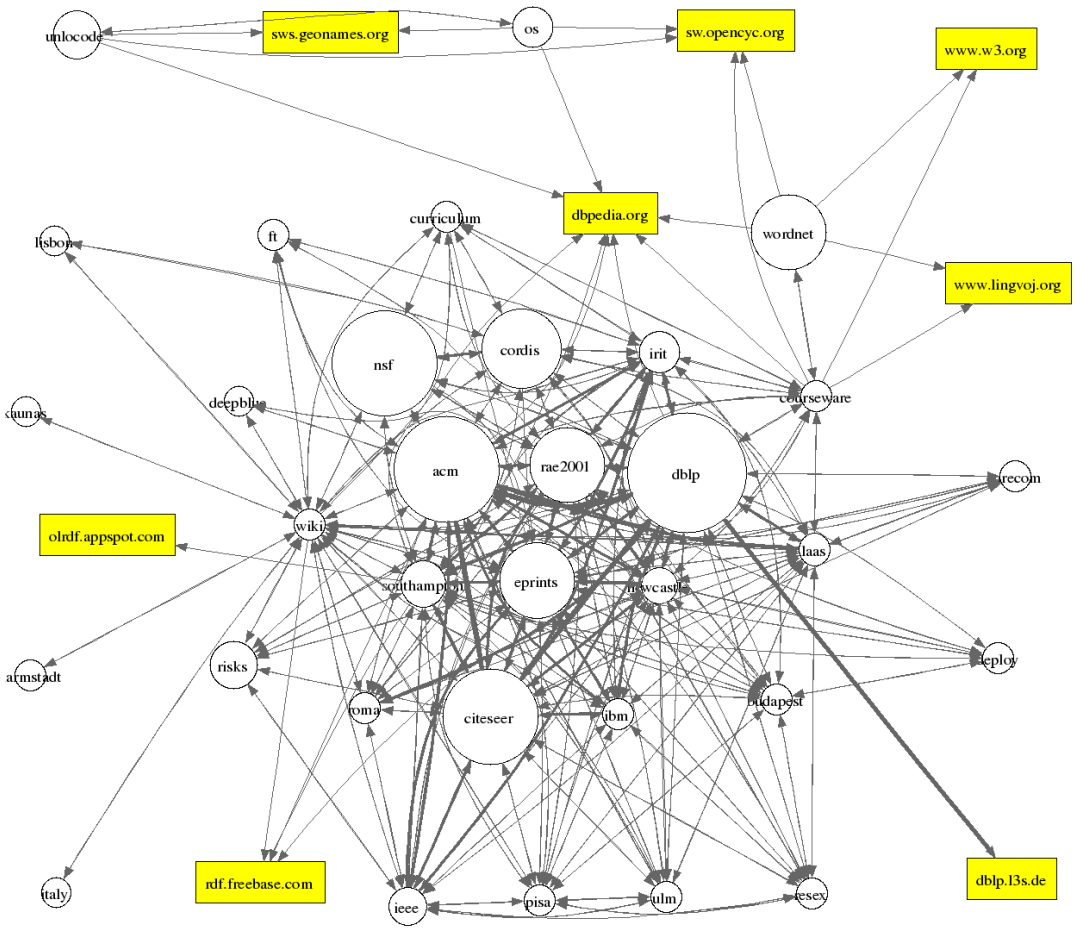

Figure 4: Co-references between CRSes – see http://www.rkbexplorer.com/linkage/

flexible, expandable and readily compatible notation for conceptualising co-reference, and that the CRS implementation provides a broad strategy for co-reference management that integrates the process of reference management into the architecture of the Semantic Web by utilising both social and technical engineering.

Readers are encourage to experiment with and if possible make use of the `rkbexplorer.com` services discussed in this paper, and we welcome any feedback. The core CRS implementation may be available on request.

7. ACKNOWLEDGMENTS

This work is funded in part by the ReSIST Network of Excellence (NoE) which is sponsored by the EU Sixth Framework programme (FP6) under contract number IST-4-026764-NOE, and in collaboration with The Korea Institute of Science and Technology Information (KISTI).

We would also like to thank our colleagues at Southampton, Newcastle, and DERI, along with numerous members of the Linking Open Data community who have contributed both directly and indirectly through informative and enlightening discussion.

8. REFERENCES

[1] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. voiD Guide - Using the Vocabulary of interlinked Datasets, 2008.

[2] S. Bechofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. Mcguiness, P. Schneider, and L. Stein. OWL Web Ontology Language Reference, 2004.

[3] C. Bizer, R. Cyganiak, and T. Heath. How to publish Linked Data on the Web, 2007.

[4] D. Booth. Why URI Declarations? A Comparison of Architectural Approaches. In *1st Workshop on Identity and Reference for the Semantic Web (IRSW2008)*, 2008.

[5] R. Cyganiak and C. Bizer. D2R Server – Publishing Relational Databases on the Web as SPARQL Endpoints. In *15th International World Wide Web Conference (WWW2006)*, 2006.

[6] H. Glaser, I. C. Millard, T. Anderson, and B. Randell. ReSIST Deliverable D10: Prototype Knowledge Base, 2006.

[7] H. Glaser, I. C. Millard, T. Anderson, and B. Randell. ReSIST Deliverable D23: Resilience Knowledge Base – Version 2, 2007.

[8] P. Hayes and H. Halpin. In Defense of Ambiguity. In *Workshop on Identity, Identifiers and Identification (WWW2007)*, 2007.

[9] A. Jaffri, H. Glaser, and I. C. Millard. URI Disambiguation in the Context of Linked Data. In *1st Workshop on Linked Data on the Web (LDOW2008)*, 2008.