A Query and Inference Service for RDF

Stefan Decker (University of Karlsruhe < Stefan.Decker@aifb.uni-karlsruhe.de>),
Dan Brickley (University of Bristol <daniel.brickley@bristol.ac.uk>),
Janne Saarela (World Wide Web Consortium <jsaarela@w3.org>),
Jürgen Angele (University of Karlsruhe <jan@aifb.uni-karlsruhe.de>)

Abstract

The creation of RDF raises the prospect of a widely accepted standard for representing expressive declarative knowledge on the Web. However, just representing knowledge and information is not enough: users as well as information agents have to query and use the data in several ways. An RDF Query specification would allow a range of applications to exploit any information source which can be represented in terms of the RDF data model.

In this paper we describe some existing techniques used in logic program evaluation and see how RDF descriptions fit into this framework.

Introduction

RDF provides a framework for representing machine-processable data on the Web. The RDF Model & Syntax specification [1] defines a formal set-theoretic data model which can have several isomorphic representations:

1. 3-tuples (triples)
2. acyclic directed labeled graph
3. XML transfer encoding

For example, the following examples show each of these representations for one and the same RDF data model.

```
{ language, http://foo.com/Welcome.html, en }
{ language, http://foo.com/Bienvenue.html, fr }
{ variant, http://foo.com/Welcome.html, http://foo.com/Bienvenue.html }
```

Figure 1. 3-tuple representation

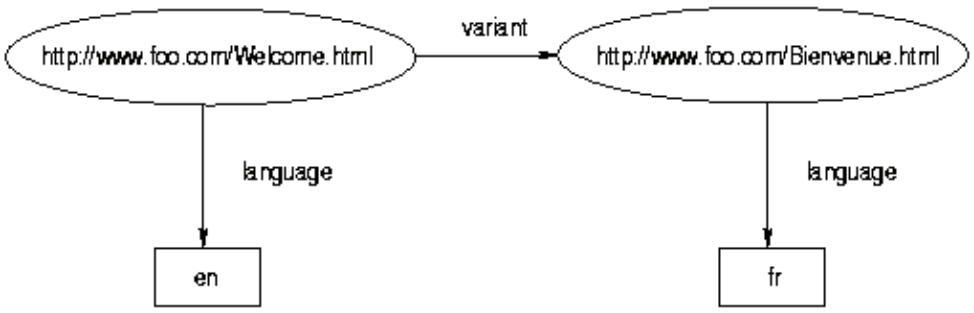

Figure 2. graph representation

```
<rdf:RDF xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
         xmlns:a="http://www.schema.org/usefulpredicates/">
 <rdf:Description about="http://www.foo.com/Welcome.html">
   <a:language>en</a:language>
   <a:variant rdf:resource="http://www.foo.com/Bienvenue.html" />
 </rdf:Description>

 <rdf:Description about="http://www.foo.com/Bienvenue.html">
   <a:language>fr</a:language>
 </rdf:Description>
</rdf:RDF>
```

Figure 3. XML transfer encoding

However, a knowledge representation format alone is not enough to enable a large community of potential users to process RDF effectively: standard query languages and, based on that, standard tools are needed to enable the creation of RDF-aware applications.

Since RDF is defined using an XML syntax, it might appear on the first sight, that a query language and system for XML would also be applicable to RDF. This is, however, not the case, since XML encodes the structure of data and documents whereas the RDF data model is more abstract. The relations or predicates of the RDF data model can be user defined and are not restricted to child/parent or attribute relations. RDF also provides for the merging of information from multiple data sources; a query language based on XML element hierarchies and attribute names will not easily cope with the aggregation of data from multiple RDF/XML files.

