Contextualization of a RDF Knowledge Base in the VIKEF Project

Heiko Stoermer[1], Ignazio Palmisano[2], Domenico Redavid[2], Luigi Iannone[3], Paolo Bouquet[1], and Giovanni Semeraro[2]

[1] University of Trento,
Dept. of Information and Communication Tech.,
Trento, Italy
{stoermer, bouquet}@dit.unitn.it
[2] Dipartimento di Informatica, Università degli Studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy
{palmisano, redavid, semeraro}@di.uniba.it
[3] Computer Science Department, Liverpool University
Ashton Building, Ashton Street, L69 BX Liverpool, UK
luigi@csc.liv.ac.uk

Abstract. Due to the simplicity of RDF data model and semantics, complex application scenarios in which RDF is used to represent the application data model raise important design issues. Modelling e.g. the temporary evolution, relevance, trust and provenance in Knowledge Bases require more than just a set of universally true statements, without any reference to a situation, a point in time, or generally a context. Our proposed solution is to use the notion of context to separate statements that refer to different contextual information, which could so far not explicitly be tied to the statements. In this paper we describe a practical solution to this problem, which has been implemented in the VIKEF project, which deals with making explicit and intelligently useable information contained in vast collections of documents, databases and metadata repositories.

1 Problem Description and Motivation

The VIKEF project[1] deals with creating large-scale information systems that base on Semantic Web technology. At the center of the envisioned systems there is an RDF (Resource Description Framework)[2] knowledge base (KB) that contains a large amount of information about documents and their contents. This information is gathered by information and knowledge extraction processes at the base level, then semantically enriched and related to ontological knowledge, and finally stored in a RDF triple store called RDFCore, which will be described

[1] European Commission 6^{th} Framework Programme IST Integrated Project VIKEF - Virtual Information and Knowledge Environment Framework (Contract no. 507173, Priority 2.3.1.7 Semantic-based Knowledge Systems http://www.vikef.net)
[2] http://www.w3.org/TR/rdf-concepts/

S. Sugimoto et al. (Eds.): ICADL 2006, LNCS 4312, pp. 101–110, 2006.

in more detail in Sect. 3.1. On top of this KB, semantic-enabled services will be implemented to provide a next-generation information system.

Current RDF triple stores are built to represent a single bag of RDF triples, i.e. all statements are stored in the same information space together. However, from a Knowledge Representation point of view, RDF statements in general are context-free, and thus follow a notion of *universal truth*, while in our opinion knowledge in an information system is context-dependent. In effect, without a context-based system, it is possible (and probable) that semantically contradictory statements will be stored in the KB, such as for instance "Silvio Berlusconi is the Prime minister of Italy" and "Romano Prodi is the Prime minister of Italy" as the result of knowledge extracted from articles written in different years (supposing that being Prime Minister is possible only for a single person). These contradictions are however unwanted in a logical system because they would interfere with both simple queries over the data (e.g., the question "who is the Premier in Italy?" brings two answers instead of just one) and higher level reasoning that is to be performed to provide Semantic Web functionality, such as semantic browsing, search, visualization, etc. Additionally, we would like to be able to model other aspects such as relevance, credibility and validity of a statement, all of which require further qualification.

If we think about the Semantic Web as a whole, with a large number of uncoordinated information systems, the problem becomes even more evident. If every peer builds up a KB of unqualified RDF statements, the set of universally true facts in the Semantic Web becomes enormously large and impossible to handle from a semantic point of view; this is the case when, for example, tools for automatic extraction of metadata are used, as in [5] and [11]. In our opinion, such contradictions, contradictory beliefs and facts that become semantically incorrect in the absence of additional pragmatic or contextual information are likely to impose serious problems on the coordination and interoperation of information systems in the Semantic Web.

The remainder of the paper is organized as follows: after giving some definitions of *context* in Sect. 2, we present our architecture in Sect. 3. Some empirical evaluation results are presented in Sect. 4, and finally we draw some conclusions and future work directions in Sect. 5.

