Introducing Context into RDF Knowledge Bases*

Paolo Bouquet[1], Luciano Serafini[2] and Heiko Stoermer[1]

[1] University of Trento,
Dept. of Information and Communication Tech.,
Via Sommarive 10, 38050 Trento, Italy
{bouquet,stoermer}@dit.unitn.it
[2] ITC-Irst
Via Sommarive 18, 38050 Trento, Italy
luciano.serafini@itc.it

Abstract. Large-scale knowledge representation (KR) with RDF unveils shortcomings which become obvious when facts have to be further qualified with contextual aspects to represent anything else but simplistic binary predicates. Motivated by requirements in the VIKEF project, in which we are required to store a large amount of information extracted from a heterogeneous set of documents and encoded in RDF triples, we are proposing an architecture for modelling context in RDF knowledge bases. Our approach – based on well-researched theories of context in KR – avoids issues that other approaches face, by preserving standard RDF within a context and adding context semantics and relations between contexts around the standard. In this paper we will present our approach, including formal definitions of our extensions and an illustration of further works.

1 Introduction

The motivating scenario for the work presented in this paper is the process of building and maintaining a large knowledge base (KB) about documents in distributed document repositories in the VIKEF project[3].

One of the main goals of this project is to store knowledge about documents, gathered by information and knowledge extraction processes, in an architecture including an RDF-triple repository and multiple ontologies.

A first solution would be to represent all the facts extracted from a pool of documents in a plain list of RDF triples. However, from a Knowledge Representation point of view RDF statements in general are context-free, and thus follow a notion of *universal truth*, while documents contain *context sensitive information* i.e., information whose interpretation depends on the context in which

* The work described in this paper has been partly funded by the European Commission through grant to the project VIKEF under the number IST-507173.

[3] For more information about the VIKEF architecture, features and application scenarios please refer to the project's website: http://www.vikef.net

the document is written. This way to proceed can easily generate contradictory statementes such that for instance "Silvio Berlusconi is the Prime minister of Italy" and "Romano Prodi is the Prime minister of Italy" as the result of articles written at different point in time.

It is obvious that in order to interpret the sentences about a document one should take into account a set of contextual (meta?) parameters about the document. Examples of such parameters are: the time, the place, the author, etc. These set of parameters allow to *contextualize* the content of the document.

However, this can not be accomplished easily, because the RDF data model does not allow for the further qualification of statements, they merely consist of a binary predicate of the form Subject+Predicate+Object [5].

The objective of this paper is to extend RDF semantics with contextual features in order to consistently cope with the following scenarios, which otherwise might lead to inconsistent RDF repositories.

During lifetime and evolution of an RDF knowledge base, from a semantic point of view, among others we foresee the following issues:

- Contradictory statements. In the course of time, we might add statements to that are in contradiction with former statements. Part of the reason for this is that RDF has been designed to be monotonic [8], so it is not possible to retract statements. So adding a statement to the knowledge base that contradicts with another statement, without the possibility of e.g. retracting one of the statements will lead to unwanted inconsistencies.
- Unknown relevance. If we have a set of statements involving an object (as subject or object of the statement), there is no way to determine if this statement is still relevant. Even if we had a rule about life-span of the statement, we could not apply the rule, because the statement is unqualified.
- Unknown validity. Aoccording to theory of context, there basically are no statements of universal truth [1, 9, 13, 12]. Currently, we cannot determine which statement is valid in a reasoning process and which is not, because in an RDF knowledge base we have no information based on which a statement's validity could be assessed.

In Sect. 2 we will provide an overview to standard RDF semantics as a starting point for our work. Section 3 explains our architecture in contrast to related work. Formal semantics of our extension to RDF are being provided in Sect. 4 and more details as well as topics for further works can be found in Sect. . Finally, Sect. provides a summary of the benefits of our approach and wraps up the paper.

2 Basic RDF Semantics

In order to illustrate the setting of our approach, let us use an example to explain the RDF semantics that are relevant in this case.[4]

[4] The authoritative source for RDF semantics is [8]. No special reference to parts of this document will be given here.

Imagine a domain of resources IR = {`Restaurant1, Restaurant2, Price1, Price2, Dish1, Dish2, Dish3, Menu1, Menu2`} and a set of properties IP = {`PriceOf, PartOf, ServedAt, MenuOf`}. Names for resources or properties in RDF can be i) plain literals, that stand as such[5], and the set of plain literals LV is part of the universe of interpretation UI, which is formed by the union $\{IR \cup IP \cup LV\}$, or ii) URI references, referring to an object defined elsewhere, e.g. in an ontology or an RDF schema. In our context it is noteworthy that in RDF a URI reference is taken to have the same meaning *wherever* and *whenever* it occurs. No special attention is paid to knowledge that might be encoded *within* a URI, such as special access protocols, the type of source of a document etc; RDF simply defines a mapping IS from URI references into UI. These facilities and our example universe of interpretation allow us to state the following facts:

