Qualifying Triples and Selective Reasoning

Stuart Hendren[1] and John Yesberg[2]

[1] Defence Science and Technology Laboratory
Porton Down, Salisbury, Wiltshire, SP4 0JQ, UK
shendren@dstl.gov.uk
[2] Defence Science and Technology Organisation, C3I Division*
PO Box 1500, Edinburgh, South Australia 5111 Australia
jyesberg@dsto.defence.gov.au

Abstract. We address the need to add metadata, such as uncertainty and source, to statements in OWL ontologies and consider the implications of trusting different sources on inferences in the knowledgebase. Traditional approaches lead to undecidable models. We describe two techniques that do allow us to reason about models that include statements and meta-statements.

1 Introduction

We wish to draw conclusions about the world based on observations that may be unreliable, erroneous and conflicting. This requires reasoning not only about the observed data but also about its reliability, taking into account its source, timing and any other provenance information. OWL does not provide a satisfactory means for both expressing and reasoning about such data.

To provide a simple example, if I receive the information "Alice likes Bob" from Carol. However, I then get the information "Alice dislikes Bob" from Dave. These two triples are arguably contradictory due to the relations being (at least in principle) disjoint. A natural way of dealing with this type of situation is to consider whether we trust one source more than the other. To replicate this within OWL it is insufficient to simply record the data "Alice likes Bob" and "Alice dislikes Bob". Instead we need "Carol said Alice likes Bob" and "Dave said Alice dislikes Bob" along with trustworthiness information. This trustworthiness information may be the result of reasoning over past (or subsequent) statements. Other aspects, such as who else trusts the sources, may influence the conclusion.

We aim to explore the use of systems such as Bayesian belief networks, transferable belief models [1], and subjective logic [2] in order to give qualifications (such as "how much support is there for this conclusion") to the results of various queries. We also wish to be able to conduct temporal analysis [3], that might indicate who knew what and when.

* John Yesberg is on secondment to the Defence Science and Technology Laboratory's Information Management Department in the UK. This document does not represent the views of either Dstl or DSTO.

The reasoning on meta data is undoubtedly of interest. Products such as Sentinel Visualizer [4, 5] and Analysts Notebook [6] allow the filtering of data based on metadata to perform what-if queries.

To be able to reason and gain added value from the collected information, we wish to restrict ourselves to OWL DL. However, the standard method to make assertions about triples, RDF reification, causes the model to be OWL Full and therefore undecidable.

In this paper we consider two different approaches to this modelling problem. The first steps outside the normal techniques by using "quads" (labelled triples) and segregating parts of the model into separate OWL DL ontologies. The second tries to stay within OWL DL and duplicate the functionality of the quad approach by using a consistent method to reifying relations in OWL DL across all relations in the ontology.

2 The General Problem

Within the RDF and OWL knowledge representation framework an assertion is a triple of the form `:subject :predicate :object` (written in the Turtle notation [7]) where each part is identified by a uniform resource identifier (URI). Both the definition of the ontology and the data that populates the ontology to form the knowledgebase are in this `:s :p :o` form.

We are interested in the case that the predicate is in the modelled domain (an object or data-type property) and not part of the OWL or RDF specifications, and we shall refer to such triples as statements. In this situation the predicate carries some inbuilt semantics; if it is a data-type property the subject must be an instance of some class in the domain (or possibly just `owl:Thing`) and the object is a value from some data type. If the predicate is an object property then both the subject and object are instances of `owl:Thing`.

As shown above, there is often the need to express further information about the statement, such as the source, the time stated and time applicable. This causes problems in OWL DL. We wish to record in our knowledgebase that Alice likes Bob and that we obtained this information is from Carol. Using the predicates `likes` and `saidBy` this would naturally have the form.

```
:Alice :likes  :Bob .
:(:Alice :likes :Bob) :saidBy :Carol
```

However, a statement in OWL cannot be given a URI, so it is impossible to assert this directly in the language. This situation is usually encoded using RDF reification in the following way:

```
:Alice      :likes        :Bob .
:statement1 rdf:type      rdf:Statement ;
            rdf:subject   :Alice ;
            rdf:predicate :likes ;
            rdf:object    :Bob ;
            rdf:saidBy    :Carol .
```

Using this technique in OWL makes the knowledge base OWL Full, and hence undecidable [8]. We wish to avoid this situation, as we want to reason both about what was stated and the sources of the statements. In the following section we explore a practical solution that goes outside the OWL specification based on "quads": triples that *do* have a URI to identify them. We implement some software to provide the required functionality. We then show how this can be imitated within OWL DL using reification (in the same way as for n-ary relations) and rules.

