Living with topic maps and RDF

Topic maps, RDF, DAML, OIL, OWL, TMCL

By: Lars Marius Garshol
Affiliation: Ontopia
Email: larsga@ontopia.net
Web: http://www.ontopia.net

Table of contents

Abstract

This paper is about the relationship between the topic map and RDF standards families. It compares the two technologies and looks at ways to make it easier for users to live in a world were both technologies are used. This is done by looking at how to convert information back and forth between the two technologies, how to convert schema information, and how to do queries across both information representations. Ways to achieve all of these goals are presented.

This paper extends and improves on earlier work on the same subject, described in [Garshol01b].

Biography

Lars Marius Garshol is currently Development Manager at Ontopia, a leading topic map software vendor. He has been active in the XML and topic map communities as a speaker, consultant and open source developer for a number of years.

Lars Marius has also been responsible for adding Unicode support to the Opera web browser. His book on *Definitive XML Application Development* was published by Prentice-Hall in its Charles Goldfarb series last year.

Lars Marius is one of the editors of the ISO Topic Map Query Language standard, and also co-author of the Topic Map standard itself.

1. Introduction

> Essentia non sunt multiplicanda praeter necessitatem. (William of Ockham.)

For someone looking into topic maps and RDF today the similarities between them are obvious, and it may appear absurd that users should be forced to choose between two technologies that to them must seem almost indistinguishable. However, as this paper tries to show, there are a number of reasons why this is so.

The first and most obvious explanation for this situation is the historical. Topic maps started in the early 90s from work on managing indexes to documentation and was worked on for several years before being adopted as an ISO work item in 1996. That work resulted in the publication of the topic map standard as ISO 13250 in early 2000, and later of XTM 1.0 in early 2001[Pepper99].

RDF, on the other hand, came out of work done by R.V. Guha at Apple on MCF (the Meta Content Framework), which was later turned into an XML application and taken up by the W3C. The W3C turned it into what we today know as RDF, which was published as a W3C Recommendation in early 1999.

So how does this explain anything? Well, it wasn't before early 1999 that the two communities started to become aware of one another, and by that time it was already too late. The RDF Recommendation was already published, and the topic maps standard was submitted to ISO not long after. Had they discovered one another earlier the two efforts might have merged, but as it was that meant retracting already published specifications and rethinking concepts that were already felt to be well understood.

This is where things stand today: we have two technical communities, each with its own family of standards, and each well entrenched. A merger appears politically impossible, and, as will be seen, there are also technical arguments against it. Thus, the focus of this paper is to compare the models of the two technologies, and to use this comparison to describe ways in which the two technologies can be made to work together. In other words, the goal of this paper is to make it easier for users to live with both RDF and topic maps.

1.1. Differences in outlook

While the technologies are clearly similar it is equally clear that they are intended for different purposes. Topic maps were created to support high-level indexing of sets of information resources to make the information in them findable. RDF, on the other hand, was intended to support the vision of the semantic web through providing structured metadata about resources and a foundation for logical inferencing.

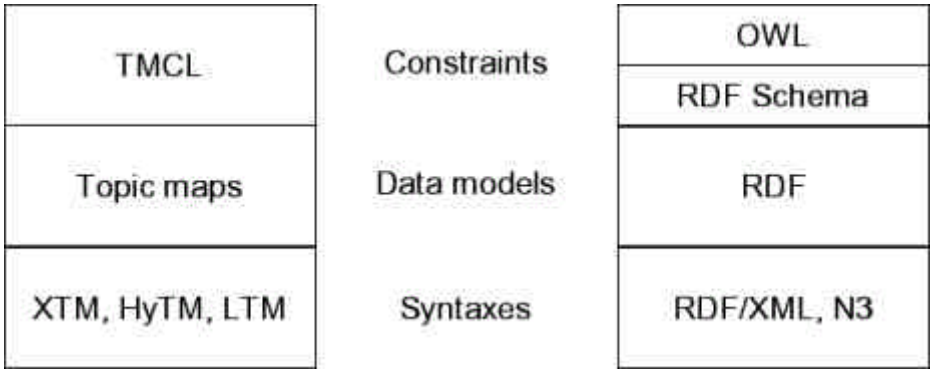
While reading the rest of this paper it is worth bearing in mind these differences, as they do explain some of the differences between these two technologies. The differences in outlook also make it harder to merge these two into a single technology, since the communities do not in fact have the same goals.

The differences also provide a rationale for the continuing separation between the two: different tools for different purposes. It is hoped that the rest of the paper will help make it clearer what each technology is best suited for, although this is not the purpose of the paper.

1.2. Understanding the standards families

As anyone who looks into topic maps and/or RDF will find, there are a number of acronyms floating around, and it may often be difficult to work out exactly how noodle X relates to noodle Y in this alphabet soup. Looking at the diagram below may help, however.

The two standards families



This splits the two families into three areas: syntaxes, data models, and constraints. Both RDF and topic maps have a number of interchange syntaxes, some standardized, and some not. In this paper we will ignore the syntaxes entirely, as interoperability on this level is not really very useful, and instead focus on the higher levels.

What one means, or should mean, when saying "topic maps" is the abstract model of topic maps, which consists of topics, associations, occurrences, and so on. Similarly with RDF, which consists of nodes and statements. This is the level at which one needs to compare the two to understand the relationship between them, and so this is what the next section will do.

However, a lot of the buzz around these technologies, particularly in the case of RDF, is around "ontologies", and keywords often mentioned in this context are OWL, DAML, and OIL. These are all constraint languages for RDF, and what they do for RDF is similar to what DTDs, RELAX-NG, and XML Schema do for XML. However, since RDF is very different from XML, what these languages do is also somewhat different from what the XML constraint languages do. (We will return to this subject in section 1.4..)

The reader may have noted that OIL and DAML are not in the diagram above, and the reason for this is that they have been superseded by OWL. OIL and DAML started as research projects, and eventually merged into DAML+OIL. This was then taken up by the W3C, and is now being standardized as OWL, which builds on the older, and more basic, RDF Schema.

We will return to the subject of constraint languages in a later section, but first we will compare the data models that are the foundation of the constraint languages.

2. Comparing the technologies

When comparing two information technologies, especially ones as large and complex as topic maps and RDF, the best approach is to find some point of correspondence, some area where they can be made to match up almost exactly, and then to proceed from there. Luckily, such a point exists with RDF and topic maps, and, what is even

better, this point sits at the heart of both technologies.

2.1. Symbols and things

RDF and topic maps are both identity-based technologies. That is, the key concept in both is "symbols" representing identifiable "things", which statements can be made about. For example you may create a "symbol", then say, "this represents a person", then assign a name, and finally associate the "person" thing with a "company" thing via an employment relationship.

In topic maps the term for "thing" is *subject*, and the term for the construct used in topic maps to represent a subject is *topic*. So topic maps use topics to represent subjects, which are the things that we created the topic map to talk about.

In RDF the term for "thing" is *resource*. Now this may sound like it only means certain kinds of things, like documents, audio files, etc, but the definition of resource [1] makes it clear that it is as broad as that of subject, and can be anything at all. Inside RDF resources are represented by nodes (often called RDF nodes for clarity).

The concept of symbols representing things may sound like something every data representation has, but XML, for example, has nothing like it. XML data consists of elements, attributes, and character data, and none of these things are directly interpretable as symbols representing things. An element may represent a thing, a property, or it may just group a number of items. In addition, XML has no concept of the identity of an element, which actually makes it very different indeed from both topic maps and RDF.

This means that we can set up the table of correspondences shown below. The column headed "Correspondence" shows how closely the RDF and topic map terms correspond.

Reference	Topic maps	RDF	Correspondence
Thing	Subject	Resource	Exact
Symbol	Topic	Node	Close

Given this, we have a fixed point to start from, and can move on to see what light this sheds on the other aspects of the two technologies.

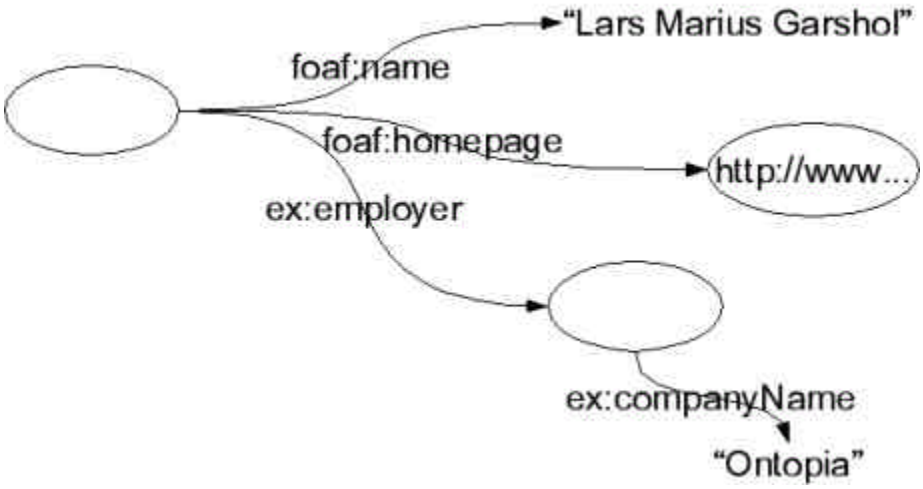
2.2. Assertions

So far we have only looked at the concept of "things", but what is interesting about topic maps and RDF is of course what they allow us to *say* about these things. This is where the first difference between RDF and topic maps shows up. RDF has only one way to make assertions about things, whereas topic maps have three different kinds of *topic characteristics*. These are names, occurrences, and associations.

