SesameTM: Building Topic Maps on RDF

Arnim Bleier, Benjamin Bock, and Lutz Maicher

University of Leipzig, Faculty of Mathematics and Computer Science
Johannisgasse 26, 04103 Leipzig, Germany
{bleier,bock,maicher}@informatik.uni-leipzig.de
http://www.topicmapslab.de

Abstract. Over the past decade RDF has developed to become the dominant standard for representation and interchange of structured data on the web. In portal development, widely unrecognized by Semantic Web research, subject-centric topic maps are actively used and have evolved from an ancient SGML and intermittent XML-based standard to a pure data model. This data model can be represented as a graph and served various integration strategies, put forward over the past years, as a starting point. However, none of these strategies really appreciates the way in which the technologies are used resulting in a poor tool interoperability. To overcome this state we propose a Topic Maps engine acting as congurable wrapper for Sesame. The software library we develop and describe in this paper implements the Topic Maps Application Programming Interface (TMAPI) enabling the usage of Topic Maps infrastructure instead of working at the level of RDF triples.

Keywords: topic maps, RDF, integration, ontology mapping

1 Introduction

In the last years, a terminological shift from Semantic Web to Linked Data could be observed. This is not just a rebranding, but a step from an idea what the Semantic Web is towards the goal of connecting information. The new pragmatism frees us from ideologic constraints towards a particular technology. However, we regard the relationship between RDF and Topic Maps still as an unresolved issue in this intertwined world. The reason for bringing these two standards together is that both can complement each other serving a common purpose: fo-stering the representation and interchange of structured data. About eight years ago Moore wrote "... if we have a knowledge tool X we would expect to be able to import some XTM syntax, some RDF syntax and then run either an RDF or TMQL query in the space of tool X and expect sensible results back..." [1]. A lot of work in terms of standardization and mapping approaches has been put forward since then. The data model behind Topic Maps has been standardized and a W3C RDF/Topic Maps Interoperability Task Force[1] has been set up. However, the original goal of being able to run a SPARQL or TMQL query on one tool remains intact and unreached. In this paper we cover the implementation of a feature complete RDF-based Topic Maps engine as well as questions of ontology mapping and tool integration. We begin by distilling the

[1] http://www.w3.org/2001/sw/BestPractices/RDFTM/

essence of modern topic maps - their data model - providing a starting point for considerations towards two integration strategies. Next, we implement and integrate these strategies focusing on the practical usability of the resulting software library. The first strategy is about the development of a Topic Maps engine for writing, accessing and querying an RDF graph representing the Topic Maps Data Model. The second strategy addresses questions of ontology mapping upon reusing existing Topic Maps tools as well as extending our engine to enable \live mapping" from arbitrary RDF vocabularies. The paper concludes with remarks on the used approach.

2 Topic Maps

2.1 The Genesis

The roots of Topic Maps date back to the early 1990's when ways for interchange of electronic documentation had been discussed at the Davenport Group and the DocBook DTD had been developed as a result. A problem reaming though was the lack of rules for merging indexes of dierent documentations. The notion that indexes conform to structured models of the available information, and that a formal representation of these models can be used to facilitate the merging of these indexes. This eventually led to the idea of topics dened and described by their characteristics as well as relationships between them [2, 3]. The process of standardizing such formal models, called Topic Maps, by the International Organization for Standardization (ISO) begun 1999 in parallel but independent to the standardization of the Resource Description Framework by the World Wide Web Consortium (W3C). In 2001 Topic Maps became the ISO/IEC 13250 standard for describing real-world and web resources by the means of HyTime (HyTM) or XML (XTM) [4]. While the syntaxes initially lacked an explicit data model, the Topic Maps Data Model has later provided this.

2.2 The Data Model

The Data Model is dened in terms of the XML Information Set [5] and has the form of an entity-relationship diagram [6] with constraints and transition rules to determine how to resolve potential invalid states, called Merging Rules [7]. The diagram knows six kinds of entities or objects. The objects dier in their attributes and by which relationships they can be connected to each other. On the semantic level, entities are dened as *anything whatsoever that can be referred to in a discourse*. The objects are called constructs and divided into topic maps (as container) as well as topics, associations, roles, occurrences, names and their variants. The constraints ensure that no pair of entities exists that represents the same subject in a topic map. To guarantee this the TMDM species conditions of equivalence of entities and rules on how to merge the respective entities in case the conditions are met [7]. Such an implementation-independent denition of an abstract data structure has the advantage that Topic Maps can be represented in a variety of processors and formats.