Also, the fact that RDF introduces several alternative ways to encode the same data model in XML for more effective authoring and to enable embedding of RDF inside HTML, means that syntax-oriented query languages will be unable to query RDF data effectively.

```
 <rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
      <rdf:Description about="http://www.w3.org/staffId/85740">
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org</v:Email>
      </rdf:Description>
    </s:Creator>
  </rdf:Description>
</rdf:RDF>


<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator rdf:resource="http://www.w3.org/staffId/85740"/>
  </rdf:Description>

  <rdf:Description about="http://www.w3.org/staffId/85740">
    <v:Name>Ora Lassila</v:Name>
    <v:Email>lassila@w3.org</v:Email>
  </rdf:Description>
</rdf:RDF>
```

Figure 4. Two XML encodings with the same datamodel

In the example shown in figure 4, a single XML query can not easily answer the question "What is the name of the creator of the resource http://www.w3.org/Home/Lassila?" for both encodings.

Requirements for an RDF Query Language and Inference System

Having motivated the need of an RDF query language and inference system, it is possible to identify several requirements for it:

- It should support the data model of RDF (resources, properties, values), which is very close in spirit to object oriented and frame based languages. This means a query language should support concepts familar from object-oriented modeling, such as class hierarchies and inheritance.

- In the context of distributed hetrogenous metadata represented in RDF and defined for a wide range of applications, it is also necessary to have access to the schema definitions for the vocabularies used in any given block of RDF data. These are defined using the RDF Schema Specification (RDFS) (see [4]), and provide information about relationships between classes and properties which might be used in queries. Thus an RDF query language should not only be able to query RDF data semantically, but also the schemas which are implicitly part of any RDF data model which uses them.

- While the RDF data model is adequate to represent the structure of queries, the RDF/XML serialisation syntax may not be the most appropriate format for representing these queries to human users. Consequently an RDF query language should be defined in terms of the abstract model, with the syntactic representation(s) a secondary concern. A syntax more oriented to conventional conventional SQL and Datalog styles may be easier to understand than an XML representation of the same structures.

- The RDFS specification includes features which require basic inferencing facilities in the storage/query system. For e.g. the rdfs:subClassOf and rdfs:subPropertyOf predicates are transitive - the RDF data model includes the notion of "implied" properties. These can not easily be expressed with standard relational or document-storage database technology, although the base facts in an RDF database can be stored in such systems.

  Furthermore, there are at least two other inference tasks, that are useful for a RDF specification together with a schema description:

1. RDF data can be checked for consistency against any 'contraint resources' which are used in the RDFS schemas referenced by that data. The RDFS core defines some simple constraint resources (range, domain) as well as an extensibility mechanism so that cardinality constraints and other richer constructs can be added later without confusing version 1.0 clients. Since these constraints might usefully be of arbitrary semantic complexity, simple (non-inference based) database techniques may not suffice.

2. Conversely, the schema itself can be used to derrive new information: e.g. if a predicate (property) "cooperatesWith" has a range constraint "Researcher" for a type "Researcher", and we know that "Dan Brickley" is a Researcher and he cooperates with "Janne Saarela" (but nothing more is known about "Janne Saarela"), we can conclude, that also Janne Saarela is also a Researcher. Usage of schemata in this way is especially useful in the Web environment, where information is usually incomplete. The application of schemata in such a way is discussed further in e.g [5] and [6].

RDF and Logic-based Languages

The triples ("statements") of the RDF data model introduced above can be thought of as the equivalent of ground facts in a logic based language. Consequently, logic based approaches to information management and querying map very simply onto the RDF model. Web applications require a fairly efficient implementation, so general theorem-prover systems are not likely to be appropriate. However, logic programming and deductive database techniques address all the requirements and issues sketched above.

The ability to specify complex integrity constraints requires the careful selection of an appropriate semantics (and so not every system is usable): because complex integrity constraints often require the need of negation, a semantics dealing with negation has to be used. However, stratified semantics are not usable: because the most used predicate used in an RDF translation is "triple" (see above), rules are often non-stratified. So a semantics able to deal with non-stratified negation has to be selected: an appropriate semantics is the Well-Founded-Semantics (see [7]).