The only form of assertion in RDF is what is known as a *statement*. A statement relates the subject (the node the statement is about) to the object (the node that is the value) via a property (a node that defines the relationship). The object may be either a node representing a resource, or a *literal* (a string).

The figure below shows RDF statements about me. The oval on the left is an RDF node representing me, and there are three statements about me. One assigns a name (using a property from the FOAF vocabulary; about which, more below), another provides my home page, and the third connects me with my employer, Ontopia. (The last two statements use the "ex" vocabulary, which was invented for this example.)

RDF statements



2.2.1. Names

In topic maps the simplest assertion that can be made about a thing is to assign a name to the topic that represents it. A name is a string, but can be qualified (as we will discuss in section 5.2.). Essentially this is a string property of the topic, which is privileged in the sense that the knowledge that it is a name for the topic is built into the model.

The diagram below shows a topic map with a topic representing me, which has a single name assigned. The name is shown on the topic, since the topic only has a single name.

Topic with name

In RDF assigning a name to a resource is done by using a name property (and which property that will be will depend on the RDF vocabulary being used) to assign a literal to the resource. This means that in order to recognize the property as a name one must have knowledge of the RDF vocabulary being used, something that is not necessary

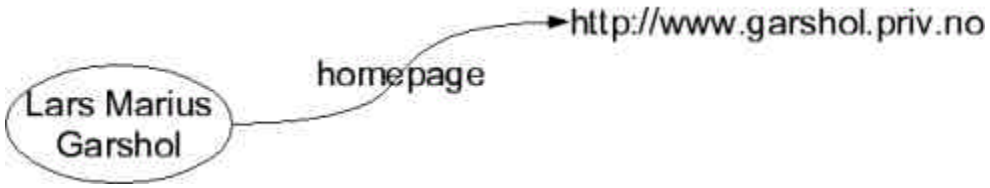
with topic maps. Above we chose `foaf:name` because we are using the foaf vocabulary as our example vocabulary in this paper.

2.2.2. Occurrences

We can also assign occurrences to topics. Occurrences can be properties of the topic stored as strings inside the topic map, or they can be references to information resources that are considered relevant to the topic. Occurrences have a type, which is a topic. The interesting thing about occurrences is that they are structurally identical to RDF properties, if we ignore the support for qualification. Semantically, however, there is significant difference, since RDF properties have less constrained semantics than do occurrences in topic maps.

The diagram below shows the topic map extended with a "homepage" occurrence for me.

Topic with occurrence

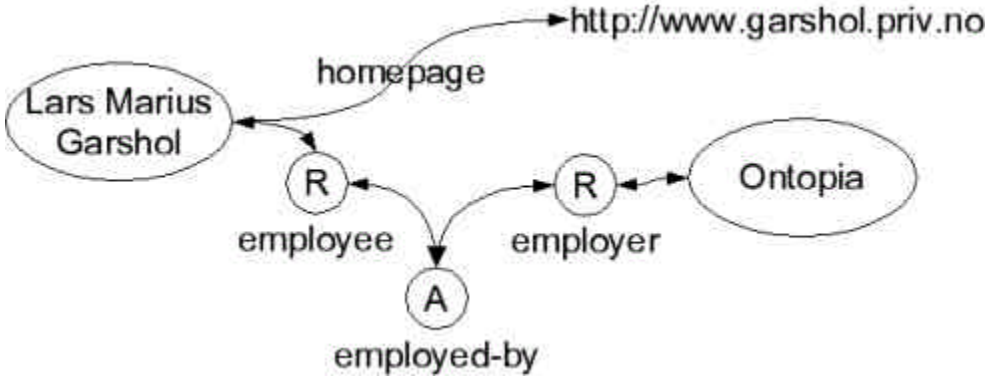


2.2.3. Associations

Finally, we can create associations between topics. Associations represent relationships between things. Associations have a type, and each topic participating in the association plays a role (which also has a type) within the association. This means that any number of topics can participate in an association, each playing its own role. (The types are all topics, just as with occurrences.)

The diagram below shows the topic map extended with an association of type "employed by" between me and Ontopia.

An association

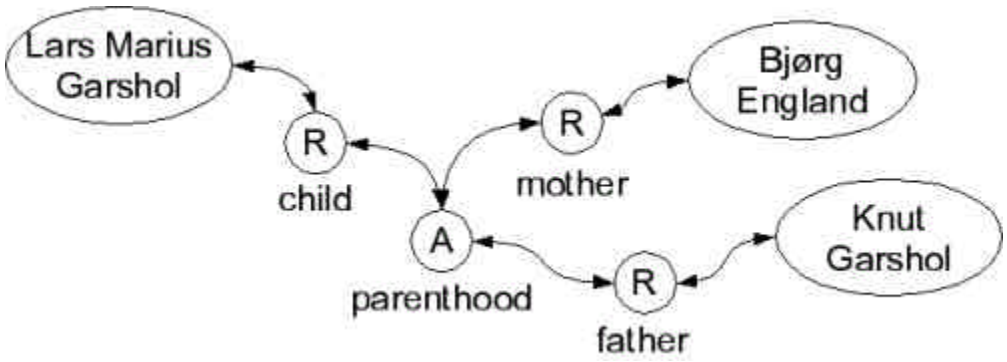


What is worth noting here is that firstly, a relationship between a person and a

company is considered different from a relationship between a person and a web page. Secondly, associations are completely different in structure from RDF statements. They have roles representing the involvement of each topic in the association, and they go both ways. That is, in topic maps saying that I am employed by Ontopia is *the same statement* as saying Ontopia employs me. This means that the issue of whether to make my employment a property of Ontopia or of me is a non-issue in topic maps; it will always be both.

A further consequence of this is that in topic maps associations can involve more than two topics, something that is impossible for RDF statements. An example of this can be seen below: an association representing my relationship with my parents. (And, obviously, theirs with me.)

A ternary association



Clearly, associations are structurally more complex than RDF statements, and also contain more information, in the form of the role types.

2.2.4. Summary

To summarize, a statement in RDF can be a name, an occurrence, or an association in topic maps. Names compare easily to RDF: they are RDF statements where the object is a literal and where the property has name semantics. The remaining RDF statements where the object is a literal are occurrences. However, RDF statements where the object is another resource are either occurrences or associations, depending on the semantics of the statement. If they *are* associations role types must be supplied. In addition to this comes the cases where in a topic map the association has more than two roles, in which case an intermediate RDF resource must be created.

Given this the terminology table below summarizes the terminology.

Reference	Topic maps	RDF	Correspondence
Assertion	Topic characteristic assignment	Statement	Close

2.3. Identity

Identity is about discovering when two symbols represent the same thing, how this is done, and what the consequences of such discoveries are. This concept is especially useful when it allows identity discoveries to be made automatically with data from different sources. Both RDF and topic maps are designed to support this, but in slightly different ways.

In RDF there are three kinds of nodes:

- literals (which we have already discussed).
- URI nodes. A URI is just a node that has a URI label on it, where the URI identifies the resource represented by the node. Since the URI directly identifies the resource represented by a node, RDF assumes that nodes with the same URI represent the same resource.
- *blank nodes*. These nodes are called blank, as they have no URI label. (Two examples can be seen in the RDF diagram example in section 2.2..) For blank nodes the only way to discover which resource they represent is to look at the statements made about them. RDF itself provides no standardized way to identify which blank nodes represent the same resources, although higher-level vocabularies like OWL do. (More about this in section 2.1.4..)

In topic maps, topics can be like blank nodes, but also like URI nodes. URIs can be attached to the topics to identify the subjects of the topics, which can make topics like URI nodes in RDF. If no such URIs are assigned, the topic is exactly like a blank node.

However, so far there is one problem we have glossed over. When we say that a URI can be used identify the thing a symbol represents it is not clear *how* this works. Let's say we use the URI `http://www.ontopia.net/` to identify a thing. Now, what does this actually identify? The information resource we get by resolving the URI? Or the thing described by that information resource? In practice, one finds that URIs are being used in both ways.

Topic maps distinguish these two cases, so that when assigning a URI to a topic as an identifier, the URI can be considered to be a *subject address* or a *subject identifier*. In the first case, the subject identified is the resource. In the second case it is whatever is described by the resource. In RDF, however, this distinction does not exist, and given a URI node there is no way to tell a priori which of the two ways the URI should be interpreted.