2.3 The Object and Semantic Level

Moore pointed out that dierent levels of semantic interpretation allow RDF to be interpreted as RDF statements as well as a representation of a topic map [1]. Later on Ogievetsky, Lacher and Cregan further developed this idea [8,9,6]. It is tempting to try a roundtrip and interpret an RDF graph as a topic map and its constructs again as RDF statements. However the ontological commitments that come with the Topic Maps Data Model and its constraints limit our ability to make statements about resources used in the topic map if we want these very statements still to be a model of the TMDM. As a consequence Pepper resumes that all existing approaches fall into two distinct categories: object mapping and semantic mapping [10]. Moore described the object mapping as \modeling the map" and the semantic mapping as \mapping the model" [1].

3 An RDF-based Topic Maps Engine

3.1 Cregan's Vocabulary

Object mappings harness the ability of RDF to be used as entity-relationship model carrying a topic map. Moore has done the first object mapping; however, not building on any standardized data model (because there was none available at this time). The approaches of Ogievetsky [8] and Lacher [9] build on the (now outdated) Topic Maps Processing Model [11]. Cregan's approach is the first to build the object mapping on the standardized Topic Maps Data Model, which superseded the Processing Model. Cregan presented a construction of the TMDM as an OWL-DL ontology [6]. This ontology is a proof-of-concept of the viability of modeling Topic Maps conforming to the TMDM in RDF. Cregan's approach enables topic map authors to use OWL-DL's functionality and the respective applications supporting OWL-DL to do verication of TMDM constraints as well as querying and visualization of topic maps. Due to the ability of RDF statements to be interpreted as constituting a topic map, there is signicant overlap between the functionality provided by OWL-DL processors and Topic Maps engines. However, the previously addressed two levels of semantic interpretation cause also some issues:

- Editing topic maps in RDF always requires the awareness of the ontology, drawing attention from what is modeled to how it has to be modeled.
- Cregan's ontology only allows for the testing whether a topic map is TMDM-compliant but doesn't embrace the merging mechanisms provided by the TMDM.
- As Cregan points out, functional overlap between processors handling OWL and Topic Maps does not include exchangeability of the Topic Maps Query Language (TMQL) and SPARQL nor does it allow utilizing the Topic Maps Constraint Language (TMCL) [6].

To overcome these issues we suggest a Topic Maps engine that exposes topic maps modeled in RDF in a standardized and by the Topic Maps community accepted API.

3.2 The Topic Maps Application Programming Interface

The Topic Maps API (TMAPI) 2.0 is the second generation of an interface for handling data held in a topic map[2]. Leading open source Topic Maps engines are exposing their functionality in TMAPI, implementations of a TMCL[3] validator and a TMQL engine[4] build on it. In addition we developed the RTM[5] library reusing Java TMAPI implementations such as SesameTM to provide a high level Topic Maps API for the Ruby programming language. Consequently we consider the TMAPI as an ideal way of exposing Topic Maps functionality and rene our goal to implementing this interface to handle an RDF graph conforming to the ontology suggested by Cregan.

3.3 The Implementation

The Topic Maps engine implements the TMAPI on top of Sesame[6] and manipulates the RDF graph model of the TMDM in such a way that it is always in a legal state. The code in Figure 1 demonstrates the processor in conjunction with our RTM library by modeling a foaf:knows association between John and Mary. First, a connection to the Sesame API is set up, establishing a new topic map system. Then a new topic map is initialized. The next step is the creation of the association with the respective foaf predicate. Furthermore we add to the association Mary and John as players of the role foaf:Person. In a last step we assign to the topic representing John the name "Johnson", typed by foaf:family name.

```
topic_map_system = RTM.connect(:backend => :sesametm)
tm = topic_map_system.create("http://www.ex.org/tm")
john = tm.get!("http://www.ex.org/tm/John")
tm.create_association("http://xmlns.com/foaf/0.1/knows",
    "http://xmlns.com/foaf/0.1/Person" => [john,
                              "http://www.ex.org/tm/Mary"])
john["-http://xmlns.com/foaf/0.1/family_name"] = "Johnson"
```

Fig. 1. RTM Code.

