Extending MIDST To Semantic Annotation

Pierluigi Del Nostro
Università Roma Tre, Italy
pdn@dia.uniroma3.it

Stefano Paolozzi
Università Roma Tre, Italy
stefano.paolozzi@irpps.cnr.it

Paolo Atzeni
Università Roma Tre, Italy
atzeni@dia.uniroma3.it

Abstract

Semantic annotation remains a significant part of the Semantic Web activities and a number of formalisms have been employed to annotate. Due to the impossibility of having a unique standard it may be important to have some techniques that allow the interoperability among the different kinds of annotations. We present an extension of our framework MIDST (Model Independent Schema and Data Translation), in order to translate from one semantic annotation formalism to another, where translations are specified as Datalog programs. To exploit our approach in the Semantic Web context, we have considered two formalisms for whom interoperability issues have been often addressed by the research community, namely RDF and Topic Maps.

1. Introduction

The translation of schemas from a data model to another has been studied for many years [1, 11]. An ambitious goal is to consider translations in a model generic setting [4, 5] where the major problem can be formulated as follows: given two data models $M_1$ and $M_2$ (from a set of models of interest) and a schema $S_1$ of $M_1$, translate $S_1$ into a schema $S_2$ of $M_2$ that properly represents $S_1$.

In this paper, we propose our approach to the management of interoperability of semantic annotation formalisms, based on model management techniques [5] with reference to a recent approach to this problem, the MIDST proposal [2]. MIDST assumes that there is a set of generic constructs, each with a number of possible variations. Constructs with the "same" meaning in different models are defined in terms of the same generic construct (called meta-construct); for example, "entity" in a Entity-Relationship model and "class" in an object model both correspond to the abstract construct. Then, a data model is defined by specifying the constructs it includes.

The paper is structured as follows. In Section 2 we briefly describe the context showing a motivating example, in section 3 we describe the main features of our approach and in section 4 we explain the proposed extension to manage interoperability of different semantic annotations formalisms. In section 5 we present an example of transformation between RDF and Topic Maps and vice versa. In section 6 concludes and illustrate our future works.

2. Semantic Annotations

Semantic annotation is the creation of metadata and relations between them with the task of defining new methods of access to information and enriching the potentialities of the ones already existent.

The main goal is to have information on the Web, defined in such a way that its meaning could be explicitly interpreted also by automatic systems, not just by humans. To reach such a target, it is necessary to associate metadata with Web resources through semantic annotations. In particular, we want to annotate the resources with semantic information which give indications about the content of the resource itself.

A variety of standards to represent semantic annotation have been proposed from the less formal (such as unstructured metadata) to the more formal (e.g. ontologies). An important issue to consider is the interoperability between different standards. The possibility of changing between a semantic annotation standard can be useful in order to take advantage from already defined annotations.

2.1 A Motivating Example: RDF vs Topic Maps

The two formalisms we have chosen to consider in this paper are RDF and Topic Maps since they are commonly used in the Semantic Web context and a lot of work about the interoperability between them have been done [8]. In this section we will give a brief description of both models, underlying which are the main differences and the matching points. Topic Maps and RDF have been conceived as a means to describe information resources with some different perspectives. Topic Maps are used to support high-level indexing to let information be easily findable for humans.
3 MIDST: A generic approach to interoperability

MIDST (Model Independent Data and Schema Translation) has been developed in order to handle the translation of schemas and instances from a data model to another [2, 3]. MIDST is based on the concept of supermodel, that is a model within which “every” model is defined. Indeed each model can be described as a set of metaconstructs belonging to a higher level model called metamodel. A metamodel is a set of constructs, called metaconstructs, which can be used in model definition. The approach is based on Hull and King observation [9] that the constructs used in most known models can be expressed by a limited set of generic (i.e. model-independent) metaconstructs: lexical, abstract, aggregation, generalization, function.

MIDST supermodel virtually contains all constructs of each model. Therefore, as each model is a specialization of the supermodel, each schema also belongs to the supermodel. In other words the supermodel is like a pivot model, so the translation from a source scheme $S_1$ of a model $M_1$ to a target schema $S_2$ of a model $M_2$ can be seen as a translation between metaconstructs of the same metamodel (i.e. the supermodel). With this approach the number of translations is reduced to a linear number with respect to the total number of the models recognized by the system. Indeed the only needed translations are those within the supermodel with reference to the target models; a translation is performed by eliminating constructs not allowed in the target model, and possibly introducing new constructs.

MIDST tool is based on a relational dictionary which contains the needed metadata for the translation (i.e. the supermodel, the models and the schemas). The dictionary is constituted by four parts, namely:

- **MetaSuperModel** that describes the structure of the needed metaconstructs
- **Supermodel** that contains the schemas which must be translated
- **Metamodels** that describe the constructs of every models of interest; each construct is associated with the corresponding one in the supermodel
- **Models** that contains the associated schemas

The translation technique employed in MIDST is based on the property stating that a schema of a model is also a schema of the supermodel, in other words translations between schemas are performed within the supermodel: every time we want a translation from one model to another the first operation is the translation of the constructs of the source model in term of supermodel constructs. The translations are performed through a convenient set of rules. These
rules are defined in a Datalog-like formalism with OID invention. Therefore we would need one translation for each model (from the supermodel to it), but with many different models to manage this would be difficult. In order to overcome this problem, MIDST (following MDM [4], and concurrently with other proposals (see for example [6]) has complex translations that are built as composition of elementary ones. The idea is to have “basic” translations that perform elementary steps that refer to generic constructs, and so can be reused.