This is actually quite a thorny problem for interoperability between topic maps and RDF, and is also indicative of differences in the thinking behind the two. RDF practitioners would say that RDF models consist of statements and resources, ignoring the fact that the resources are not really part of the RDF model, but are represented by RDF nodes. In RDF, the distinction between the RDF model and the world it represents is not given much emphasis, whereas in topic maps this distinction premeates the whole model.

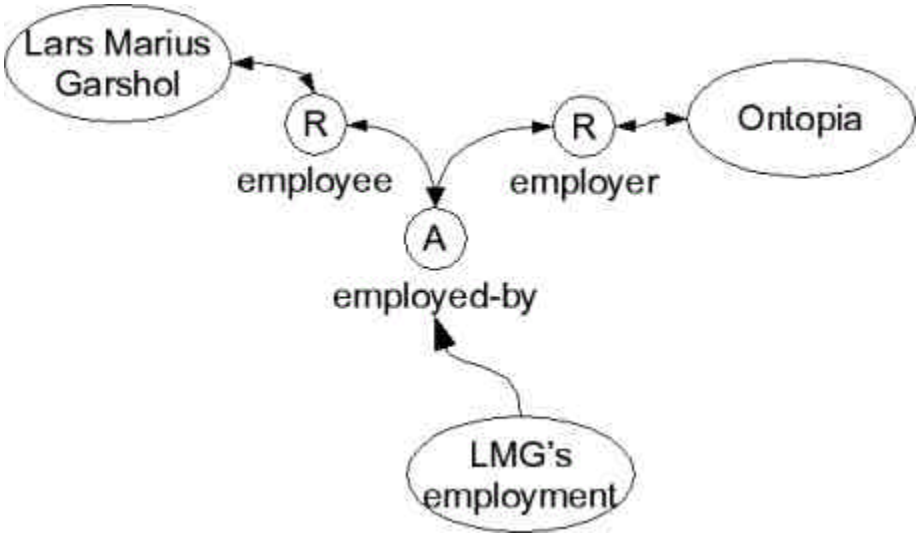
A more in-depth discussion of this issue can be found in [Pepper03].

2.4. Reification

The concept of reification has had the misfortune to have had stuck onto it a rather intimidating name, but it is in fact quite simple. Making assertions about things is easy, as you can create symbols to do so and then just make the assertions. Making assertions about assertions, on the other hand, is tricky. However, if you can create a symbol that represents an assertion you can just use that symbol to do it. Creating a symbol that represents an assertion is exactly what reification is[2].

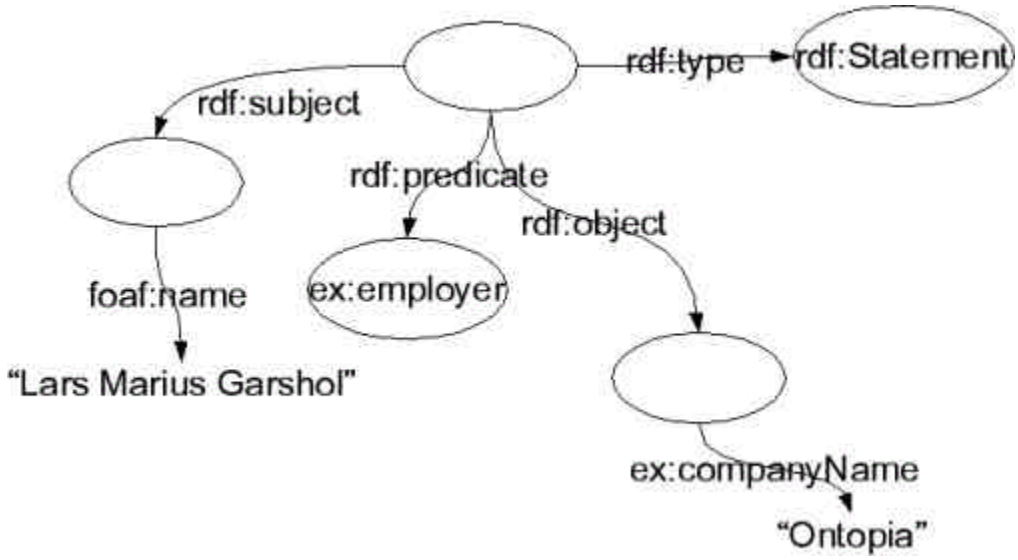
In topic maps you do this by creating a topic and then giving it a subject identifier that points to the name, occurrence, or association you want it to identify. After that the topic can be used like any other, but topic map software will detect what the topic represents, and act accordingly. The diagram below shows how it works.

Reifying an association



In RDF this works in a different way. A blank node is created and given the type of `rdf:Statement`. A special vocabulary is then used to add its subject, property, and object through statements. While this works, it is awkward, since reified statements must be treated in a different way from other statements. When traversing from a node to another via statements of property X, reified statements are traversed in a way that is different from unreified ones, which makes reification difficult to work with.

Reifying a statement



As the diagram shows, reification of RDF statements modify them in a way that reification does not modify topic characteristics. The path from me to Ontopia remains the same in the topic map diagram, but changes in the RDF diagram.

This problem is generally recognized in the RDF community, but to the best of the author's knowledge, no solution has been proposed. It should be added that while this is a problem, not supporting reification directly makes RDF more light-weight, and reification is not a very commonly used feature.

2.5. Qualification

Sometimes one wishes to qualify assertions made about things in order to record which authority claims they are true, what the source of the assertion is, or in what context the assertion is true.

In topic maps there is a built-in feature for this: *scope*. When an assertion is made in a topic map (in the form of a name, an occurrence, or an association) a scope is always attached to it. The default scope is the *unconstrained scope*, which means that there is no known limit to the validity of the assertion. Topics can be added to the scope to restrict under what circumstances it is considered to be true. Some examples are shown below.

- Topic maps are called "topic maps" in English, but in Norwegian they are called "emnekart". This is best represented by creating a single topic with two names, one in the scope English, and one in the scope Norwegian.

- There is a topic map with topics for each officially identified issue with the topic map standard, and these have occurrences of type "opinion", which gives an opinion about the issue. Each opinion is scoped with a topic representing the person who held the opinion.

- There is a group of people who believe that Francis Bacon wrote Shakespeare's

plays, and this might be represented by adding extra `authorship` associations between the plays and Francis Bacon, and scoping it with a topic representing this group of people.

In RDF there is no built-in feature for this, except that literals may have a language identifier attached to them, which is a kind of qualification. (A language identifier is a string conforming to RFC 3066[RFC 3066], such as `en-uk` or `pt-br`.) However, it is possible to achieve this through reification, since this turns the statement into a node about which statements (including qualifying ones) may be made. On the other hand, reification in RDF is, as we noted above, rather awkward in practical use.

Again it should be added that not having direct support for this makes RDF more light-weight, but this is a feature that is quite often needed, especially for internationalization, but also for tracking the sources of assertions. It should be added that although RDF itself does not support this, support for it can be built into individual applications by creating extra resources for assertions.

The key problem here is that statements in RDF have no identity, which means that it is impossible to make resources that represent them (without changing the statements) and since the model does not directly support qualification support for qualification cannot be added through reification. This is one of the most fundamental differences between topic maps and RDF, and one that has so far frustrated all attempts to model topic maps in RDF in a natural way.

2.6. Types and subtypes

One of the most important assertions you are likely to want to make about a thing is what *type* it belongs to. Doing so allows you to make it clear that, for example, this paper is a different kind of thing from its author, and that different rules apply to the two. This paper is a "paper", and papers have authors. Its author, on the other hand, is a "person", and persons may have employers. (Note the implications here: papers may *not* have employers, nor may persons have authors.)

In addition, we may also wish to record that papers are a special kind of document, so that every paper is a document, but that there are documents which are not papers (for example, standards are not papers). This is done with subtyping ("paper" is a subtype of "document"). The notion of types and subtypes is the basis for constraint languages, and is also crucial for querying and inferencing, so these are central features of any information technology.

In RDF, the type of a resource can be asserted using a statement with the `rdf:type` property, which is part of RDF itself. Subtyping is expressed with the `rdfs:subClassOf` property, which is part of RDF Schema, rather than RDF itself.

In topic maps the situation is similar: the topic map standard provides defined association types for the type-instance and supertype-subtype relationships. The only difference with RDF is really that both are part of the topic map standard itself and that both use the term "type" rather than "type" and "class".

2.7. Merging

Merging is the process of taking pieces of data and merging them together in such a way that assertions about the same things can be seen to be about the same things. Typically this happens through a stage where it is worked out what symbols represent the same things, and these symbols may then be merged.

In RDF nodes which are the same literal or use the same URI are merged in the sense that they become the same node. Beyond that, no merging is required. However, higher-level vocabularies like DAML+OIL and OWL do provide mechanisms that allow the identity of nodes to be inferred from the statements made about them, as described in section 2.1.4..

In topic maps topics which have the same subject address or subject identifier are merged, and become the same topic. Beyond that, no merging is required, although the new topic map standard provides a mechanism that allows identity to be inferred from assertions made about the topics. This mechanism is also discussed in section 2.1.4..