4 Ontology Mapping

4.1 The need and challenge

The object mapping used by the part of the engine introduced so far results in an RDF graph representing the TMDM. While this allows a complete representation of all aspects of Topic Maps as demanded by Pepper [10], the resulting RDF doesn't correspond to the schemas in which information usually is expressed in RDF. The consequence is that information conforming to standard vocabularies, such as FOAF, DC, or SIOC rst needs to be mapped in order to be

[2] http://www.tmapi.org/2.0/
[3] http://code.google.com/p/tmcl-validator/
[4] http://tmql4j.topicmapslab.de/
[5] http://rubygems.org/gems/rtm
[6] http://www.openrdf.org/

used by our RDF Topic Maps engine SesameTM. The main obstacle for mapping these lightweight ontologies to the relatively verbose vocabulary of the TMDM is that a single RDF statement has to be transformed to up to three entities, each to be represented by multiple statements. One way to achieve such a mapping would be to use SPARQL CONSTRUCT queries, with their heads in the source schema and their tails conforming to the respective constructs in the Cregan vocabulary. While in principle possible, writing these rules turns out to be very complex and requires writing a dierent set of rules for each schema to be transformed [12].

4.2 The RDF2TM Vocabulary

Another way is to use the congurable RDF deserializer TMAPIX[7], that translates a stream of supplied data into calls to the TMAPI creating an internal representation thereof. TMAPIX is congured using the RDF to Topic Maps vocabulary (RDF2TM) introduced by Garshol [13]. Its goal is enabling a semantic mapping between Topic Maps and RDF. In contrast to the object mapping previously addressed, semantic mapping aims at preserving the way in which information would naturally by expressed [10]. The vocabulary tries to solve the problem that the RDF statements should be mapped to occurrences, names and associations. A remaining question is: which RDF predicates should be mapped to which kind of Topic Maps construct. Garshol lines out that additional map-ping information is required. His mapping vocabulary RDF2TM is a mean to provide this mapping information. The central predicate provided by the vocabulary is rdf2tm:maps-to[8]. It is used to describe which RDF predicate maps towhich construct in a topic map. The domain of rdf2tm:maps-to is the predi-cate described, the range is the respective RDF2TM construct. The properties declared to be mapped to rdf2tm:association also require the roletypes of the subject and the object. These roletypes are declared by rdf2tm:subject-role and rdf2tm:object-role respectively.

```
rdf:type rdf2tm:maps-to rdf2tm:instance-of.
foaf:knows rdf2tm:maps-to rdf2tm:association;
     rdf2tm:subject-role foaf:Person;
     rdf2tm:object-role foaf:Person.
foaf:family_name rdf2tm:maps-to rdf2tm:name.
```

Fig. 2. RDF2TM mapping.

4.3 Live Mapping

Both approaches are a form of forward chaining in which the transformation happens before the information is needed and everything is transformed as long a mapping is dened for it. While in theory they are capable of providing a Topic Maps representation of available RDF graphs (given the necessary vocabularies are dened), it is hard to imagine that such a mirroring would happen at a large scale, and the duplication of information is only partially advisable. Another

[7] http://code.google.com/p/tmapix/
[8] The prex rdf2tm is used to signify the namespace http://psi.ontopia.net/rdf2tm/#

approach is to do a dynamic mapping. This is done only for the parts of the RDF graph that are actually needed and when they are needed. Such a *live mapping* has the advantage that while it gives a Topic Maps view on all available information only those parts that are needed have to be converted. Consequently, we decided to extend our Topic Maps engine to provide a live view on RDF. To allow this Topic Maps view on data conforming to a wide range of ontologies, we implemented a live mapping Topic Maps engine, congurable by the RDF2TM vocabulary discussed above. The live mapping works essentially in three steps: First, in case a getter function is called the engine looks up in a mapping vocabulary which RDF properties map to the object(s) the getter returns. In the second step, a SPARQL query string is built by inserting the respective properties in a template and evaluated against an RDF repository. In the last step, the query result is provided as TMAPI objects and returned by the function[9]. The TMQL implementation TMQL4J implements the current draft [14] of the query language. Since TMQL4J is built on the TMAPI, we can use this query processor to deliver on the original goal of running a TMQL query against an RDF repository. Before we can evaluate TMQL queries on an RDF repository containing a graph conforming to an everyday vocabulary, we have to congure our engine. The required code is similar to a standard TMAPI setup; however, setting the TopicMapSystemFactory property MAPPING to an InputStream provides the necessary RDF2TM mapping vocabulary.

```
factory.setProperty(
    PROPERTY.MAPPING, mappingVocabularyInputStream);
```

Furthermore this time the topic map is initialized upon getTopicMap with the RDF graph to be browsed as topic map in the argument.

```
TopicMap tm = tmSys.getTopicMap("http://www.ex.org/tm");
```

After these initial steps we can now use TMQL4J in conjunction with our engine to evaluate a Topic Maps Query Language query against an RDF repository. Please consider the following TMQL query string as an example.

```
select $p / name
where
    $p isa foaf:Person
```

In case the topic map is initialized on a FOAF graph and the RDF2TM vocabulary is set to the one in example 2 the query expression will return the all foaf family names.