3 Quads solution technique

In this first approach, we define two separate ontologies: a world ontology and a meta ontology. We then assert world statements in the normal way, except that we may optionally label triples with identifiers, thus making them "quads", of the form `(id, s, p, o)`. To make a meta statement about a world statement, we can use the identifier `id` of the world statement as the subject or object of the meta statement.

We have built a "Quad Processor" over Jena [9] that reads these two ontologies and a set of quads. (Statements with no label are treated as quads with an empty identifier.) Using some simple string comparisons, the quad processor can determine which triples should be added to the meta ontology to make the meta model[3]. We can then reason about and query (using "Reasoner 1" in figure 1) the meta model to select (by their identifiers) world statements of interest. The quad processor can then assert these specific world statements with the world ontology to create a world model, over which inference and reasoning can be performed (using "Reasoner 2" which could, but need not, be separate from Reasoner 1).

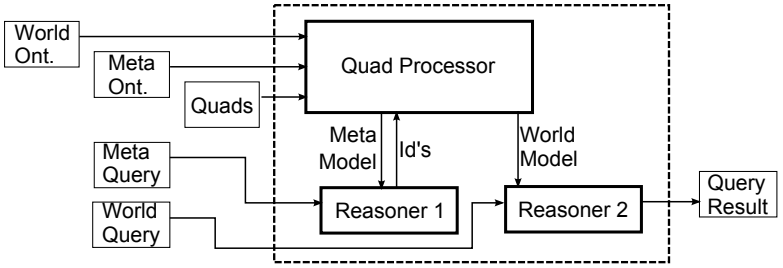

Fig. 1. Data Flow in the Quad Solution

This is illustrated by the following example:

[3] A statement `(id, s, p, o)` will lead to a statement `(s,p,o)` in the world model if s, p, and o are objects from the world ontology. If s, p, and o are objects in the meta ontology (which would be the case if `s` or `o` are statement identifiers), then this quad would lead to a statement in the meta model.

```
:Alice :likes  :Bob           :id1 .
:id1   :saidBy :Carol         :id2 .
:Alice a       :Person        :id3 .
:Carol a       :TrustedSource :id4 .
:Alice :likes  :Ed            :id5 .
:Ed    a       :Person        :id6 .
:id5   :saidBy :Dave .
:Dave  a       :UntrustedSource .
```

We may wish to know everyone whom Alice likes. This would be easy in standard ontologies. But we may want to discover whom we are *sure* Alice likes. This could be discovered by asking the meta model to exclude all statements from non-TrustedSources, and then creating a new model from that limited set, and making inferences on the limited set.

The (OWL DL) meta model would be as follows.

```
:id1   :saidBy :Carol .
:Carol a       :TrustedSource .
:id5   :saidBy :Dave .
:Dave  a       :UntrustedSource .
```

A simple SPARQL query will reveal that the only statement made by a trusted source is id1.

```
SELECT ?id
WHERE { ?id  meta:saidBy ?o .
        ?o   a  meta:TrustedSource . }
```

Alternatively, we can ask the Quad Processor to create the world model from all of the quads *except* for those that are the result of a query. In that case, we can ask for it to exclude all statements made by untrusted sources. The advantage is that this can include statements that may not have any meta data, such as `:Alice a :Person`. Thus the following query would yield the statements listed.

```
:Alice :likes :Bob .
:Alice a      :Person .
:Ed    a      :Person .
```

3.1 An proposed extension to Turtle syntax

A fragment of the existing Turtle grammar EBNF specification [7] is:

```
[8]  objectList ::=  object ( ',' object)*
[13] object     ::=  resource | blank | literal
```

Since every `object` leads to a new triple, we wish to allow the optional addition of an identifier to every object. We propose to modify this as follows:

```
[8]  objectList      ::= identifiableObj ( ',' identifiableObj)*
[8a] identifiableObj ::= object identifier | object
[8b] identifier      ::= resource
```

No additional punctuation is needed, as the existing object is clearly delineated. There can be no ambiguity about which part is an object, and which part is an identifier. We are in the process of building a parser that will interpret this extended syntax.