2.8. Summary

From the comparison we have made so far it seems clear that RDF and topic maps are very similar, yet different. In some key areas they match up very closely, while in others they are quite different. The most fundamental differences seem to be:

- the ways in which URIs are used to identify things, (one way in RDF, two ways in topic maps),
- the distinction between three kinds of assertions in topic maps, but only one in RDF, and
- the approaches taken to reification and qualification of assertions.

These three problems make it technically very difficult to merge topic maps and RDF into a single technology without making deep changes to one or both that are unlikely to be acceptable to their users. The problem with the two uses for URIs and the three kinds of assertions are the easiest to solve, but the problem with reification and qualification of assertions cannot be solved without changing the RDF representation of statements. The conclusion seems to be that a merge of the two technologies is out of the question.

It is also clear from the comparison that topic maps are higher-level than RDF, in the sense that a topic map contains more information about itself than does an RDF model. Without knowledge of specific RDF vocabularies an applications knows less about an RDF model than it does about a topic map. We can also conclude that RDF is inherently more light-weight than topic maps are. Optimization techniques for topic map implementations exist that to some extent make up for this difference.

This conclusion fits well with our earlier observation that RDF was meant for metadata about resources while topic maps are intended to provide a more high-level view of the subject domain covered by the resources.

3. Converting data

Now that we have done our tour of the two models, the time has come to look at how we can make them work together. Since we cannot merge the two technologies we should at least be able to move data between the two with as little effort as possible. Once the data has been moved from one form to another it should integrate naturally with data originally created in the new form and should take on the same form it would had it been created in this way originally. This to ensure that the end result is as natural to use as possible.

There are several ways in which one might envisage using topic map and RDF data together, the most obvious being:

- doing traditional conversion of data from one form to the other, and
- creating live virtual views that make RDF data appear to be topic map data (or vice versa) when viewed through a particular interface.

In both of these cases some form of mapping from the one representation to the other must be created. As it turns out, this is harder than it may at first glance seem, but some obvious alternatives to doing this do present themselves, as listed below.

- Modelling topic maps in RDF.
- Devising generic mappings.
- Devising vocabulary-specific mappings.
- Writing converters.

All of these are clearly within the realm of technical possibility, but all have their particular problems and advantages. Rather than summarize them here, we will move on to examining each in detail in the following sections.

3.1. Modelling topic maps in RDF

Modelling topic maps in RDF has been done several times, for example in [Moore01], [Lacher01], [Ogievetsky01], and [Garshol02]. Invariably, this results in using RDF nodes to represent names, occurrences, and associations, and using statements to attach types, scopes, and so on to the nodes.

As an illustration, below is shown parts of the topic for Puccini from Steve Pepper's Italian Opera topic map using the model of [Garshol02].

```
<tm:Topic>
  <tm:baseName>
    <tm:BaseName>
      <tm:value>Puccini, Giacomo</tm:value>
    </tm:BaseName>
  </tm:baseName>

  <!-- ...and so on... -->
```

As can be seen, the disadvantage of this approach is that it models topic maps in general, rather than the vocabulary used by the topic map, which means that the result is both heavy-weight and rather awkward to work with. Any query or retrieval specified in end-user terms will have to explicitly take into account topic map model features, and information from topic maps will not inteoperate cleanly with other RDF information.

One might attempt to improve on this by creating properties for the basic topic map constructs from which ordinary RDF properties might be specialized. This would tell us how to interpret the properties, and yet would allow the data to remain natural. Unfortunately, this leaves no other way to handle reification and scope than to use RDF reification, which is not really a satisfactory solution.

The conclusion must be that although this approach is easy to use, the results do not meet the criterion of clean integration with other RDF data, and that the problem lies in the approach itself rather than in the individual executions of it. We therefore abandon it without further ado.

One can easily model RDF in topic maps, as shown in [Moore01], but as this is no more satisfactory than what is discussed above we will not discuss this further.

3.2. Devising a generic mapping

Given how close RDF and topic maps are it may seem that it should be possible to devise a method for converting RDF data to topic maps directly, producing results that are more natural than those shown above. However, it turns out that there is a fundamental problem here, which is that not all the necessary information to do the mapping is present in the RDF data. Look at the RDF triple below, for example. (The first URI below is that of the subject, followed by that of the property, and finally the object.)

```
(http://example.com/X, http://example.com/Y, "foo")
```

Clearly, this RDF triple assigns a string to the subject. Equally clearly, we don't know if the string is a name or just a string property. Without this information we can't do the mapping to topic maps, since we don't know whether to create a name or an occurrence. We have a similar problem with the following triple.

```
(http://example.com/X, http://example.com/W, http://example.com/Z)
```

Again it is clear in RDF terms what is happening—a relationship between two resources is being asserted—but in topic map terms it is not at all clear. Is this an occurrence relationship, or a general association? And if it is an association, what are the types of the roles played by the subject and the object?

There are other problems as well, but already it seems clear that in the absence of knowledge about the vocabulary used by the RDF data it is not possible to convert this information to topic maps in such a way that the end result meets the criterion of naturalness.

3.3. Devising vocabulary-specific mappings

However, although the attempt to create a generic mapping failed it contains the germ of an idea that *does* work. Why not provide the converter with the distinctions necessary to do the conversion? Obviously, this information will be specific to the vocabulary being mapped, but even so it will only have to be provided once, and, once provided, can be used again and again.

The discussion above tells us that for each statement we must know whether it maps to the assignment of a name or occurrence, or to an association. A closer look informs us that they can also be mapped to subject identifier and subject address assignments. This information will then have to be provided as part of the mapping.

Having established this we can look at how to map the subject, and the obvious answer is that it must become a topic. The difficulty is what to do with its URI if it has one. Clearly, it must become either a subject identifier or a subject address, but how do we know which? This information, too, must be provided by the mapping.

Once we have decided what to do with the subject it turns out that in most cases completing the mapping requires more information, and that there are restrictions on what kind of node the object node may be. The table below summarizes this.

Mapping to	Information needed	Legal node types
Name	Scope	Literal
Occurrence	Type and scope	Literal and URI
Association	Type, scope, and role types	URI and blank
Subject address		URI
Subject identifier		URI

Having noted this, it's a short step to realizing that what we want to map a statement to will depend on the property used in the statement, since the property tells us how to interpret the statement. This means that we can use RDF to annotate the properties used in a vocabulary with the information needed to map the statements they are used in to topic maps. A mapping vocabulary has been designed for this purpose, and consists of the RDF properties (and values) shown below.

- `rtm:maps-to` is used to say what an RDF property maps to. Its possible values are `rtm:basename`, `rtm:occurrence`, `rtm:association`, `rtm:source-locator`, `rtm:subject-identifier`, `rtm:subject-address`, and `rtm:instance-of`. The last is a shorthand for creating a type-instance association between the subject and the object.

- `rtm:type` is used to specify the type of the occurrence or association the statement is mapped to. If not specified the type defaults to the topic the property itself maps to.

- `rtm:in-scope` is used to specify the members of the scope on the base name, occurrence, or association the statement is mapped to.

- `rtm:subject-role` and `rtm:object-role` are used to specify the types of the rule played by the subject and object respectively when mapping a statement to an association.

An RDF Schema for this mapping vocabulary, including more extensive documentation can be found in [Garshol03].

The greatest benefit of the approach is perhaps that it allows RDF vocabularies to also be topic map vocabularies. RDF information may with this carry its own mapping to topic maps around as part of itself. Similarly, when publishing an RDF schema for a vocabulary, a mapping to topic maps may be published at the same time and as part of the same package. No special syntax is needed to express this mapping, either, as the RDF syntaxes can do it just fine. And, most importantly, the same URIs are used in the topic map representation of the information as in the RDF representation.

As our example RDF vocabulary throughout this paper we will be using the "friend of a friend", or FOAF, vocabulary, created by Dan Brickley and Libby Miller. For more information about it, see [Brickley03]. Below is data about the author of this paper in the FOAF vocabulary:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">

<foaf:Person rdf:ID="larsga">
  <foaf:name>Lars Marius Garshol</foaf:name>
  <foaf:nick>larsbot</foaf:nick>

  <foaf:mbox rdf:resource="mailto:larsga@garshol.priv.no"/>
  <foaf:homepage rdf:resource="http://www.garshol.priv.no/" />

  <foaf:knows>
    <foaf:Person>
      <foaf:mbox rdf:resource="mailto:grove@ontopia.net"/>
    </foaf:Person>
  </foaf:knows>
</foaf:Person>
</rdf:RDF>
```

This should be mostly self-explanatory, with the possible exception of the `foaf:nick` property, which contains the nickname I use on IRC (Internet Relay Chat). A mapping

for the parts of the FOAF vocabulary used here can be found below, using the scheme described above. (Note that namespace prefixes are used in the `rdf:about` and `rdf:resource` attributes to make the example more readable, even though this is syntactically wrong.)

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:rtm="http://psi.ontopia.net/rtm/#">

<rdf:Description rdf:about="rdf:type">
  <rtm:maps-to rdf:resource="rtm:instance-of"/>
</rdf:Description>

<rdf:Description rdf:about="foaf:name">
  <rtm:maps-to rdf:resource="rtm:basename"/>
</rdf:Description>

<rdf:Description rdf:about="foaf:nick">
  <rtm:maps-to rdf:resource="rtm:basename"/>
  <rtm:in-scope rdf:resource="foaf:nick"/>
</rdf:Description>

<rdf:Description rdf:about="foaf:mbox">
  <rtm:maps-to rdf:resource="rtm:occurrence"/>
</rdf:Description>

<rdf:Description rdf:about="foaf:homepage">
  <rtm:maps-to rdf:resource="rtm:occurrence"/>
</rdf:Description>

<rdf:Description rdf:about="foaf:knows">
  <rtm:maps-to rdf:resource="rtm:association"/>
  <rtm:subject-role rdf:resource="foaf:Person"/>
  <rtm:object-role rdf:resource="foaf:Person"/>
</rdf:Description>
</rdf:RDF>
```