5 Conclusion and Remarks

In this paper we addressed the interoperability of RDF and Topic Maps. The review of literature on this topic yielded two dierent strategies, responding to the demands of complete and natural representation of Topic Maps. We followed both of them, focusing on closing the open gaps when necessary without reinventing the wheel. We developed a Topic Maps engine reusing Cregan's work on vocabularies for complete representation of Topic Maps. Later we

[9] Being more precise, in most cases this step is repeated at least once to extend the query for those resources that are to be merged with the initial query result according to the TMDM equality rules [7].

have drawn our attention to live mapping of RDF to Topic Maps, enabling a natural Topic Maps view on RDF graphs conforming to everyday lightweight ontologies. For this we reused the existing vocabulary RDF2TM to congure the mapping, liberating the user from introducing new ways of capturing the necessary mapping information. On the downside of the rst presented approach is that the complete representation of Topic Maps as an RDF graph is quite verbose and does not correspond to our notion what a typical graph looks like. The vocabulary used for the live mapping is straight forward in its usage with lightweight ontologies; however, this comes at the cost of making it dicult to map less frequently used constructs such as reication. Since we expressed the functionality of our Topic Maps engine in a widely used API we were able to deliver on the goal of running TMQL queries against an RDF repository. While through the paper we used a very simplistic FOAF example covering a view on it from dierent angels, we tried to implement the approach as robustly as possible. To grant this, more than 100 JUnit test have been written in addition to the 280 tests that come along with the TMAPI test suite as well as over 1000 RSpec examples that have been developed as part of our RTM project. Moreover we published the library[10] under an open source license and maintain it in active developed. An interesting option for further development of the used integration strategy would be to circumvent TMAPI for evaluating TMQL and follow a straight query translation approach. This technique could also have a positive impact on performance, especially for complex queries going beyond the TMAPI.

References

1. Moore, G.: RDF and Topic Maps: An Exercise in Convergence. In Proceedings of XML Europe (2001)
2. Pepper, S.: Euler, topic maps, and revolution. In Proceedings of XML Europe (1999)
3. Pepper, S.: The TAO of Topic Maps. In Proceedings of XML Europe (2000)
4. Biezunski, M., Newcomb, S., Bryan, M.: Guide to the topic map standards. ISO SC34, http://www1.y12.doe.gov/capabilities/sgml/sc34/document/0323.htm
5. Cowan, J., Tobin, R.: XML information set. W3C Recommendation (2001), http://www.w3.org/TR/xml-infoset/
6. Cregan, A.: Building Topic Maps in OWL-DL. In Proceedings of Extreme Markup Languages (2005)
7. ISO/IEC IS 13250-2:2006: Information Technology - Document Description and Processing Languages - Topic Maps - Data Model. International Organization for Standardization, Geneva, Switzerland, http://www.isotopicmaps.org/sam/sam-model/
8. Ogievetsky, N.: XML topic maps through RDF glasses. In: Markup Languages, vol. 3, pp. 333 { 364. MIT Press, Cambridge (2001)
9. Lacher, M., Decker, S.: On the Integration of Topic Maps and RDF Data. In Proceedings of Extreme Markup Languages (2001)
10. Pepper, S., Vitali, F., Garshol, L.M., Gessa, N., Presutti,V.: A Survey of RDF/Topic Maps Interoperability Proposals (2006). W3C Working Group Note, http://www.w3.org/TR/rdftm-survey/
11. Newcomb, S., Biezunski, B.: Topicmaps Processing Model (2001), http://topicmaps.net/pmtm4.htm

[10] http://code.google.com/p/sesametm/

12. Euzenat, J., Polleres, A., Schare, F.: Processing Ontology Alignments with SPARQL. In: Complex, Intelligent and Software Intensive Systems, pp. 913-917 2008. CISIS, Barcelona (2008)
13. Garshol, L.M.: Living with topic maps and RDF, Online only (2003)
14. ISO/IEC WD 18048::2008: Topic Maps Query Language. International Organization for Standardization, Geneva, Switzerland, http://www.isotopicmaps.org/tmql/tmql.html