3.2 Discussion of the Quads technique

We have not explored the limits of this technique. It does successfully solve the problem that we identified: we have used an OWL DL reasoner to reason about both statements and meta statements. The Quads technique relies on information (identifiers) which are not expressible in the current OWL syntax, but we have proposed an extension that will allow this.

One possible question relates to meta-meta-statements; statements that describe meta-statements. The proposed syntax would be sufficient to allow the expression of such statements. However the Quad Processor software that we developed could not (in its present state) handle such statements. We believe that as long as separate between the levels is maintained the models will be OWL DL. Extensions to the Quad Processor to allow this may be feasible.

The Quad Processor currently uses very simple string analysis of namespaces to decide whether something is a world statement or a meta statement. While suitable for a concept demonstration, this could be extended so as not to create such restrictions on the namespaces used.

We have demonstrated the concept using very simple non-overlapping world and meta ontologies. It may be that both of these should import, say, the OWL-Time ontology [3]. The impacts of such an overlap or potentially more complex ontology relationships have not been explored.

4 Reification solution technique

In this approach we create a single OWL DL model and use rules to obtain the required functionality. We use the reification technique, usually applied to n-ary relations, to separate the *world ontology*, where instances represent objects in that world, and the *meta ontology* where instances represent the sources and related concepts. The ontologies of the two models can be created in OWL DL in a standard way. However we duplicate the relations in the world model in the following way.

```
world:likes       a                 owl:ObjectProperty .
meta:Statement    a                 owl:Class .
meta:relationName a                 owl:AnnotationProperty .
meta:world_likes  rdfs:subClassOf   meta:Statement ;
                  meta:relationName world:likes .
```

The standard `:Alice world:Likes :Bob` relation becomes figure 2 with the added meta information.

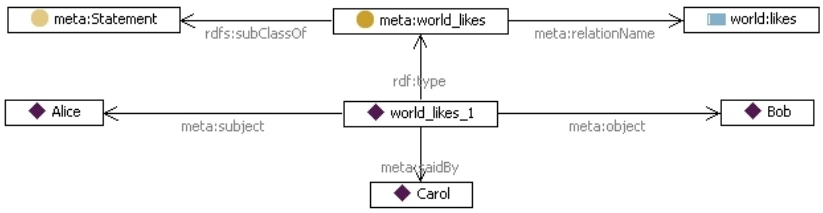

Fig. 2. OWL reified relation with meta information

We can build the `Statement` subclasses using SPARQL `CONSTRUCT` queries; each relation gets a class under the `Statement` and is annotated with the original relation. This remains OWL DL as the annotation property `relationName` is ignored by the reasoner but can be used in rules and queries. The resulting OWL DL ontology is populated with the data and metadata using only the reified relations.

We reason in the standard way over the resulting knowledgebase to make inferences in the meta part of the model. Then we can query the knowledgebase to give a selection of the statements from the world model. We use the class `SelectedStatement` to hold the results of the query as in the following example.

```
CONSTRUCT { ?id a meta:SelectedStatement }
WHERE { ?id     meta:saidBy ?source .
        ?source a           meta:TrustedSource . }
```

The rule below asserts the instances of `meta:SelectedStatement` into the world model and we can now reason and query it in the standard way. This allows us to do selective reasoning in the world model to try and understand the contributions to the knowledge base from different sources, how they interact and test the validity of hypotheses.

```
CONSTRUCT { ?subject ?predicate ?object }
WHERE { ?statement a meta:SelectedStatement ;
                   a ?relation ;
                   meta:subject ?subject ;
                   meta:object ?object .
        ?relation  meta:relationName ?name .
        OPTIONAL{ ?predicate a owl:ObjectProperty } .
        OPTIONAL{ ?predicate a owl:DatatypeProperty } .
        FILTER( ?p = ?name ) }
```

4.1 Discussion of the reification approach

This technique appears to mirror the first approach, but does so entirely in OWL, within a tool such as TopBraid Composer. Rather than requiring an extension to the OWL syntax, data is provided in an already reified format. Then, instead of having a Quad Processor construct the world model from statements identified by a query, we use SPARQL `CONSTRUCT` queries to create the appropriate triples.