And that's all that's needed. This converts to the following, shown in XTM syntax below. (Again we use namespace prefixes in URIs to make the result more readable.)

```
  <topic id="id35">
    <instanceOf>
      <subjectIndicatorRef xlink:href="foaf:Person"/>
    </instanceOf>
    <baseName>
      <baseNameString>Lars Marius Garshol</baseNameString>
    </baseName>
    <baseName>
      <scope>
        <subjectIndicatorRef xlink:href="foaf:nick"/>
      </scope>
      <baseNameString>larsbot</baseNameString>
    </baseName>
    <occurrence>
      <instanceOf>
        <subjectIndicatorRef xlink:href="foaf:email"/>
      </instanceOf>
      <resourceRef xlink:href="mailto:larsga@ontopia.net"/>
```

```
  </occurrence>
  <occurrence>
    <instanceOf>
      <subjectIndicatorRef xlink:href="foaf:homepage"/>
    </instanceOf>
    <resourceRef xlink:href="http://www.garshol.priv.no"/>
  </occurrence>
</topic>

<association>
  <instanceOf>
    <subjectIndicatorRef xlink:href="foaf:knows"/>
  </instanceOf>
  <member>
    <roleSpec><subjectIndicatorRef xlink:href="foaf:Person"/></roleSpec>
    <topicRef xlink:href="#id35"/>
  </member>
  <member>
    <roleSpec><subjectIndicatorRef xlink:href="foaf:Person"/></roleSpec>
    <topicRef xlink:href="#id52"/>
  </member>
</association>
```

As can be seen, this produces a result that meets the criterion of naturalness, and after having used this method on a number of real-life RDF data sets successfully I feel confident in claiming that it suffices for the great majority of RDF to topic map conversion tasks.

The method produces topic map data that is equivalent in form to the original RDF data, which means that in cases where one wants to change the form of the data an additional step in the conversion process is needed. That is in the nature of simple, declarative mappings, and so this is not seen as a shortcoming.

In the beginning of this section we also noted a last approach: to write a special-purpose converter that did the conversion. This is of course possible, but since we have here found a much simpler declarative approach we abandon this approach as unnecessarily complicated without further examination.

3.4. Going the other way

So far we have only shown how to map RDF data to topic maps, but a complete solution will also need to be able to go the other way. Being more comfortable with topic maps than with RDF I would prefer to be able to maintain my FOAF data in topic maps, and only convert it to RDF for interchange with RDF users. The FOAF example shown above I would for example prefer to write as follows, using the LTM syntax[LTM].

```
[larsga : person = "Lars Marius Garshol"
                 = "larsbot" / nick]
{larsga, mbox, "mailto:larsga@ontopia.net"}
knows(larsga : person, grove : person)
```

```
{grove, mbox, "mailto:grove@ontopia.net"}
```

Now, given that the problem with converting RDF to topic maps is only that necessary information is missing, it would seem that going the other way should be possible *without* conversion, since one would expect that in this case the necessary information *is* present. In the case above, what is missing is the correct FOAF URIs for the vocabulary, which we can add as follows.

```
[person = "Person" @"http://xmlns.com/foaf/0.1/Person"]
[nick = "Nick"     @"http://xmlns.com/foaf/0.1/nick"]
[mbox = "Email"    @"http://xmlns.com/foaf/0.1/mbox"]
[knows = "Knows"   @"http://xmlns.com/foaf/0.1/knows"]
```

Given this we should have all the information we need to turn the topic map into proper FOAF RDF. We start with the `larsga` topic, which obviously turns into a blank node. The type we handle by adding an `rdf:type` property statement (this behaviour can be built in). The object of that statement is somewhat awkward, however. Clearly we need a node for the `person` topic, but how can we give it a URI? To solve the problem, let's assume that topics which are part of the vocabulary (that is, the types) get as their URI one of their subject identifiers, chosen at random. (The remaining identifiers can be handled with `owl:equivalentClass`, which is discussed below.)

So far, so good. The next step is the name, and here we run into a problem: we don't know which property to use for the name. RDF resources of different types tend to have different name properties, so here we are left without the necessary information. Let's ignore that for the moment and move on, remembering that if we can find a way to provide this information we can at least create vocabulary-specific mappings.

Next is the `"larsbot"` name, and since it has a single topic in its scope we can use that topic to give us the right property by using the same method we used with the type. We can handle the `mbox` occurrence in the same way, as the distinction between internal and external occurrences in topic maps tell us whether to create a URI node or a literal for the object.

The `knows` association we can turn into a statement where we create the property from the association type using the same method as before. However, we have a problem in that we don't know which role to make the subject of the statement. If the two roles were of different types the mapping could tell us that, but this is a symmetric association, and so there is no way to tell the roles apart. This is solved by picking a subject at random (and using `owl:SymmetricProperty` to assert that the property is symmetric).

From this it appears that it is impossible to create a generic mapping from topic maps to RDF, but that vocabulary-specific ones similar to what we created for going from RDF to topic maps should be possible. It would be ideal if the RDF to topic maps mappings could also be used to go the other way. This works well for associations, since the mapping would tell us which association role to start from, but does not work for names, as many RDF properties map to names.

In addition, there are many possible cases that might occur in a conversion that we

have not yet covered. More work is necessary to devise a complete mapping, but some of the issues can be solved as described below.

- Multiple URIs for the same topic can be handled using the RDF properties for equivalence found in OWL.

- Associations with more than two roles can be turned into resources whose type is the association type, and each role can then be represented as a separate statement with the role type as the property and the association resource as the subject.

- Reification and scoping can in general be represented by using RDF reification to represent the statement that would connect the topic characteristic with the topic. A special property will have to be defined for representing scope. As for the reification this is done by simply merging the resource representing the topic characteristic assignment with that representing the reifying topic.

- Binary non-symmetric associations can be handled by having the mapping contain one association from the association type to the preferred subject role.

- Selection of name properties can be done by having the mapping contain an association from the topic type to a topic representing the preferred RDF name property.

This does not cover all issues, unfortunately, such as the handling of unary associations, variant names, and a number of tricky edge cases. In addition, some thinking is necessary to come up with good rules for which topics and associations to leave out of the conversion, as one probably will not always want to convert *all* the information in the topic map to RDF. A later version of this paper will describe how this can be done.

4. Aligning the standards families

The previous section provides mechanisms for moving data between RDF and topic maps, and this means that we now have at least some level of interoperability between the two in terms of their data models. This also provides syntax interoperability, since mapping then becomes a matter of first deserializing the data, then mapping it.

However, topic maps and RDF are both families of standards, and include more than just syntaxes and a data model. There are also constraint languages, and at some point in the future there will be query languages. For full interoperability these will also have to be considered.

4.1. The constraint languages

If we have created a schema for our data in its original form, whether RDF or topic maps, it would be ideal if this could be reused when the data is represented in the other form. If this is not possible it will mean that the schema will have to be rewritten in a new constraint language for the new representation, which will require a considerable amount of manual effort.

The easiest way to make this possible would be to merge the RDF Schema/OWL and TMCL constraint languages into a single language. The difficulty with this is that topic maps and RDF are structurally different, which means that constraints will by necessity be specified differently in the two languages. For example, TMCL needs to be able to constraint the scope of topic characteristics as well as the roles of associations, neither of which exist in RDF. However, some of the parts of OWL that are not about constraints might also be used in topic maps. We will consider this in more detail below.

Clearly, making it possible to reuse schemas depends on two things: whether the design of the constraint languages allows this and whether we can come up with a method for converting between the constraint languages. As neither OWL nor TMCL are finished yet it should be possible to avoid incompatibilities in design. Unfortunately, this also makes it difficult to create mappings between them. However, already now we can see that there are two different approaches we might take:

- Validating topic map information using an RDF schema and a mapping for that vocabulary from RDF to topic maps. (And vice versa.)

- Directly converting from the RDF constraint language to a topic map constraint language, using the RDF to topic map mapping to drive the conversion. (And, again, vice versa.)

On closer examination it becomes clear that these two approaches are in fact quite closely related. The principal difference is that the first requires the writing of specific utilities for all the different purposes the schema may be put to (validation, schema-driven editing, ...), while the second is limited to what can be expressed in existing topic map constraint languages. Since our goal is to align the standards families more than to create a validation solution for topic maps we choose the latter approach here.

