Context Dependent Reasoning for Semantic Documents in Sindice*

Renaud Delbru and Axel Polleres and Giovanni Tummarello and Stefan Decker

Digital Enterprise Research Institute
National University of Ireland, Galway
Galway, Ireland

Abstract. The Sindice Semantic Web index provides search capabilities over today more than 30 million documents. A scalable reasoning mechanism for real-world web data is important in order to increase the precision and recall of the Sindice index by inferring useful information (e.g. RDF Schema features, equality, property characteristic such as inverse functional properties or annotation properties from OWL). In this paper, we introduce our notion of context dependent reasoning for RDF documents published on the Web according to the linked data principle. We then illustrate an efficient methodology to perform context dependent RDFS and partial OWL inference based on a persistent TBox composed of a network of web ontologies. Finally we report preliminary evaluation results of our implementation underlying the Sindice web data index.

1 Introduction

Reasoning over semantically structured documents enables to make explicit what would otherwise be implicit knowledge: it adds value to the information and enables an entity-centric search engine to ultimately be much more competitive in terms of precision and recall [13]. The drawback is that inference can be computationally expensive, and therefore drastically slow down the process of indexing large amounts of semantic documents.

Our work, implemented within the Sindice project [16], is specifically concerned on how to efficiently reason over very large collections of semantic documents which have been published on the Web. To reason on such documents, we assume that the ontologies that these documents refer to are either included explicitly with `owl:imports` declarations or implicitly by using property and class URIs that link directly to the data describing the ontology itself. This later case should be the standard if the W3C best practices [15] for publishing ontologies and the Linked Data principles [2] are followed by ontology creators. As ontologies might refer to other ontologies, the import process then needs to be recursively iterated (see Section 2.2 for a detailed formalisation).

* This material is based upon works supported by the European FP7 project *Okkam - Enabling a Web of Entities* (contract no. ICT-215032), and by Science Foundation Ireland under Grant No. SFI/02/CE1/I131. We thank Jérôme Euzenat for the useful discussions on the paper.

A naive approach would be to execute such recursive fetching for each harvested document and create a single model composed by the original document plus the ontologies. At this point the deductive closure (see Sect. 2.3 for a detailed formalisation) of the document can be computed and the document – including ontologically inferred information – can be indexed.

Such a naive procedure is however obviously inefficient since a lot of processing time will be used to recalculate deductions which, as we will see, could be instead reused for possibly large classes of other documents during the indexing procedure.

To reuse previous inference results, a simple strategy has been traditionally to put several (if not all) the ontologies together compute and reuse their deductive closures across all the documents to be indexed. While this simple approach is computationally convenient, it turns out to be sometimes inappropriate, since data publishers can reuse or extend ontology terms with divergent points of view.

For example, if an ontology other than the FOAF vocabulary itself extends `foaf:name` as an inverse functional property, an inferencing agent should not consider this axiom outside the scope of the document that references this particular ontology. Doing so would severely decrease the precision of semantic querying, by diluting the results with many false positives.

For this reason, a fundamental requirement of the procedure that we developed has been to confine ontological assertions and reasoning tasks into contexts in order to track provenance of inference results. By tracking provenance of each single assertion, we are able to prevent one ontology to alter the semantics of other ontologies on a global scale.

The main contribution of this paper is to provide an efficient methodology for reasoning large amounts of RDF documents as retrieved from the Web. Other contributions of this paper are 1. an adaptation and application of the context mechanism of [11] for linked data; 2. a conceptual model of a contextualised reasoning engine.

In the following Section 2, we introduce the basic concepts and settings of our approach. Section 3 describes the context dependent reasoning procedure based on a persistent TBox, called ontology base. The conceptual model of the ontology base is detailed and our query and update strategies are discussed. Section 4 presents the distributed prototype implementation and preliminary results of an empirical evaluation before reviewing related work in Section 5 and concluding in Section 6.

2 Preliminaries

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 Sindice. 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 [3]). The ABox and TBox[1] 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.

2.1 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 [5]. 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 [11] 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 [10]. The notation $ist(c, \varphi)$ states that a proposition φ is true in the context c.

The notion of context presented in this paper 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.

Lifting Rules Since contexts are first class objects, it becomes possible to define expressive formulae whose domains and ranges are contexts. An example 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 range 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.

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

2.2 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 Sindice, 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 [15] and Linked Data principles [2], 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 correspond 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 ([11]). *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:*

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

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 [1]. 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:*

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

2.3 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 [6], 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 [20] as implemented by off-the-shelf rule-based materialisation engines such as for instance OWLIM[2]. 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 [12]). 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 φ_3 a* newly entailed proposition *in the aggregate context $c_1 \wedge c_2$. We denote the set of all newly defined propositions Δ_{c_1,c_2}, i.e.,*

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

[2] `http://www.ontotext.com/owlim/`

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 property:[3] if aggregate context $c_1 \subseteq c_2$ then $ist(c_2, \phi)$ implies $ist(c_1, \phi)$, or respectively, for overlapping contexts, if $ist(c_1 \cap c_2, \phi)$ implies both $ist(c_1, \phi)$ and $ist(c_2, \phi)$. We will exploit this property in the following outlined reasoning procedure.

3 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.

3.1 Ontology Base Concepts

The ontology base uses the notion of named graphs for modeling 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 to localise 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.

[3] 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 [15], 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 pointer to a frame of knowledge that is necessary for completing the semantics of O_A and are mapped to an *importsFrom* lifting rule.