2 Context in RDF Knowledge Bases

We think that the mentioned issues can be approached by introducing the notion of *context* into RDF, to limit the scope of a RDF statement to the context in which it is relevant or valid, because in our opinion this is required for anything sensible to be expressed in the Semantic Web. We want to present a mechanism to qualify statements and thus to model that a statement is true *only under a certain set of conditions*, which will help us store information in the KB that would cause contradictions or inconsistencies in a plain RDF A-Box[3].

[3] In Description Logic, an A-Box is the set of assertions about instances (Assertional Box), while the T-Box is the portion of the KB containing the axioms, such as class and property definitions (Theoretical Box).

2.1 Context in KR - Multi Context Systems

The theoretical ideas presented in this paper base on the logical theory of *Multi Context Systems* and the principles of *Locality* and *Compatibility* presented e.g. in [8], with influences from [3,4]. Basically, this theory states that contexts can be seen in a peer-to-peer view, resembling more general aspects such as human beliefs, agent knowledge or distributed systems. The important aspect of this theory is that reasoning within a context follows standard mechanisms, as the non-elementary view on the large part of the axioms does not require to keep track of the context they are relevant for. Relations between contexts however, i.e. to reason across contexts, are to be expressed in so-called *compatibility relations* (CRs), that formalize exactly how under certain circumstances knowledge from other contexts becomes relevant. Regarding RDF in this case we claim that a RDF context can be thought of as a locally coherent set of axioms, each one with a set of parameters and values for these parameters, that specify the conditions under which the set of axioms is valid. We envision CRs to be modeled as a *semantic attachment* [12], as we will describe in more detail below.

2.2 Main Idea

The basic idea is to have all statements that belong to a context in a separate named RDF graph, and extend the RDF semantics in a way to enable contexts to appear as standard objects in RDF statements of other contexts. As we will illustrate in more detail in Sect. 3.2, for a reference implementation we will base on features of the SPARQL[4] query language.

Then, we want to model the mentioned CRs between contexts, to allow for reasoning across contexts. This aspect is probably the most important one, because from an application perspective it is crucial that sensible queries can be issued and *all* relevant information is taken into account - which requires reasoning across contexts and reasoning on the relations between contexts (i.e. on statements of the form <c_x R c_y> where c_x and c_y are RDF Contexts, or <f R c'> respectively <c' R f> with $f \in c$). We are only starting to explore in full depth the aspects of CRs that are relevant for the VIKEF project.

Several approaches can be thought of to model CRs in our architecture. First of all, one could think of allowing the implementer of an information system to provide their own vocabularies (ontologies) to describe relations between contexts. A similar option would be for us to provide such an ontology as part of the architecture. However, in our opinion the basic problem with these approaches is the fact that many interesting relations between architectures cannot be fully formalized with the help of a Semantic Web ontology, which is based on Description Logics.

As an example for this claim take a relation such as

<c' EXTENDS c>

which expresses that c' represents an extension to c, e.g. for the reason that it is about the same object, but composed at a later point in time. The underlying

[4] http://www.w3.org/TR/rdf-sparql-query/

assumption of the *EXTENDS* relation is that the two contexts are compatible, i.e. they agree on the relevant context parameters. The semantics of this relation have to be expressed algorithmically:

```
if c and c' are compatible

then if no answer to a query q can be given in c

propagate query to c'
```

One of the questions that might arise is how these CRs are supposed to be modeled. At the moment, we see three approaches to do this, which, among other basic and preliminary results including some of the above ideas, have been presented in [1], which we recommend to the reader for more detailed information, references and a discussion of related work.

This work has led us to the conclusion that the approach to be chosen is to implement a CR as a *semantic attachment* [12], which can be thought of as a sort of plugin to the system, one attachment per CR. This has the positive effects that i) there is no restriction on how many and which kind of CRs are part of such a system and ii) implementation of the CRs is generally not restricted to any specific language or system.