4.1.1. Converting RDF Schema

As previously noted TMCL does not yet exist, but there is an existing schema language for topic maps: OSL (Ontopia Schema Language), defined in [Ontopia02]. This is a simple constraint language for topic maps created as input to the TMCL process, but certainly different from the final result of that process. To demonstrate the feasibility of our approach we will define a method for converting RDF Schema into OSL.

RDF Schema is relatively simple, and allows the definitions of two kinds of things: classes and properties. Both may be assigned a label and a comment as documentation. Classes may have superclasses, and properties may have domains (the allowed classes of subjects) and a range (the allowed class of the object). There are a few more features in RDF Schema, but this is the heart of it. RDF Schema is an RDF

vocabulary, and so is expressed in RDF.

Below is a subset of the RDF Schema for FOAF:

```
<rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Person"
  rdfs:label="Person"
  rdfs:comment="A person.">
  <rdfs:subClassOf rdf:resource="http://xmlns.com/wordnet/1.6/Person"/>
</rdfs:Class>

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/geekcode"
  rdfs:label="geekcode"
  rdfs:comment="A textual geekcode for this person, [...]">
  <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
</rdf:Property>

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/nick"
  rdfs:label="nickname"
  rdfs:comment="A short informal nickname [...]">
</rdf:Property>

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/name"
  rdfs:label="name"
  rdfs:comment="A name for some thing.">
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"
</rdf:Property>

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/plan"
  rdfs:label="plan"
  rdfs:comment="A .plan comment, [...]">
  <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"
</rdf:Property>

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/knows"
  rdfs:label="knows"
  rdfs:comment="A person known by this person [...]">
  <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  <rdfs:range rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
</rdf:Property>
```

OSL is rather different, in that it is organized around the declaration of topic and association classes. Topic classes may have base name, occurrence, and role constraints, while association classes may have role constraints only. There is support for cardinality constraints, for constraining scope, and for indicating whether or not the schema (as well as individual classes) are open or closed.

For a schema to be open means that if topics or associations of classes not defined in the schema are found they are ignored rather than considered errors. If the schema is closed, however, such topics/associations are considered to be in error. Similarly, a topic class can be closed or open, which controls whether topic characteristics not matching a declared constraint are considered errors or just silently ignored. RDF Schema, by contrast, is always open, though individual applications may of course consider it closed.

Despite these differences it *is* possible to convert an RDF Schema document with a

topic map mapping into an equivalent topic map with an accompanying OSL schema. The reason the mapping must produce both a topic map and a schema is that an OSL schema is not a topic map, and yet the RDF Schema provides information about the typing topics that would normally be part of a topic map. The most important examples of this are the names of the typing topics and comments about them.

A mapping for the parts of RDF Schema that are best represented in topic map form are given below.

```
<rdf:Description rdf:about="http://www.w3.org/2000/01/rdf-schema#label">
  <rtm:maps-to rdf:resource="http://psi.ontopia.net/rtm/#basename"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/2000/01/rdf-schema#comment">
  <rtm:maps-to rdf:resource="http://psi.ontopia.net/rtm/#occurrence"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/2000/01/rdf-schema#subClassOf":
  <rtm:maps-to rdf:resource="http://psi.ontopia.net/rtm/#association"/>
  <rtm:type rdf:resource="http://www.topicmaps.org/xtm/1.0/core.xtm#superclas
  <rtm:subject-role rdf:resource="http://www.topicmaps.org/xtm/1.0/core.xtm#s
  <rtm:object-role rdf:resource="http://www.topicmaps.org/xtm/1.0/core.xtm#su
</rdf:Description>
```

We also need to convert the constraints in the RDF Schema into an equivalent OSL schema, which is done by following the procedure below.

1. For every `rdfs:Class` create a topic class.

    1. For every `rdf:Property` that has this class as a legal domain, create a constraint. Use `rtm:maps-to` to determine whether to create a base name, occurrence, or role constraint.

    2. If the property has `rtm:in-scope` statements define the corresponding scope rules on the constraint. Require an exact match.

    3. For occurrence constraints set the type to that given by `rtm:type`, or, if `rtm:type` is absent, the property itself. If there is a `rdfs:range` statement for the property, make the occurrence an internal occurrence if the value is `rdfs:Literal`, and make it external if the value is anything else. If there is no value, leave it open.

    4. For role constraints set the association type to that given by `rtm:type`, or, if `rtm:type` is absent, the property itself. Set the role type to that given by `rtm:subject-role`.

2. For every `rdf:Property` which has an `rtm:maps-to` statement with `rtm:association` as the value, create an association class. Set the type to that given by `rtm:type`, or, if `rtm:type` is absent, the property itself.

    1. If the property has `rtm:in-scope` statements define the corresponding scope rules on the class. Require an exact match.

2. If the association type is not symmetric, create one role constraint with the type set to that given by `rtm:subject-role`, with a cardinality of exactly 1. Then, create one role constraint with the type set to that given by `rtm:object-role`, with a cardinality of exactly 1.

3. If the association type *is* symmetric, create one role constrain with the type set to that given by `rtm:subject-role`, with a cardinality of exactly 2.

Since RDF Schema assumes an open world the schema and the classes generated are all left open. However, one problem that is not so easy to solve is the issue of which classes topics are allowed to be members of. In RDF Schema it must be explicitly expressed if instances of one class may not be instances of another, but in OSL it is the other way around: for instances of one class to be allowed to be instances of another, this must be explicitly stated.

One way to solve this may be to explicitly allow every generated class to be an instance of every other generated class which is not specified to be disjoint in the RDF Schema. This should probably be made a user option in an actual converter implementation, as it is likely that not all RDF Schemas will provide disjointness information.

If we use this algorithm on the subset of the FOAF schema shown above the result is the following OSL schema.

```
<tm-schema match="loose">
  <topic match="loose" id="id2">
    <instanceOf subclasses="yes">
      <subjectIndicatorRef href="http://xmlns.com/foaf/0.1/Person"/>
    </instanceOf>
    <occurrence internal="yes">
      <instanceOf subclasses="yes">
        <subjectIndicatorRef href="http://xmlns.com/foaf/0.1/plan"/>
      </instanceOf>
    </occurrence>
    <occurrence internal="either">
      <instanceOf subclasses="yes">
        <subjectIndicatorRef href="http://xmlns.com/foaf/0.1/geekcode"/>
      </instanceOf>
    </occurrence>
    <playing>
      <instanceOf subclasses="yes">
        <subjectIndicatorRef href="http://xmlns.com/foaf/0.1/Person"/>
      </instanceOf>
      <in>
        <instanceOf subclasses="yes">
          <subjectIndicatorRef href="http://xmlns.com/foaf/0.1/knows"/>
        </instanceOf>
      </in>
    </playing>
  </topic>

  <association>
    <instanceOf subclasses="yes">
      <subjectIndicatorRef href="http://xmlns.com/foaf/0.1/knows"/>
```

```
    </instanceOf>
    <role min="1" max="1">
      <instanceOf subclasses="yes">
        <subjectIndicatorRef href="http://xmlns.com/foaf/0.1/Person"/>
      </instanceOf>
    </role>
    <role min="1" max="1">
      <instanceOf subclasses="yes">
        <subjectIndicatorRef href="http://xmlns.com/foaf/0.1/Person"/>
      </instanceOf>
    </role>
  </association>
</tm-schema>
```

Clearly, this is a useful method that allows RDF vocabularies to have their RDF Schemas converted into both topic map information and a corresponding OSL schema. One expects that once TMCL is defined it will be possible to devise a similar mapping to it. Having such a mapping mechanism makes it much easier for users to live with both topic maps and RDF.

One of the things one might consider using RDF Schema for is to validate topic maps with a standardized constraint language before TMCL appears. However, as was shown above, that is only possible in a very limited way. The combination of RDF Schema and our mapping vocabulary cannot deal with scope except in the most limited way, and also cannot handle associations with more than two roles, and so on.

If we now reconsider the approach that we did not take: that of writing a topic map validator that uses an RDF Schema and a mapping to do its validation it becomes clear that any such solution would suffer from the same problem, and so is not very interesting.

4.1.2. Converting OWL

Converting OWL to topic maps and OSL is a rather different proposition from converting RDF Schema, since OWL goes far beyond the facilities provided by RDF Schema, and indeed beyond those provided by OSL. We start out with a review of OWL features and how they can best be represented in topic maps.

OWL is an RDF vocabulary, but describing it as a constraint language is a little unfair, because the language goes beyond that. It has the ability to constrain the values of properties to some extent, and it also supports cardinality constraints, but it also goes beyond constraints into what is perhaps best described as semantic annotation of RDF data. This allows certain aspects of the interpretation of RDF vocabularies to be described, and is perhaps why OWL is described not just as a schema language, but as an "ontology language".

OWL contains within it two defined subsets of the language:

- OWL Lite is a subset of the OWL vocabulary intended for those who want a language that is easier to implement. Most of the semantic annotation capabilities are left out, as are some of the constraint capabilities.