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

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

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

where a path is a sequence of n vertices $O_A, O_B, \ldots, O_n$ *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.

3.2 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.

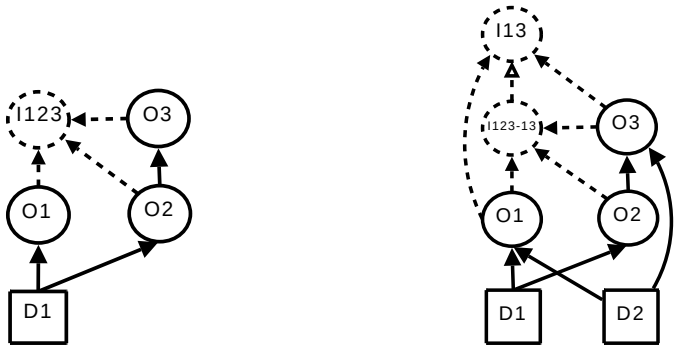

Fig. 1. On the left, the ontology network after processing a document D_1. On the right, after processing D_2, the virtual context I_{123} has been split in 2 parts.

3.3 Ontology Base Querying Strategy

When a document imports an ontology O, we query the ontology base to lift the import closure $G_O = (\mathcal{O}', \mathcal{L}')$ (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 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 Hadoop[4]. Each Hadoop node has its own ontology base which support 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.

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 lift the content of the refered 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 framework.[5]

4.1 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.

4.2 Preliminary Resuts

Using the experimental setup described in the previous section, we are able to process[6] 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 in Sindice can be checked online on the Sindice web site[7]. For example, for the search result "tim berners lee"[8], the hyperlink "Cached"[9] will display the cached version of the

[4] Hadoop: `http://hadoop.apache.org/core/`

[5] OpenRDF Sesame: `http://www.openrdf.org/`

[6] including document pre-processing and post-processing (metadata extraction, indexing, etc.)

[7] Sindice.com: `http://www.sindice.com`

[8] `http://sindice.com/search?q=tim\%20berners\%20lee&qt=term`

[9] `http://sindice.com/search/page?url=http\%3A\%2F\%2Fwww.w3.org\%2FPeople\%2FBerners-Lee\%2Fcard`

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 by Sindice. We observe that on a snapshoot 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 in Sindice 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.

5 Related Work

The notion of context has been extensively studied since early 1990s, starting with a first formalisation by Guha [10] followed by McCarthy [14] and by Giunchiglia [9]. For more information about the domain, [18] 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 [4, 11].

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 [19] 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 [7] 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 does not try to maintain consistency, instead we just maintain the provenance of each inferred statements.

A side remark in [8] 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.

6 Conclusion

In this paper, we discussed a context dependent methodology for large scale web reasoning.

We first define the problem conceptually and then illustrate 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 support a subset of RDFS and OWL. We find this level of inference

support to be in line with Sindice's 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 [20].

Reasoning of the Web of Data enables the Sindice search engine 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 Sindice 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 paper 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.

References

[1] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. W3C Recommendation, February 2004.

[2] T. Berners-Lee. Linked data. W3C Design Issues, July 2006.

[3] P. Bouquet, C. Ghidini, F. Giunchiglia, and E. Blanzieri. Theories and uses of context in knowledge representation and reasoning. *Journal of Pragmatics*, 35:455–484(30), March 2003.

[4] Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. Contextualizing ontologies. *Journal of Web Semantics*, 1(4):325–343, 2004.

[5] J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.

[6] M. d'Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, and E. Motta. Characterizing Knowledge on the Semantic Web with Watson. In *EON*, pages 1–10, 2007.

[7] Johan de Kleer. An Assumption-Based TMS. *Artif. Intell.*, 28(2):127–162, 1986.

[8] Li Ding, Jiao Tao, and Deborah L. McGuinness. An initial investigation on evaluating semantic web instance data. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1179–1180, New York, NY, USA, 2008. ACM.

[9] Fausto Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, 345:345–364, 1993.

[10] R. V. Guha. *Contexts: a formalization and some applications.* PhD thesis, Stanford, CA, USA, 1992.

[11] R. V. Guha, R. McCool, and R. Fikes. Contexts for the Semantic Web. In *International Semantic Web Conference*, pages 32–46, 2004.

[12] P. Hayes. RDF Semantics. W3C Recommendation, World Wide Web Consortium, February 2004.

[13] J. Mayfield and T. Finin. Information retrieval on the Semantic Web: Integrating inference and retrieval. In *Proceedings of the SIGIR Workshop on the Semantic Web*, August 2003.

[14] John McCarthy. Notes On Formalizing Context. In *Proceedings of IJCAI-93*, pages 555–560, 1993.

[15] A. Miles, T. Baker, and R. Swick. Best Practice Recipes for Publishing RDF Vocabularies. Technical report, 2006.

[16] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1), 2008.

[17] A. Polleres, C. Feier, and A. Harth. Rules with contextually scoped negation. In *3rd European Semantic Web Conference (ESWC2006)*, volume 4011 of *LNCS*, Budva, Montenegro, June 2006. Springer.

[18] Luciano Serafini and Paolo Bouquet. Comparing formal theories of context in AI. *Artificial Intelligence*, 155(1-2):41, 2004.

[19] Heiko Stoermer, Paolo Bouquet, Ignazio Palmisano, and Domenico Redavid. A Context-Based Architecture for RDF Knowledge Bases: Approach, Implementation and Preliminary Results. In *RR*, pages 209–218, 2007.

[20] H. J. ter Horst. 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, 2005.