2.3 Related Work

As mentioned before, [1] provides a discussion of relevant related work. The only related approach that has lead to actual results, up to our knowledge, is that of the W3C Named Graph Interest Group[5]. A substantial article has been published in 2005 [2], and implementational results are now part of the Named Graphs API for Jena (NG4J) [6]. The approach describes a way to represent a graph as an object in a RDF KB, and has mainly been driven by the need for developing a trust model in RDF, but it could also serve as an underlying implementation in order to provide a base for the CRs discussed above.

3 The Proposed Solution: System Architecture

Our practical solution to context issues is based on the following requirements:

- Easy and simple identification of contexts
- Separate and independent storage for each context
- Easy querying of one or more contexts
- Easy reasoning on context parameters values
- Ability to plug new CRs in the architecture
- Ability to use CRs of higher expressive level, i.e. higher than OWL and/or DL

[5] http://www.w3.org/2004/03/trix/

[6] http://www.wiwiss.fu-berlin.de/suhl/bizer/ng4j/

As sketched in Fig. 1, the two main parts of our implementation are what is "inside" RDFCore (i.e. the RDF storage level) and "outside" of it. In Sect. 3.1 and Sect. 3.2 we will discuss the details of the architecture.

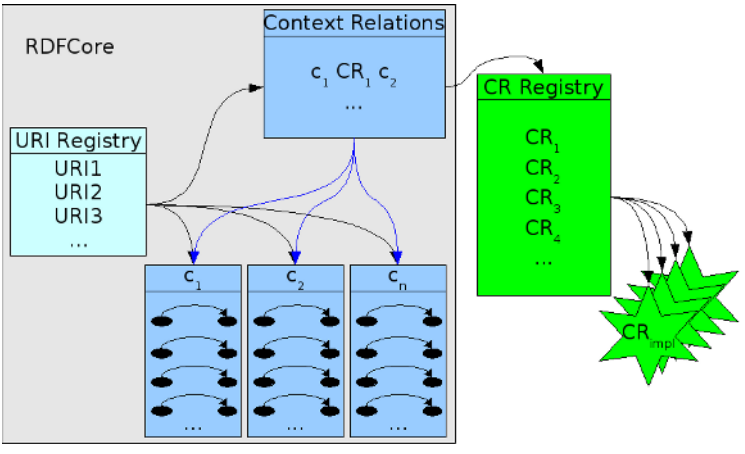

Fig. 1. Compatibility Relation Association Architecture

3.1 RDF Storage

As RDF storage, the VIKEF project chose to use RDFCore: presented in [7], it is a component used for storage and retrieval of RDF graphs, including multiuser support and extensible support for query languages.

In the VIKEF Project, RDFCore is the basic component for RDF metadata storage; being the VIKEF architecture based on the Web Services paradigm, its SOAP[7]-exposed services have been wrapped as a Web Service[8] for metadata storage, retrieval and querying.

RDFCore also has extensible support for different physical persistence solutions. At the time of writing, there are three implementations of *RDFEngineInterface* (the basic interface to be implemented by plugins), two based on the Jena Toolkit[9], one with MySQL RDBMS[10] as persistent storage, called *RDFEngineJENA*, and the other one using Microsoft SQL Server[11], called *RDFEngineMsSQL*. The third implementation is based on simple RDF/XML files, and is called *RDFEnginePlain*. All these implementations are based on the Jena API.

The component also offers multiuser support; users can choose whether some of the models they own should be private, publicly readable or writable, and can restrict access to single users or groups of users. This support is useful

[7] http://www.w3.org/2000/xp/Group/
[8] http://www.w3.org/2002/ws/
[9] http://jena.sourceforge.net
[10] http://dev.mysql.com/doc/mysql/en/index.html
[11] www.microsoft.com/sql/

when designing cooperative applications, thus enabling geographically dispersed teams to work together easily. RDFCore also can use a graph redundancy check algorithm (REDD)[6], which is useful in searching redundant portions of RDF graphs, i.e. those parts of the models that do not carry semantic information, or that duplicate information carried by other parts.