Furthermore, an RDF inference engine should be easily usable in the Web environment and integratable with other software components, querying RDF is not a task on its one, but only a subtask for more advanced techniques on the web.

Since RDF syntax is not ideally suited as the basis for the concrete representation of the query language itself (the RDF-in-XML serialisation syntax is a little verbose for some purposes) it would be useful to explore whether these requirements for a query and constraint language can be met (at least partially) by an existing declarative language. This would also alow the reuse of existing inference engines.

As an example we examined Frame-Logic (F-logic), which accounts in a clean and declarative fashion for most of the structural aspects of object-oriented and frame-based languages and integrates higher-order concepts.

We illustrate here the similarities between RDF/RDFS and F-logic with some examples and show also some differences. *(note that in the following examples, namespace declaration are ommitted).*

```
The creator of the resource
http://www.w3.org/Home/Lassila" is Ora Lassila.

  <rdf:RDF>
    <rdf:Description about="http://www.w3.org/Home/Lassila">
      <s:Creator>Ora Lassila</s:Creator>
    </rdf:Description>
  </rdf:RDF>

Representation in F-logic

  "http://www.w3.org/Home/Lassila"[Creator->>"Ora Lassila"]
```

Figure 5. RDF and corresponding Frame-logic expression

Figure 5 shows one of the simplest possible RDF and F-logic expressions; these are directly equivalent: some described resource (object) is defined to have a predicate (attribute) with value "Ora Lassila".

RDF Schema statements can also be expressed in F-logic. The following example defines two classes (`Employee` as a subclass of `Person` and `Researcher` as subclass of `Employee`) and a property `cooperatesWith` where the range and domain of that property are both the class `Researcher`.

```
<rdf:RDF
```

```
<rdfs:Class rdfs:ID="Employee">
  <rdfs:subClassOf rdf:resource="http://ontology.org/human-ontology#Person"/>
</rdfs:Class>

<rdfs:Class rdfs:ID="Researcher">
  <rdfs:subClassOf rdf:resource="#Employee"/>
</rdfs:Class>

<rdf:Property ID="cooperatesWith">
  <rdfs:domain rdf:resource="#Researcher"/>
  <rdfs:range  rdf:resource="#Researcher"/>
</rdf:Property>
</rdf:RDF>

Representation in F-logic:

  Employee :: Person.
  Researcher :: Employee[cooperatesWith=>>Researcher].
```

Figure 6. RDFS and corresponding Frame-logic expression

This examples shows also one of the differences between RDFS and F-logic: as in most object oriented language, F-logic defines a class to have an attributes with a certain type. By contrast, RDFS defines a property as having a certain domain and range. This however, is not a difficulty in practice: it is still possible to use an attribute with all available objects, which was the main motivation for this design decision in RDFS.

More complex expressions are also possible with F-logic, as the following examples show.

```
  <rdf:RDF>
    <rdf:Description about="http://www.w3.org/Home/Lassila">
      <s:Creator>
        <rdf:Description about="http://www.w3.org/staffId/85740">
          <v:Name>Ora Lassila</v:Name>
          <v:Email>lassila@w3.org</v:Email>
        </rdf:Description>
      </s:Creator>
    </rdf:Description>
  </rdf:RDF>

Representation in F-logic:

"http://www.w3.org/Home/Lassila"[Creator->>
             quot;http://www.w3.org/staffId/85740"[
                        Name->>"Ora Lassila";
                        Email->>lassila@w3.org<
                        ]].
```

Figure 7. Complex RDF and corresponding Frame-logic expression

However, not all RDF expressions can be directly expressed in F-logic: examples include e.g. bags, sequences, alternatives. We are currently exploring syntactic and semantic expressions for representing these constructs within F-logic.

These basic expressions can be combined by introducing variables and boolean expressions to form arbitrary complex queries. An example is shown in figure 8. This query uses the ability of F-logic to quantify about all properties.