- OWL Description Logic (or DL) contains the full vocabulary but restricts how it may be used, in order to provide logical inferencing engines with certain properties desirable for optimization.

To distinguish it from these two subsets OWL itself is sometimes also referred to as OWL Full.

A rough summary of the OWL language features and how they map to topic maps and OSL is given below.

- Metadata. OWL has features for providing information about an ontology, such as importing other OWL ontologies, version information, and compatibility relationships with other versions. These can be mapped to topic maps directly, being simple descriptive properties.

- Class definitions. `owl:Class` is a subclass of `rdfs:Class`, and in addition to what RDF Schema does OWL can define an enumerated class as well as classes defined by set operations on existing classes. OSL has no counterpart to this, and would have to be extended to support this capability. We note this as a desirable feature for TMCL, but can do no more at present.

- Value restrictions. These allow restrictions on property values from RDF Schema to be further constrained based on the class of the subject. For example, an OWL ontology might say that the object of `hasMaker` properties of `Wine`s must be instances of `Winery`, even though this does not apply to this property in the general case.

- Cardinality restriction. These allow the cardinalities of statements as applied to specific classes to be constrained. This is the same as the cardinality constraints in OSL, and can easily be mapped to these, by extending the algorithm given above.

- Global cardinalities. `owl:FunctionalProperty` is a subclass of `rdf:Property` which is the class of properties where subjects may only have a single value. (This is essentially the same as giving the property a maximum cardinality of one.)

  `owl:InverseFunctionalProperty` is another property class where two different subjects are not allowed to have the same value. To put it another way, the property uniquely defines the subject. This is the same notion as the unique characteristic type of [SAM], and so best mapped to topic maps by changing the property class to the unique characteristic type defined by SAM.

- Subclass relationships. OWL reuses the `rdfs:subClassOf` property from RDF Schema, with the same semantics. It can therefore be mapped in the same way.

- Equivalent and disjoint classes. `owl:equivalentClass` can be used to specify that two classes have the same extension, although not necessarily the same intension. OSL does not have this, and it is most naturally mapped to an association between the topics representing the two classes in topic maps.

`owl:disjointWith` is used to express the fact that the instances of one class never can be instances of another class. OSL has the opposite of this (one must state when two classes may share instances, as the default is that this is not allowed). This may be mapped to an association between the class topics in topic maps.

- Object versus datatype properties. OWL defines two subclasses of `rdf:Property` (`owl:DatatypeProperty` and `owl:ObjectProperty`). The first allows only literals as values, while the second only allows instances of a class (essentially non-literals). This is essentially a subset of the capabilities provided by `rdfs:range`, and so can be used to make the same inferences while converting schemas.

- Equivalent properties. The `owl:equivalentProperty` property can be used to assert that two properties have the same extension, although they may still have different intensions. Topic maps have no equivalent, and this is best mapped to an association with the same semantics.

- Inverse properties. The `owl:inverseOf` property can be used to assert that one object property is the inverse of another. As object properties map to associations in topic maps, this information is not needed in topic maps.

- Transitive and symmetric properties. OWL defines two classes of properties (`owl:TransitiveProperty` and `owl:SymmetricProperty`), which are both subclasses of `owl:ObjectProperty`. Topic maps do not have an equivalent, but these classes can be reused directly to provide additional information about association types.

  Note that the `owl:SymmetricProperty` class is not really needed in topic maps, as this can be inferred from the role types of associations.

Below is a mapping for the parts of the OWL vocabulary that can be expressed as part of topic maps. (Note that entities are are used for the URI prefixes in order to make the example more readable.)

```
<rdf:Description rdf:about="&owl;equivalentClass">
  <rtm:maps-to rdf:resource="&rtm;association"/>
  <rtm:subject-role  rdf:resource="&owl;Class"/>
  <rtm:object-role  rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&owl;disjointWith">
  <rtm:maps-to rdf:resource="&rtm;association"/>
  <rtm:subject-role  rdf:resource="&owl;Class"/>
  <rtm:object-role  rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&owl;equivalentProperty">
  <rtm:maps-to rdf:resource="&rtm;association"/>
  <rtm:subject-role  rdf:resource="&rdf;Property"/>
  <rtm:object-role  rdf:resource="&rdf;Property"/>
</rdf:Description>
```

```
<!-- this handles the property classes -->
<rdf:Description rdf:about="&rdf;type">
  <rtm:maps-to rdf:resource="&rtm;instance-of"/>
</rdf:Description>
```

Thus we see that OWL plays two roles here: it provides more constraints that allow us to produce tighter OSL schemas from RDF, and it also provides semantic annotations of the types in the schema.

4.1.3. Conclusions

The obvious conclusion is that converting the constraints in an RDF Schema or an OWL ontology to the future TMCL should not be too difficult, provided an RDF to topic maps mapping is provided. It also appears that the parts of OWL that go beyond simple constraints can be reused directly in topic maps for the most part. For this reason it seems best if TMCL is created to only support constraints, and that rather than define a topic map ontology language OWL is reused as it stands within the topic map standards family.

It seems unlikely that OWL can be reused entirely without changes, but a technical report or something similar might be published which explains which parts of OWL are meaningful in topic maps and how to interpret those parts. Conventions for the use of OWL in topic maps which specifies which role types to use and so on could also be given in such a technical report.

4.2. The query languages

ISO has defined a new work item for TMQL (Topic Map Query Language), to become ISO 18048, and work on this has started in the form of requirements and use case gathering. No specific query model has been decided on yet, although a number of proposals have been put forward. The W3C is in a similar state with RDF, where a number of proposals exist, but no official query language activity has been started yet.

The creation of query languages provides a second possible approach to interoperability between the two technologies: integration at the query language level. That is, in theory the same language might be used to query both forms of information. This has a number of uses.

- An application may keep some information in topic map form, and some in RDF form, and yet query it as if it were a single store of information. This may remove the need to perform conversions and may be more useful in situations where the RDF and topic map stores are updated by different applications.

- With a query language that supports modification cross-querying can be used to perform more complex conversions than those supported by the simple mapping vocabulary described in this paper.

- Network protocols for accessing topic map and RDF data stores can be defined that use querying yet are independent of whether the storage form is topic maps

or RDF. The mapping solutions described in this paper can be used to provide query results in the form expected by the client application.

Two approaches to making it possible to query both data representations stand out as the most likely:

- Using a query language designed for one of the two and using the mapping vocabulary to translate queries into a query language designed for the other.
- Creating a query language that can query both representations.

The first approach may look the best and easiest at first sight, but there are some technical problems. For queries that do joins across technology boundaries the translator will have to implement joins itself, something that may in be impossible to do efficiently in many cases (for example when the data is stored in databases).

As for the second approach, although it may look difficult it turns out that it is not actually difficult at all. The tolog query language proposed by Ontopia ([Garshol01] and [tolog]) can query topic maps, and it turns out that it can be extended to also support querying of RDF. In fact, tolog can perfectly well be used as a pure RDF query language with no connections to topic maps at all.

The core of tolog is simply Prolog-like predicate clauses and rules, with syntactic sugar for direct support for or, as well projection, aggregate functions, and ordering as known from SQL. Syntactically, queries typically look like the following:

```
select $A, count($B) from
  foo($A, $C),
  bar($C, $B)
order by $B desc?
```

tolog provides built-in predicates for querying topic maps as well as a way of querying topic map associations using a minor syntactic extension. Using this extension the following finds all employees of Ontopia:

```
employed-by($A : employee, ontopia : employer)?
```

RDF statements map very easily to tolog predicates, so adding support for RDF can be done in two ways.

- Adding a predicate `rdf(subject, property, object)` that queries RDF triples directly.
- Adding the ability to name RDF properties as predicates directly in the same way that association types can be named as predicates. This would give a syntax like `property(subject, object)?`

The first is more awkward than the second, but allows greater flexibility, as one can find all properties linking two specific resources. However, it turns out that there are some problems with both approaches:

- How to refer to RDF resources? RDF resources can be referred to via URIs and clearly the identifiers used above will not suffice for this. There are some solutions that might be done as simple extensions, but these are kluges[3], and so not the way to go.

- When a query does joins across topic maps and RDF, how are values to be compared? The processor needs to be able to compare RDF nodes with topics in order to do joins across the two representaions.

The current version of tolog (at the time of writing[4]) is 0.1, and the language is in any case ready for some generalizations and extensions. We will cover these in the next section before moving on to looking at how this will help us query both RDF and topic maps, and how this is relevant to interoperability between the standards families.

4.2.1. Proposals for tolog 1.0

The relevant parts proposed for tolog 1.0 are really three:

- A module system which declares binds identifier prefixes used in queries to specific name spaces. These name spaces can be of several different kinds.

- A type system that defines minimal rules for data types supported by the query language. This allows modules to add new data types to the system without requiring the language itself to change.

- A set of built-in predicates for querying all aspects of topic maps.

The type system simply requires that every value must have a well-defined type, and that the type must define rules for ordering values of that type, and for comparing them for equality.