3.2 Context Querying: SPARQL

We identified SPARQL as the query language that satisfies many of the requirements listed before, since it includes facilities to query more than one RDF model at a time, and the models to use can be specified with URIs. With this approach, a context can be easily represented as a RDF model, identified by a URI – in other words, it can be viewed as a named graph. The only step needed to complete the pipeline and enable a generic repository to answer a SPARQL query on multiple contexts is the retrieval machinery to provide the RDF data for the SPARQL *Dataset* to the SPARQL engine.

We use ARQ[12] as SPARQL engine for RDFCore; since ARQ uses the Jena class `com.hp.hpl.jena.util.FileManager` in order to retrieve the RDF data needed to build the *Dataset* for the SPARQL query, this is the point in which we insert our mappings from graph names to URLs that the RDFCore component has to supply. Since RDFCore has multiuser support, however, it is necessary to implement a check on whether the user making the query has read access to the involved models; to do this, RDFCore extracts the graphs' URIs and checks that all the required models are accessible before pushing the SPARQL query to ARQ. Access to the data is done by ARQ through the use of a RDFCoreLocator, which implements the Locator interface defined in Jena. A small sketch of the process is depicted in Figure 2.

When the query is issued by an external application using the SPARQL protocol[13], the query can be executed only when all involved models are readable by any user (thus including any application that does not act on behalf of a user, and therefore has no explicit access to any model). At the moment, RDFCore satisfies only the basic requirements for the SPARQL Protocol (HTTP and SOAP access), that is, RDFCore only accepts SPARQL queries with embedded dataset, where the dataset is composed of URIs that are registered as identifiers for RDF models publicly accessible in RDFCore. Accessing these models is realized through simple HTTP connection to a related RDFCore service, and it is automated in the query component through the implementation of the *Locator* interface in the Jena API, that is used as input to create the RDF dataset in the ARQ component. The use of the SPARQL protocol simplifies the design of those VIKEF components that only need read access to specific RDF models; in the case of distinct contexts, this is an easy way to ensure that no application can modify the information contained in a specific context.

[12] `http://jena.sourceforge.net/ARQ/`
[13] `http://www.w3.org/TR/rdf-sparql-protocol/`

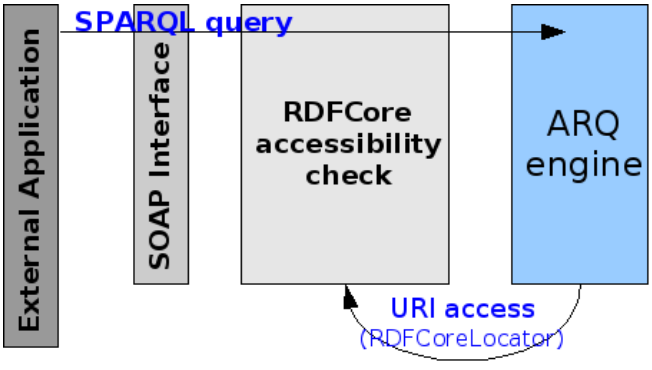

Fig. 2. SPARQL query processing

3.3 System Architecture

As sketched in Figure 1, the main parts of the architecture are:

- A *URI Registry*, used by applications to get the list of context URIs contained in a particular instance of RDFCore (including accessible and non accessible ones).
- A *Compatibility Relations model*, containing statements of the kind <c' R c>, meaning that context c' is in relation R with context c (all three should be read as URIs for the contexts and the relation).
- A *Compatibility Relation Registry*, where each URI that identifies a CR is related to an implementation for that CR (*semantic attachment*).