1. `<Price1 PriceOf Dish1>`
2. `<Price2 PriceOf Dish2>`
3. `<PriceOf PartOf Menu1>`

These so-called triples denote simple truth values, and the interpretation of the triples follows model-theoretic standards: they are true if the objects of the domain denoted by the names (and referenced by a mapping, in the case of URI references) are in the respective relation. So in our example, the first statement denotes that there is an object in our domain that is referred to by the name of `Price1`, and it stands in a relation named `PriceOf` with the object referred to by the name of `Dish1`. If we find two objects in our world/universe for which the relation holds true, the statement becomes true. There are special objects in RDF which are called *blank nodes*. These are objects that exist, but do not have an identifier. A reader familiar with first-order logics can think of these blank nodes as existentially qualified variables, the interpretation of which follows standard semantics.

There is a specialty in RDF that allows us to make statements about properties, which makes sense from the point of view that following the above definition, a property – being an object of the universe – can also act as subject or object of an RDF triple. To distinguish between the way a property acts as – denoting a relation or acting as subject/object – RDF defines a special mapping $IEXT(p)$, with $p \in IP$ that maps from IP into the powerset of $IR \times IR$ containing the set of pairs $< x, y >$ with $x, y \in UI$ for which the property p holds true. This mapping is called the *extension* of p. In other words: if a property acts as subject or object of a statement, it can be seen as a placeholder for a set of all the object pairs of UI that are in the relation denoted by the property.

As illustrated in Fig. 1, number 3 of our example triples states that `<PriceOf PartOf Menu1>`, meaning that the pairs `<Price1, Dish1>` and `<Price2, Dish2>` as the extension of `PriceOf` form `Menu1`.

To illustrate a straightforward, but serious limitations with RDF, let us assume a set of statements like the following:

[5] It also knows *typed literals* – literals which have type information encoded in their string representation – which will not be specially treated in our case because this aspect is not affected by our work.

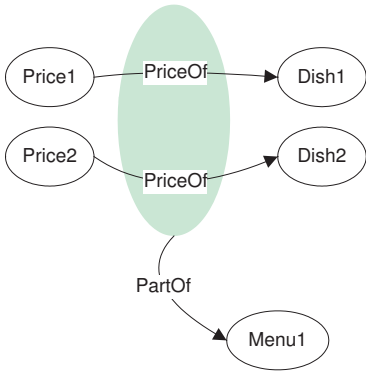

Fig. 1. Graphical view of an example IEXT mapping in RDF.

(1) `<Price1 PriceOf Dish1>`
(2) `<Price2 PriceOf Dish2>`
(3) `<Price1 PriceOf Dish3>`
(4) `<Dish1 ServedAt Restaurant1>`
(5) `<Dish2 ServedAt Restaurant1>`
(6) `<Dish1 ServedAt Restaurant2>`
(7) `<Dish3 ServedAt Restaurant2>`

If we think of evolutionary aspects of the example knowledge base, the shortcomings of plain RDF become obvious very quickly. How would we model the following simple fact: `Restaurant1` decides the change the price for `Dish1`, but `Restaurant2` does not? If we *add* a statement `<Price2 PriceOf Dish1>`, this will contradict with the existing statement (1). If we *change* statement (1), we will run into troubles because semantically in this case it applies to all restaurants that serve the dish, which is not what we wanted to express.

3 An Architecture for Contexts in RDF

We think that the mentioned issues can be attacked by introducing the notion of context into RDF, to limit the scope of an 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. Especially, we want to avoid introducing any kind of non-monotonicity into RDF, but rather present a mechanism to qualify statements 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.

A first naive approach of representing contexts would be to use the reification capability of RDF: for every statement inserted into an RDF graph we also insert

a number of meta-statements about this statement, containing all relevant context parameters, e.g.: `<1996 IsTheTimeOf "<RomanoProdi PrimeMinisterOf Italy>">` and `<2005 IsTheTimeOf "<SilvioBerlusconi PrimeMinisterOf Italy>">`.

There exist different opinions about the questions whether parameters describing a context can be limited or not [2]. In any case, this approach would be implementable using standard RDF, but not without drawbacks. With a potentially unlimited number of context parameters, we foresee a statement explosion with the this kind of approach, because for every statement we would have to add a significant number of meta-statements describing the relevant context dimensions, so the overhead is immense. A completely different approach of representing contexts in RDF is to extend RDF with the ability to represent a reference to a context directly in the data model. There have been proposals in the past, by Guha [7] or Klyne [11, 10], which do not use reification but implement context as a real extension of the RDF model theory, by moving from binary predicates to ternary predicates for identifying the context to which a statement belongs. To the best of our knowledge, these ideas have not been pursued any further. Moreover all the currently available RDF tools would have to be extended in order to deal with such an RDF model.