Representation in F-logic:

```
Give me all Resources and Employees of the W3C, such that Person
is a creator of the resource and this person is not directly
related to Nokia Research or is somehow related to Xerox-Parc.

FORALL Res,Pers
       <- Res[Creator->Pers:Employees[affiliation->>"http://www.w3.org"]]
              AND FORALL Prop,T
                Employee[Prop=>>T] AND
                ( NOT Pers[Prop->>"http://www.research.nokia.com"].
                  OR  Pers[Prop->>"http://www.parc.xerox.com"])
```

Figure 8. Complex Frame-logic query

Inference Engine Architecture

We implemented a prototypical inference engine, which is able to translate RDF and answer F-logic and predicate logic queries about them. As an implementation platform the Java language was chosen, because it is easy integratable with

other components available on the web - for example, a Java servlet 'RDF Query Server' could easily be created, and an HTTP-based API defined for query, update and result transactions.

The Inference Engine uses SiRPAC [8] to translate RDF specifications into triples, which can be queried as F-Logic expressions. However, the system is also able to handle formulas and data in a datalog like notation. An example for this is given in the next section.

The overall architecture of the system in depicted on figure 9.

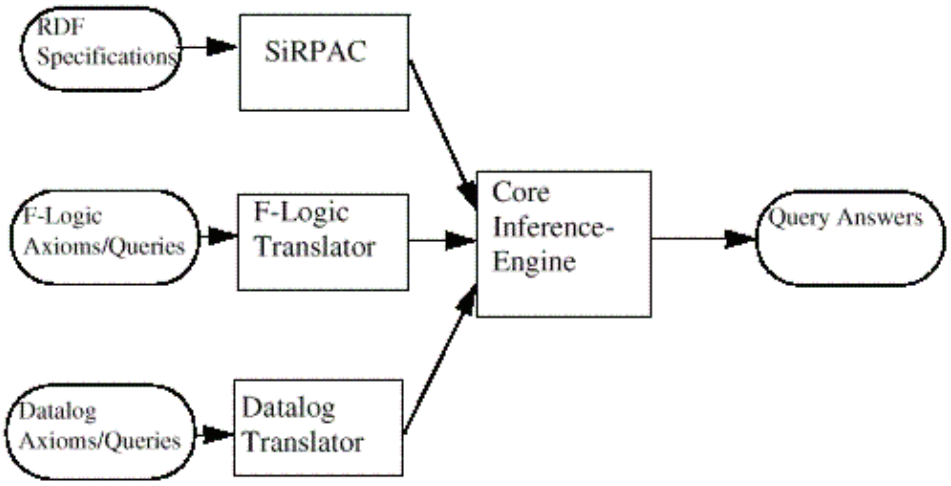

Figure 9. Architecture of RDF inference engine

Example: Classification Scheme Mapping

In this section we sketch some experiences with RDF and the inference engine. As a task the integration of data organised using differing classification schemes was chosen. The data used here has been provided by two "Subject-based Internet gateways"[9]. The problem addressed in this demo is that such gateways, while providing well organised repositories of Internet resource descriptions, invariably use different classification systems. In this case, we have a combination of data from Biz/ed (Business and Economics) and SOSIG (Social Science resources). Biz/ed uses a subset of the Dewey Decimal Classification, while SOSIG use a subset of the UDC scheme. The classification-scheme mapping data used here derrives from a study [10] undertaken within the DESIRE project.

This example explores the possibilities created by having a machine-understandable representation of both the classification data and of the relationships between different classification systems.

The raw data consists of 10,685 RDF descriptions (metadata annotations of resources), In our tests, about 46000 triples were generated by SiRPAC from these assertions; the number is larger because SiRPAC by default also generated reified representation of each statement it parser. These explicit reified statements could easily be removed from the database and replaced with logical rules implying their presence.

The following rules give a very simple example of the way in which the Inference Engine can be informed about the logical relationships between concepts.