The module system allows prefixes to be declared that work in ways analogous to the prefixes in XML Namespaces. With this system a query to find all the top-level classes in a topic map can be written as follows:

```
using xtm for "http://www.topicmaps.org/xtm/1.0/core.xtm#" as indicator
select $SUPER from
  xtm:superclass-subclass($SUPER : xtm:superclass, $SUB : xtm:subclass),
  not(xtm:superclass-subclass($OTHER : xtm:subclass, $SUPER : xtm:subclass))?
```

The last keyword in the `using` clause tells us how to interpret the URI in order to create values when the prefix is used. A URI is always formed by concatenating the prefix URI with the last part of the identifier. In this case, the topic with this URI as its subject identifier is looked up and returned. The following alternatives is being considered:

- `address`: Used to look up topics by subject address.

- `locator`: Used to look up topics by source locator.

- `module`: Reads a file with inference rules from the given URI. (May also be used to bind to other kinds of modules.)

- `rdf`: Used to look up RDF nodes by URI.

Given this we might do a query across the FOAF vocabulary and Ontopia's XML conference topic map to find out who I know that works for Empolis[5] as follows.

```
using foaf for "http://xmlns.com/foaf/0.1/" as rdf
      xc for "http://psi.ontopia.net/xmlconf/#" as indicator
select $B from
  foaf:mbox($A, "mailto:larsga@ontopia.net"),
  foaf:knows($A, $B),
  xc:employed-by($B : xc:employee, $C : xc:employer),
  xc:homepage($C, "http://www.empolis.com")?
```

The difficulty we run into here is that `$B` is first bound to RDF nodes, then directly compared with topics. Unfortunately, `$B` will be a person, and so is likely to be represented by a blank node in RDF and a topic with no subject identifier (much less a subject address) in topic maps.

Even if the RDF node/topic had had URIs assigned to it this would not have helped the query processor, since it would not have known whether to consider the URI of the RDF node a subject identifier or a subject address when comparing it with the topics. Given this, it seems clear that direct comparison of RDF nodes with topics is just not going to work.

Instead, URIs and other identifying values must be found which can determine identity across the RDF/topic maps divide. In the example above this would be the email address of the person in question. To reach this in topic maps we need to be able to query occurrences. In other cases we might have to be able to find the subject identifier and/or address of topics, which requires other extensions. tolog 1.0 will provide these, so this will not be a problem. To round off the example, here is the query above in working form.

```
using foaf for "http://xmlns.com/foaf/0.1/" as rdf
      xc for "http://psi.ontopia.net/xmlconf/#" as indicator
select $B from
  foaf:mbox($A, "mailto:larsga@ontopia.net"),
  foaf:knows($A, $B),
  foaf:mbox($B, $BMAIL),
  xc:email($BTM, $BMAIL),
  xc:employed-by($BTM : xc:employee, $C : xc:employer),
  xc:homepage($C, "http://www.empolis.com")?
```

Since both email addresses are URIs this will work just fine.

4.2.2. One language, or two?

> Frustra fit per plura, quod fieri potest per pauciora. (William of Ockham)

So far we have seen that creating a single query language that can query both topic

maps and RDF is technically possible, and that the result need not be awkward at all. This raises the question of why there should be two different query languages at all. Unfortunately, it turns out there are a few reasons in favour of making them different, and these are listed below.

- One reason can be stated in a single word: politics. Topic maps, and TMQL, are standardized in ISO, while RDF and its query language are standardized by the W3C. This means that to do this two different standards bodies, building on two different data models and terminologies, with different goals, have to communicate to create a common query language. It is possible, but it will not be easy.

- A further problem is that catering to both data models requires some extensions that might not have been necessary had the language been intended only for RDF. For example, the three extra keywords in the `as` part of the `using` clause are not needed for RDF. Similarly, pair arguments (those of the form `x : y`) do not seem needed for RDF. Quite likely, RDF will require similar extensions that will not be needed for topic maps.

- Finally, there is the question of how many query processors are likely to actually implement support for both topic maps and RDF. To do so efficiently when the information is stored in a database requires the query implementation to be built on an engine that supports both data representations, to avoid the problem of joins across the technology divide.

In short, there are a number of obstacles in the way. However, there are also some benefits to be had from merging the two query languages, the most important of which are:

- Both languages will need a set of operators for handling values of various types, such as numbers, strings, and dates. Developing a common vocabulary of such operators will spare development effort, reduce the amount that users/developers have to learn, and simplify interoperability.

- The two languages will need roughly similar functionality, and merging them makes it possible for the two communities to do a joint design (it is here assumed that this will necessarily be better than separate designs) and again this will mean less for developers and users to learn.

So far the argument appears inconclusive. The possibility of a single query language is interesting, but more work should be done to determine whether it is politically and technically feasible.

5. Conclusions

So, at the end of 30-odd pages of examination of topic maps and RDF, what have we learned? The key lessons are that:

- Merging the two technologies does not appear desirable or possible.
- It is possible to convert data back and forth between the two representations using simple, declarative, vocabulary-specific mappings.
- This makes it possible for RDF and topic maps to have shared vocabularies.
- RDF constraints can be converted to topic map constraints given such a mapping.
- Semantic annotations in OWL can be translated directly into a topic map representation of the same information. That is, the descriptive part of OWL can be used both with RDF and with topic maps.
- It is possible to create a single query language for both RDF and topic maps.

In short, it does appear that it is possible to live with both RDF and topic maps.

Acknowledgements

Thanks to Dan Brickley for answering my questions on the new RDF specifications and confirming some of my interpretations of the OWL and RDF Schema specifications.

Thanks to Steve Pepper for useful feedback on earlier drafts of this paper.

Also thanks to all the members of the #rdfig IRC channel that have taught me about RDF over the past few years. This paper would not have been nearly as good without your help.

Bibliography

Brickley03
FOAF: the 'friend of a friend' vocabulary, Dan Brickley and Libby Miller, 2003-02-23. Available from http://xmlns.com/foaf/0.1/.

Garshol01
tolog — A topic map query language, Lars Marius Garshol, Ontopia. Presented at XML Europe 2001, Berlin, Germany. Available from *http://www.ontopia.net/topicmaps/materials/tolog.html*.

Garshol01b
Topic maps, RDF, DAML, OIL, Lars Marius Garshol, Ontopia. Presented at

XML 2001, Orlando, USA. Available from *http://www.ontopia.net/topicmaps/materials/tmrdfoildaml.html*.

Garshol02
An RDF Schema for topic maps. Lars Marius Garshol, Ontopia. Available from http://psi.ontopia.net/rdf/.

Garshol03
An RDF vocabulary for RDF2TM mapping. Lars Marius Garshol, Ontopia. Available from http://psi.ontopia.net/rtm/.

Lacher01
On the integration of Topic Map data and RDF data, Martin Lacher and Stefan Decker, presented at Extreme Markup Languages 2001, Montréal, Canada. Available from http://www.semanticweb.org/SWWS/program/full/paper53.pdf.

LTM
The Linear Topic Map Notation: Definition and Introduction, Version 1.2, Lars Marius Garshol, Ontopia. Available online at http://www.ontopia.net/download/ltm.html.

Moore01
RDF and TopicMaps: An Exercise in Convergence, Graham Moore, presented at XML Europe 2001 in Berlin. Available from http://www.topicmaps.com/topicmapsrdf.pdf.

Ogievetsky01
XML Topic Maps through RDF glasses, Nikita Ogievetsky, presented at Extreme Markup Languages 2001, Montréal, Canada. Available from http://www.cogx.com/rdfglasses.html.

Ontopia02
The Ontopia Schema Language: Reference Specification, Ontopia, 2002-11-25. Available online at *http://www.ontopia.net/omnigator/docs/schema/spec.html*.

Pepper99
Euler, Revolution, and Topic Maps. Steve Pepper, Ontopia. XML Europe 1999. Available from http://www.ontopia.net/topicmaps/materials/euler.pdf.

Pepper03
Curing the Web's Identity Crisis: Subject Indicators for RDF, Steve Pepper and Sylvia Schwab, Ontopia. To be presented at XML Europe 2003.

RFC 3066
RFC 3066 - Tags for the Identification of Languages, H. Alvestrand, IETF, January 2001. Available from http://www.ietf.org/rfc/rfc3066.txt.

SAM
The Standard Application Model for Topic Maps, editors G. Moore and L. M. Garshol, 2003-03-09. To be incorporated in next revision of ISO 13250. Available from http://www.isotopicmaps.org/sam/sam-model/.

tolog

tolog 0.1, Lars Marius Garshol, Ontopia Technical Report. Forthcoming. Available from http://www.ontopia.net/topicmaps/materials/tolog-spec.html.

Footnotes

1. The term "resource" is defined in RFC 2396 as *anything that has identity*.
2. In fact, the name comes from the Latin *re*, meaning thing, and means *thingification*, since to speak about the assertion we essentially turn it into a thing. McCarthy proposed the less intimidating term thingification instead, but it never caught on.
3. Hacker jargon for an awkward solution that should be made cleaner.
4. 2003-03-10
5. A fellow topic map software vendor.