The architecture presented so far is quite straightforward; however, the reasoning task on the *Compatibility Relations model* (i.e. the model containing the CR statements between contexts) cannot be carried out by a DL reasoner, since the complete semantics of the CRs we want to represent exceeds OWL expressiveness. In order to overcome this limitation of the architecture, we devised a plugin-oriented solution, where the URI of a CR identifies a plugin that implements the correct behavior to be carried out. As an example, consider a CR saying that:

"context $context : x$ and context $context : y$ are $context : compatible$ if they have no contradictory statements"[14].

The relation named $context : compatible$, then, has to be inferred (or verified) through the use of some code that has to be associated with the relation, which in this case would do the job of taking the RDF models for the two contexts (i.e. the RDF models labeled $context : x$ and $context : y$, available in RDFCore). A reasoner should then be used on the whole resulting RDF graph in order to evaluate consistency.

[14] Note that no specific reasoner level is set here: a real rule should also specify *how* to verify contradiction.

4 Empirical Evaluation

Empirical evaluation of the contextualized KB can be focused on two main aspects: i) scalability w.r.t. the number of contexts and their size, and ii) scalability w.r.t. number and complexity of Compatibility Relations. So far, we have evaluated the first aspect.

4.1 KB Design

In order to evaluate scalability w.r.t. the number of contexts that can be queried at once, we produced a sample KB containing many artificial RDF models, where each model represents a Context, and we then ran a SPARQL query of the kind:

```
CONSTRUCT \{?x ?y ?z\} FROM <urn:a1> FROM
<urn:a2> ... WHERE \{?x ?y ?z\}
```

where *urn:a1* represents the URI of a specific context and is used to retrieve the corresponding model from RDFCore. This query template simply retrieves all triples from the models named in the *FROM* clauses, and in our experiment we use queries that involve 10, 20 and 100 models respectively; in the first phase of testing, all the models have 100 statements in them, while in the second phase all the models have 1000 statements, so the total number of statements retrieved by a query scales from 1000 to 100000; the results are presented in Table 1. The last column of Table 1 shows the results obtained executing the same query on a single model containing the same number of triples of the union of the models, in order to verify the performance impact of partitioning a model into smaller contexts. As it emerges from the data, the performance overhead is small and tends to decrease when the total number of statements increase; the growth in the elapsed time has the same trend for both approaches, so we deduce that our architecture does not affect performances in a negative way, for such simple queries (however, note that any complex query is likely to retrieve a small number of statements w.r.t. the size of the model, so these very general queries are stressing the framework more than a very restrictive query that would only retrieve a single triple).

Table 1. Test Results

(artificial) graphs	stmt/graph	stmt retrieved	elapsed secs	elapsed secs on whole models
First Phase				
10	100	1000	~ 0.4	~ 0.1
20	100	2000	~ 0.6	~ 0.25
100	100	10000	~ 2.5	~ 1.5
Second Phase				
10	1000	10000	~ 2	~ 1.5
20	1000	20000	~ 4	~ 3.5
100	1000	100000	~ 20	~ 24

5 Conclusions and Future Work

We presented a possible solution to the issues related to uncontextualized knowledge, mostly arising from the notion of *universal truth* that RDF model semantics follows, and showed the architecture of our implementation for this solution. Future work we plan to do on this implementation consists of:

- A thorough stress test for the RDFCore component that acts like a "context" server in our architecture, to check for scalability issues w.r.t. CR number and complexity.
- Some implementations of CR "attachments", in order to provide the system with the needed expressive power to match VIKEF requirements.

Possible applications for this kind of KR are manifold, as partly described in [1,2,9,10]. Aspects such as beliefs, trust, incomplete knowledge and KB evolution in our opinion can all be tackled with a sensible context system as a base. We believe that in the long run, the vast amount of knowledge represented in the Semantic Web can only be handled properly if represented *in context*.