Our approach of representing context in RDF does not require changes to RDF and will prevent a statement explosion in the KB. We base our theory on the principles of *Locality* and *Compatibility* presented e.g. in [6], and influences from [3, 4]. What is relevant regarding RDF in this case is that a context can be thought of as a locally coherent set of axioms, and relations between contexts are modelled with the help of so-called *Compatibility Relations*.

4 Context Semantics

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 similar way as the described $IEXT$ mapping to enable contexts to appear as standard objects in RDF statements of other contexts.

As explained before, the $IEXT$ mapping of standard RDF has been introduced to provide a specialized mechanism to represent a set of object-pairs that are in a certain relation with a name that can itself be used in other statements. We will introduce a new mapping, called $ICONT$, to provide a way to map from an object of the universe of interpretation of one context into the set of statements that are *contained* in this object.

In the following, we provide formal definitions of the elements we introduce as an extension to plain RDF.

Definition 1 (of RDF Context) *An* RDF Context *c is a unique name in the form of an URI reference used to denote an RDF graph g_c.*

Definition 2 (of Contextualized UI) *The contextualized universe of interpretation of an RDF Context c, $CUI(c)$, is formed by the union $\{UI \cup IC\}$,*

where UI is the universe of interpretation defined in standard RDF and IC is a potentially empty set of contexts.

Definition 3 (of $ICONT$) *$ICONT(c)$ is a mapping from an RDF Context $c \in CUI_1$ to a Contextualized UI CUI_0: for all $c \in IC$ ICONT(c) is an RDF interpretation, with the restriction that if $I(c_i) = c$, then ICONT(c) is an RDF interpretation of the RDF graph g_c. (Note that CUI_1 and CUI_0 do not necessarily have to be disjoint).*

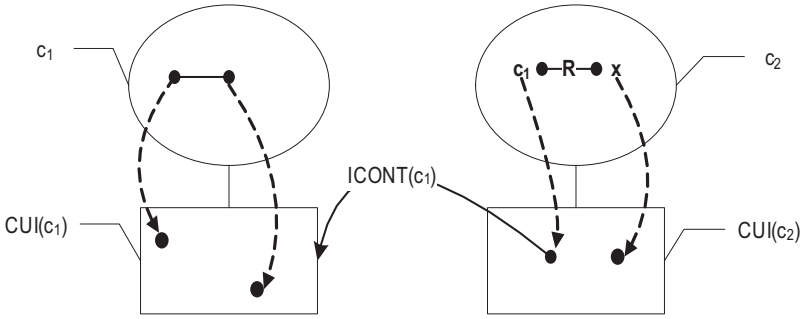

Fig. 2. Extended context semantics: a context object and it's interpretation.

In Fig. 2 we illustrate these extensions: on the left side, the ellipsis represents a context c_1, whereas the associated rectangle stands for its contextualized universe of interpretation of $CUI(c_1)$. The context c_2 on the right side contains an RDF statement <c_1 R x>, which is *about* context c_1 and could for example be a statement like <c_1 createdIn '2005'>. The local interpretation of c_1 in this statement is an *object* in the EUI of c_2. To dereference which model this statement is about, our new mapping $ICONT(c_1)$ maps to the CUI denoted by the name of c_1.

5 Compatibility Relations

So far we have illustrated an architecture of standard RDF, plus a way to model contexts including the respective semantics and interpretation function. However, what is still missing are the mentioned *compatibility relations* (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 $f \in c$). Please note that

this is currently work-in-progress. We are only starting to explore in full depth the aspects of compatibility relations that are relevant for our work.

Several approaches can be thought of to model compatibility relations 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 semantically cannot be fully formalized with the help of an ontology.

As an example for this claim take a relation like <c' `EXTENDS` c>. The semantics of this relation are envisioned to be like this: c and c' are taken to be compatible in the sense that one does not contain facts that contradict with facts of the other and that the relevant context parameters are the same; then, if no answer to a query to c can be given, the query will be propagated to c'.

Another example, which helps us to model the detection of identity between two objects of two different contexts, would be the relation <o `IN_CUI_OF` c'> with RDF Contexts c, c' and $o \in c$, to express that an object o is not only an element of its own CUI but also of a different one, which in effect means that it is *the same* object and thus there is an overlap between $CUI(c)$ and $CUI(c')$. A graphical view on this is given in Fig. 3.