While this approach works for the simple examples that we have explored, we have not proven whether it is quite as powerful an approach as that of labelling triples.

It is possible to add restrictions to subject and object to regain the functional, inverse functional and symmetric relation properties. For example (see figure 3 and 4) to declare a relation functional we create a subproperty of subject that is inverse functional and a subproperty of object that is functional.

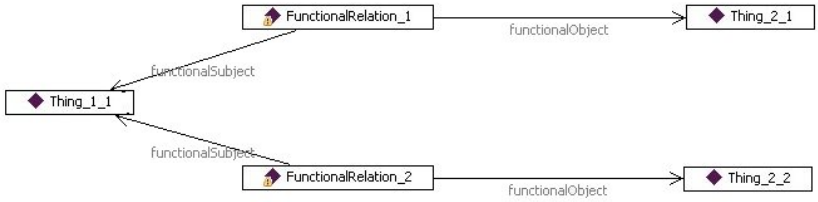

Fig. 3. Functional relation

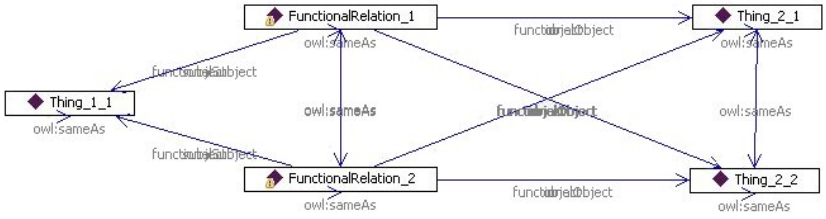

Fig. 4. Functional relation after inference

It would also be possible to add the range and domain restrictions using `owl:allValuesFrom` restrictions on `subject` and `object`. However some properties of the usual relation can not be replicated in OWL DL, for example transitivity would require the addition of a rule. Similarly, it would be valid to build

the full relation hierarchy using `rdfs:subClassOf` to gain propagation of relations up the chain. However, the inheritance property of sub classes is different to that of relations. If I create a symmetric relation in the above way then any relation lower in the hierarchy would inherit the symmetric property but this is not generally true. Consider the symmetric relation `spouseOf` and its non-symmetric subrelations `wifeOf` and `husbandOf`.

This problem could largely be solved by creating different subject and object properties for each reified relation. In situations where the world model is simple this approach could do away with the original relationship hierarchy. For larger ontologies using a rule to go back to the original hierarchy provides a simpler solution and facilitates the selective reasoning over the world ontology.

Finally, we note that in our solution, the data is initially prepared in a rather clumsy model, as there is currently no way in OWL to handle n-ary relations elegantly. It would, however, be possible to construct a layer of software that would hide the clumsiness from the user.

5 Discussion

We have already proposed a syntax for extending OWL to add identifiers to statements. We believe that assigning a URI to statements is reasonable in principle. We have not thoroughly investigated the implications of such a change, and so there may be problems we have not identified.

We note that Motik [10] has also been working in the area to add meta-modeling to OWL DL, without making the resulting model undecidable. The problem domain in that (referring to a class as an instance) is somewhat different from ours (referring to a triple as an instance). We have not assessed how Motik's proposed extension might apply to our problem.

Others are looking at the selection of minimal subsets as justifications for certain conclusions [11]. This technique could be used to examine contradictions in the knowledgebase and generate hypotheses on trustworthiness that could be tested with the selective reasoning techniques described.

Named Graphs [12,13] is a another technique that has been used for creating subsets of triples. In this case, instead of labelling individual triples, a name is given to sets of triples. The ability to identify a set of triples is strictly more expressive than the ability to identify individual triples. However, we are not convinced that the graph naming as described provides the ability to reason about the meta information in the same way as we have proposed.