Rules used

First, we give the system a relation 'about', and supply rules describing how it can infer that some web resource is 'about' some classification scheme concept. The purpose of this is to state that a resource is considered to be implicitly about some subject (for example 'Marketing') even if the base facts in the database do not implicitly assert that.

To facilitate this, a transitive relation "broader_term" is defined. The meaning of "broader_term(C1,C2)" is that the topic C2 is broader than the topic C1 and applies to more documents. The "about" relation enables us to look for documents about a certain topic, and builds upon the other defined relation. A query `FORALL Resource <- about(Resouce,"UDC:658.8")` returns a set of resources related by the property `about` to the classification "UDC:658.8". (*note that we use the abbreviation **UDC:** here instead of the full URI*)

```
FORALL O,V
 subject(O,V) <- O["http://www.desire.org/vocab/classmap#subject" ->> V].

FORALL  Concept1, Concept2, Concept3
```

```
  broader_term(Concept1,Concept3) <-
       broader_term(Concept1,Concept2) AND broader_term(Concept2,Concept3).


// "A resource is about a concept if a resource has a subject
// which is that concept..."

FORALL Resource, Concept
   about(Resource,Concept) <-  subject(Resource,Concept)
                       OR      // "...or is about a synonym of that concept, "
                       EXISTS X
                               (subject(Resource,X) AND synonym(X,Concept))
                             OR
                               (subject(Resource,X) AND synonym(Concept,X)).
                             OR  // "or is about a concept that is a broader term than that concept
                               (broader_term(Concept,X) AND subject(Resource,X)).
```

RDF Data used

The Inference Engine was loaded with data classifying Web resources from the two Internet catalogues. Some of these RDF statements use a subset of the UDC classification scheme; others used a subset of the Dewey Decimal scheme. For the purpose of this demonstration, concepts from these vocabularies were assigned URIs in two RDF/XML namespaces: http://purl.org/net/rdf/UDCsubset/ and http://purl.org/net/rdf/DDCsubset/

An RDF representation of the classification data was created. In addition to this, statements were also stored about the properties of the classification scheme items. For example, asserting that one classification concept was a 'broader term' or 'narrower term' or 'synonym' for another. Human readable labels were also attached to the categories in this manner. Both F-Logic and RDF representations were used.

```
<rdf:Description
     about="ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/sci/environment/">
        <s:subject resource="http://purl.org/net/rdf/udcsubset/c504"/>
</rdf:Description>

<rdf:Description
     about="ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/sci/psychology/misc/">
       <s:subject resource="http://purl.org/net/rdf/udcsubset/c159.9"/>
</rdf:Description>

<!-- several thousand such descriptions were used in the testbed -->

Data about the relationships between classification concepts is also
supplied. Here shown in triple syntax:

synonym("http://purl.org/net/rdf/ddcsubset/c658.8","http://purl.org/net/rdf/udcsubset/c658.8").
...
broader_term("http://purl.org/net/rdf/ddcsubset/c338.7","http://purl.org/net/rdf/udcsubset/c65").
broader_term("http://purl.org/net/rdf/ddcsubset/c657.4","http://purl.org/net/rdf/udcsubset/c657").
```

Results

To summarise: the inference engine was loaded with the following:

- several thousand classifications in UDC vocabulary from the SOSIG database (in RDF)
- several thousand classifications in Dewey (DDC) vocabulary from the Biz/ed database (in RDF)
- statements about the hierachical relationships and human labels for these concepts
- rules providing for simple inferencing

This was sufficient to allow queries against the RDF data using full F-Logic facilities.

Example Query

The example discussed here shows that, once RDF data is available, it can be easily used for inference processes.