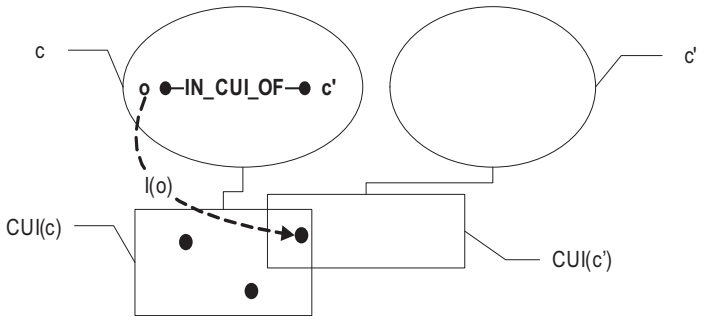

Fig. 3. Graphical illustration of the example compatibility relation IN_CUI_OF.

One of the questions that might arise is how these compatibility relations are supposed to be modelled. At the moment, we see three approaches to do this:

1. Formally define a fixed set of compatibility relations as part of our architecture and require all systems that implement the architecture to take care of providing also an implementation of the relations. In our opinion, this approach lacks flexibility, because we can think of a number of application-dependent CRs that can not be foreseen as part of this work.
2. Provide an ontology for context relations, so that there exists a vocabulary to describe these relations with the help of RDF. This approach is slightly more

flexible, because the ontology would be extendable, but we think that many interesting relations (e.g. the mentioned relation `EXTENDS`)are very hard to formalize and reason about e.g. with the help of Descrtiption Logics which underlies OWL.

3. Define a CR to be implemented as a *semantic attachment* [14], 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.

For the mentioned reasons we favor the third approach. Most of the future work regarding this architecture will involve identification of relevant CRs, and to provide their formal definitions to support implementers of the system.

6 Conclusion

We have illustrated the numerous issues that arise when trying to do serious KR with RDF. Our approach to overcome part of these issues is the introduction of the notion of context into RDF knowledge bases. To this end we presented an extension to standard RDF and provided formal definitions and semantics of this extension. An important aspect of our architecture are relations between contexts, and we have discussed several possible approaches how to model these.

The benefits of our approach are the following: a) RDF statements are no longer simplistic axioms of universal truth, but are restricted to a context which makes it possible to reason about aspects of the statement itself, e.g. validity, relevance, etc. b) Depending on the level of granularity (how many statements describe the context), we expect the overhead of representing the context to be mostly neglectable. In future work, this claim will have to be validated. c) The approach requires no changes to the core RDF data model. d) The context exists as an identifyable object in the knowledge base, so statements about this context can be made, e.g. for discovering context compatibility, context merging. Additionally, we expect whole contexts to be exchangeable between agents, which is an important property in the light of Distributed Information Systems such as for example multiple installations of VIKEF .

We are certainly aware that – although we are not modifying the core model of RDF – a considerable amount work is imposed on the implementer of such a system. However, in our opinion the elegance of the architecture is the fact that within a context, local reasoning will remain standard, and contexts and their relations are modelled around this standard.

References

[1] Y. Bar-Hillel. Indexical expressions. *Mind*, 63:359–379, 1954.

[2] Massimo Benerecetti, Paolo Bouquet, and Chiara Ghidini. Contextual reasoning distilled. *J. Exp. Theor. Artif. Intell.*, 12(3):279–305, 2000.

[3] G. Criscuolo, F. Giunchiglia, and L. Serafini. A Foundation for Metareasoning, Part I: The proof theory. *JLC*, 12(1):167–208, 2002.

[4] G. Criscuolo, F. Giunchiglia, and L. Serafini. A Foundation for Metareasoning, Part II: The model theory. *JLC*, 12(3):345–370, 2002.

[5] Eric Miller Frank Manola. *RDF Primer*, February 2004. http://www.w3.org/TR/rdf-primer/.

[6] Chiara Ghidini and Fausto Giunchiglia. Local models semantics, or contextual reasoning=locality+compatibility. *Artif. Intell.*, 127(2):221–259, 2001.

[7] Ramanathan V. Guha, Rob McCool, and Richard 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.

[8] Patrick Hayes. *RDF Semantics*, February 2004. http://www.w3.org/TR/rdf-mt/.

[9] D. Kaplan. On the logic of demonstratives. *Journal of Philosphical Logic*, 8:81–98, 1978.

[10] Graham Klyne. *Contexts for RDF Information Modelling*. Content Technologies Ltd, October 2000. http://www.ninebynine.org/RDFNotes/RDFContexts.html.

[11] Graham Klyne. *Circumstance, provenance and partial knowledge - Limiting the scope of RDF assertions*, 2002. http://www.ninebynine.org/RDFNotes/UsingContextsWithRDF.html.

[12] John L. McCarthy. Generality in artificial intelligence. *Commun. ACM*, 30(12):1029–1035, 1987.

[13] John L. McCarthy. Notes on formalizing context. In *IJCAI*, pages 555–562, 1993.

[14] R.W. Weyhrauch. Prolegomena to a Theory of Mechanized Formal Reasoning. *Artificial Intelligence*, 13(1):133–176, 1980.