We looked at classifying statements as being either `UntrustedStatements` or `TrustedStatements`, according to whether they were `saidBy` a `TrustedSource` or a `UntrustedSource`. This might have been expressed in the ontology by defining that the object property `saidBy` is the union of two disjoint object properties, `TrustedSaidBy` and `UntrustedSaidBy` as proposed below. By asserting that the range of a `TrustedSaidBy` is a `TrustedSource`, and the range is a `TrustedStatement` (and similarly for Untrusted). A reasoner could then deduce that any statement `saidBy` someone was a `TrustedStatement` (or an

UntrustedStatement), according to the person who made it. While OWL 2 allows disjoint object properties, and the union of classes, it doesn't (yet) allow the union of object properties. Adding such a feature should be considered for future versions.

```
:saidBy a                  owl:ObjectProperty ;
        rdfs:domain        :Statement ;
        rdfs:range         :Source ;
        owl3:disjointUnion (:trustedSaidBy :untrustedSaidBy) .

:trustedSaidby a                  owl:ObjectProperty ;
               rdfs:domain        :TrustedStatement ;
               rdfs:range         :TrustedSource ;
               rdfs:subPropertyOf :saidBy .

:untrustedSaidby a                            owl:ObjectProperty ;
                 rdfs:domain                  :UntrustedStatement ;
                 rdfs:range                   :UntrustedSource ;
                 rdfs:subPropertyOf           :saidBy ;
                 owl2:disjointObjectProperty :trustedSaidBy .

:stmt1 :saidBy :Carol .

:Carol a :TrustedSource .

ASK WHERE { :stmt1 a :TrustedStatement }
```

6 Conclusion

With standard modelling techniques, the introduction of meta statements takes us outside the OWL DL domain, thus preventing the use of powerful reasoning engines. We have described two techniques which extend the scope of models that we can reason about. These techniques have both been experimentally tested using Jena and TopBraid.

We have discussed a number of possible extensions to OWL: labelling of triples required for "quads" approach, support for n-ary relations required for elegant reification, and (disjoint) union of properties for improved reasoning.

It remains to fully characterise and compare the two approaches.

References

1. Smets, P., Kennes, R.: The transferable belief model. Artificial intelligence **66** (1994) 191–234 http://iridia.ulb.ac.be/ psmets/TBM-AIJ.pdf.
2. Jøsang, A.: Artificial reasoning with subjective logic. In: Proceedings of the Second Australian Workshop on Commonsense Reasoning. (1997) http://sky.fit.qut.edu.au/ josang/papers/Jos1997-AWCR.pdf.

3. Hobbs, J.R., Pan, F.: Time ontology in OWL
http://www.w3.org/TR/2006/WD-owl-time-20060927.
4. FMS Advanced Systems Group: Sentinel visualizer
http://www.fmsasg.com/Products/SentinelVisualizer/index.asp.
5. Haught, D.: Metadata means something
http://danoviz.wordpress.com/2008/03/26/metadata-means-something/.
6. i2: Analyst's notebook 7
http://www.i2.co.uk/products/analysts_notebook/default.asp.
7. Beckett, D.: Turtle http://www.dajobe.org/2004/01/turtle/.
8. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In Ganzinger, H., McAllester, D., Voronkov, A., eds.: Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99). Number 1705 in Lecture Notes in Artificial Intelligence, Springer (1999) 161–180
9. Jena open source community, including HP Labs, Bristol: Jena home page
http://jena.sourceforge.net.
10. Motik, B.: On the properties of metamodeling in OWL. Journal of Logic and Computation **17**(4) (2007) 617–637
http://www.comlab.ox.ac.uk/people/boris.motik/pubs/motik07metamodeling-journal.pdf.
11. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: International Semantic Web Conference. (2008)
http://www.cs.man.ac.uk/ horridgm/papers/2008/iswc/HoPaSa08.pdf.
12. Semantic Web Interest Group: Named graphs http://www.w3.org/2004/03/trix.
13. Carol, J.J., Stickler, P.: RDF triples in XML. Technical Report HPL-2003-268, HP Laboratories, Bristol (2003)
http://www.hpl.hp.com/techreports/2003/HPL-2003-268.html.