The simplest possible query which proves this point is shown here:

```
Query:

FORALL Classification
   <- about ("http://www.stir.ac.uk/marketing/",Classification).


Results:

Classification = "http://purl.org/net/rdf/DDCsubset/c378"
```

```
Classification = "http://purl.org/net/rdf/DDCsubset/c658.8"
Classification = "http://purl.org/net/rdf/UDCsubset/c658.8"
```

This shows all the values of 'Classification' for which the database can find a match, stored or implied, against the RDF data about the resource specified. If we look at the actual data loaded into the engine, we see that only *two* classifications are explicitly stored:

```
subject("http://www.stir.ac.uk/marketing/","http://purl.org/net/rdf/DDCsubset/c378")
subject("http://www.stir.ac.uk/marketing/","http://purl.org/net/rdf/DDCsubset/c658")
title("http://www.stir.ac.uk/marketing/","University of Stirling, Department of Marketing").
```

These facts, when combined with the declarative rule telling us that "about"ness can be implied through synonym-relations between classifications, is enough to to allow the database to conclude that since DDC:658 and UDC:658 are synonyms, the latter classification also fairly describes the resource.

Mechanisms such as these suggest ways in which searchable and browsable interfaces to Web content might be constructed which manage to hide the different organising schemes in use. This might be used, for example, to allow a user to express a query in one vocabulary and have that search successfully find relevant resources classified using a different vocabulary scheme. More generally, this example demonstrates how established logic-based mechanisms can be combined with the use of RDF to provide new approaches to information management well-suited to the hetrogenous Web environment.

Discussion and future work

We have shown a query and inference service for RDF using techniques from deductive databases and logic programming. The approach sketched above does of course not only work for our inference system, but maps well to the other deductive systems that are available. However, our engine has some advantages over the others: It is written in Java, offers reasonable performance and is relatively small in size. As such it is available on all major platforms, and, even more importantly, easily integratable with other java software. It was considered particularly important to have a lightweight and portable implementation: the engine is targeted at applications such as intelligent agents and mobile code, so that it can provide a standard component for software agents in areas such as distributed search and electronic trade.

The inference engine is available under a GNU public license and can be downloaded from http://www.aifb.uni-karlsruhe.de/~sde/rdf. By demonstrating running code and motivating examples for an RDF Query service, we hope to provide concrete evidence and a practical testbed for exploring the design trade-offs involved in defining a Query standard for RDF.

References

1. Lassila, O., Swick, R. (1998). The Resource Description Framework (RDF) Model & Syntax. W3C Working Draft. *http://www.w3.org/TR/WD-rdf-syntax*.
2. Detsch, A., Fernandez, M., Florescu, D., Levy, A., Suciu, D.(1998). XML-QL: A Query Language for XML. *http://www.w3.org/TR/1998/NOTE-xml-ql-19980819*.
3. Kifer M., Lausen G., Wu J.(1995). Logical Foundations of Object-Oriented and Frame-Based Languages, Journal of the ACM , vol. 42, p. 741-843.
4. Brickley D., Guha R.V., Layman A. (1998). Resource Description Framework Schema Specification *http://www.w3.org/TR/WD-rdf-schema/*.
5. Decker S.,Erdmann, M., Fensel, D., and Studer, R. (1999) Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.), Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston. *http://www.aifb.uni-karlsruhe.de/WBS/broker*.
6. Crampe I., Euzenat, J.: Object Knowledge Base Revision (1998). In: Prade, H. (Ed.) Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98), John Wiley & Sons.
7. Van Gelder, A., Ross K., Schlipf, J.S.(1991): The Well-Founded Semantics for General Logic Programs, Journal of the ACM, 38(3):620-650.
8. Saarela, J: (1998) SiRPAC - Simple RDF Parser & Compiler, *http://www.w3.org/RDF/Implementations/SiRPAC/*.
9. Subject Based Information Gateways, DESIRE Resource Discovery reports, *http://www.desire.org/results/discovery/*
10. Mapping Classification Schemes, D.Hiom, DESIRE report, *http://www.desire.org/results/discovery/cat/mapclass_des.htm*