Additionally, we envision the outcomes of this work to go beyond local aspects and also become relevant from a distributed point of view. As the nature of the Semantic Web is inherently distributed, we think we can contribute to the semantic coordination of Semantic Web agents, firstly by offering the capabilities to make explicit that two knowledge bases belong to their respective agents and to enable the agents to establish semantic links to the KBs of other peers with the help of CRs.

Acknowledgments

This research was partially funded by the European Commission under the 6^{th} Framework Programme IST Integrated Project VIKEF - Virtual Information and Knowledge Environment Framework (Contract no. 507173, Priority 2.3.1.7 Semantic-based Knowledge Systems; more information at `http://www.vikef.net`).

References

1. P. Bouquet, L. Serafini, and H. Stoermer. Introducing Context into RDF Knowledge Bases. In *Proceedings of SWAP 2005, the 2nd Italian Semantic Web Workshop, Trento, Italy, December 14-16, 2005. CEUR Workshop Proceedings, ISSN 1613-0073, online http://ceur-ws.org/Vol-166/70.pdf*, 2005.
2. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named Graphs, Provenance and Trust. In *Proceedings of the Fourteenth International World Wide Web Conference (WWW2005), Chiba, Japan*, volume 14, pages 613–622, May 2005.
3. G. Criscuolo, F. Giunchiglia, and L. Serafini. A Foundation for Metareasoning, Part I: The proof theory. *Journal of Logic and Computation*, 12(1):167–208, 2002.

4. G. Criscuolo, F. Giunchiglia, and L. Serafini. A Foundation for Metareasoning, Part II: The model theory. *Journal of Logic and Computation*, 12(3):345–370, 2002.
5. F. Esposito, S. Ferilli, N. Di Mauro, T. M. A. Basile, L. Iannone, I. Palmisano, and G. Semeraro. Improving automatic labelling through rdf management. In Tengku M. T. Sembok, Halimah Badioze Zaman, Hsinchun Chen, Shalini R. Urs, and Sung-Hyon Myaeng, editors, *Digital Libraries: Technology and Management of Indigenous Knowledge for Global Access, 6th International Conference on Asian Digital Libraries, ICADL 2003, Kuala Lumpur, Malaysia, December 8-12, 2003, Proceedings*, volume 2911 of *Lecture Notes in Computer Science*, pages 578–589. Springer, 2003.
6. F. Esposito, L. Iannone, I. Palmisano, D. Redavid, and G. Semeraro. Redd: An algorithm for redundancy detection in rdf models. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, volume 3532 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 2005.
7. F. Esposito, L. Iannone, I. Palmisano, and G. Semeraro. RDF Core: a Component for Effective Management of RDF Models. In Isabel F. Cruz, Vipul Kashyap, Stefan Decker, and Rainer Eckstein, editors, *Proceedings of SWDB'03, The first International Workshop on Semantic Web and Databases, Co-located with VLDB 2003, Humboldt-Universität, Berlin, Germany, September 7-8, 2003*, 2003.
8. C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning=locality+compatibility. *Artif. Intell.*, 127(2):221–259, 2001.
9. R. V. Guha, R. McCool, and R. Fikes. Contexts for the semantic web. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2004.
10. G. Klyne. *Contexts for RDF Information Modelling.* Content Technologies Ltd, October 2000. http://www.ninebynine.org/RDFNotes/RDFContexts.html.
11. G. Semeraro, F. Esposito, S. Ferilli, T. M.A. Basile, N. Di Mauro, L. Iannone, and I. Palmisano. Automatic management of annotations on cultural heritage material. In *International Conference on Digital Libraries, ICDL 2004, New Delhi, India, February 24-27, 2004, Proceedings*, pages 805–812, 2004.
12. R.W. Weyhrauch. Prolegomena to a Theory of Mechanized Formal Reasoning. *Artificial Intelligence*, 13(1):133–176, 1980.