The MIDST framework can be suitably extended to solve the interoperability problem between semantic Web languages in different domains such as semantic annotations frameworks (see [12] for a preliminary work).

4 Extending MIDST to Semantic Annotations

The supermodel has been defined with respect to a variety of models for which translations can be applied by our approach (E-R, Relational, Object-Oriented, Object-Relational, XSD). To give an idea of how the supermodel is made we list some of the main constructs that are used to model concepts and relationship between them:

Abstract (OID, Name)

BinaryAggregationOfAbstracts (OID, Name, Abstract1OID, Abstract2OID)

AggregationOfAbstracts (OID, Name)

ComponentOfAggregationOfAbstract (OID, Name, isOptional, isFunctional, isIdentified, AbstractOID, AggregationOfAbstractsOID)

The Abstract metaconstruct is used to represent abstract concepts like entities of the E-R model or classes of the OO model. To relate concepts with a binary relationship there is the BinaryAggregationOfAbstracts term while the AggregationOfAbstracts is used to describe n-ary relationships where each participant is a ComponentOfAggregationOfAbstract. The supermodel, as it is, isn’t enough expressive to describe Semantic Web formalisms, where there are some specificities that distinguish them from the data models we have involved till now in our approach. One of the most interesting characteristic to take into account is the generality of some of the constructs that are used in Semantic Web models. Looking specifically at the two formalisms we consider, in RDF there is the concept of resource that can be a web resource, a predicate, a specific person, the concept of person and all the things that can be described. It is the same for Topic Maps where (quite) everything is a topic like the described subjects, the types, the association roles etc. Data models that we have considered till now, have instead a well marked distinction between the different layers: metamodel, model, schema, data. For example, in the database context the Abstract metaconstruct is used to represent the E-R model construct Entity that can define the Schema element Person and at the instance layer we have a particular person.

To enable the supermodel representing RDF and Topic Maps, reflecting the above mentioned peculiarities, we have enriched some of the existing metacoructs with Semantic Web specific properties and other new metacoructs are introduced. The meaning of Abstract is extended to represent symbols that are used to describe the knowledge of interest in RDF and Topic Maps. With the term Abstract we generically describe the nodes of an RDF graph as well as the topics of a Topic Map. To specify the node (topic) properties, as if it would be a resource with an URI or a Literal, we use the Lexical construct. This allows us to define a generic transformation between the structure of a graph and a map keeping the particular resources separated from the concept of resource. Lexicals are also used to express the different kinds of identification that Topic Maps allow, namely SubjectIndicator and SubjectAddress. Examining the way RDF and Topic Maps relate concepts, we have noticed that they can basically fall into already defined metacoructs with some extensions. With BinaryAggregationOfAbstracts we can describe the Topic Maps Occurrence and Name relationships extending it with an optional reference to an abstract that represents the scope that we generically call Qualification. The same construct can describe the RDF Statement, for this purpose we have introduced the isDirected property that allows the representation of RDF directed arcs. The AggregationOfAbstracts, instead, has been extended with two optional references to abstracts that represent the type and the qualification of the aggregation, this is used for the n-ary Association of Topic Maps. The type and qualification are expressed as abstracts since they are topics. Each ComponentOfAggregationOfAbstract of the extended supermodel has a reference to an abstract that is the role of the participation, a topic itself. The introduction of the new metacoruct Type, which expresses the typing relation between abstracts, allows us to model the RDF construct rdf:type and the equivalent for Topic Maps that is instanceOf. We have provided the supermodel with these new constructs through which we can now illustrate how to perform a translation between Topic Maps and RDF.

5 Translations between RDF and Topic Maps

As illustrated in [8], there are many ways to implement interoperability between Topic Maps and RDF: creating vir-
tual views that make RDF appear to be Topic Maps (or vice versa) when viewed through a particular interface or doing a conversion of data from one formalism to the other by means of a somehow expressed mapping. It is quite easy to model RDF in topic maps and many works have been written about modeling topic maps in RDF (see for example [10] and [7]).

Devising a generic mapping can sound as the ideal way to perform translations, avoiding to force a model to behave as the other, implementing translations in a more natural way. But when we create the mappings between RDF and Topic Maps, it arises the problem of a lack of information because of the different expressivity of the two models. Let’s say that we want to translate the RDF statement \( <S,P,O> \) in Topic Maps. It is an assertion that can be translated in an Association, an Occurrence or a Name and it cannot be known a priori which is the right choice. We need to evaluate the meaning of the property \( P \) to select which is the right target construct. The result is a vocabulary-specific mapping that nevertheless allows us to reach satisfiable results in terms of keeping the expressiveness of the model and its meaning during the translation process.

As we demonstrate with the example below, the use of Datalog to write translation rules allows us to naturally implement the vocabulary-specific mappings by means of conditional expressions that select the correct target construct. Moreover rules are written at the metalevel between the representations of source and target model in terms of metaconstructs, keeping the independence from a given model and allowing MIDST to involve new models reusing the previously made work.

As illustrated in section 3, the first step when translating with MIDST is to describe the source and target model’s constructs in terms of the supermodel metaconstructs, then to define the source schema that can be imported into the supermodel by a copy operation. Following the correspondences presented in section 4 we can represent both RDF and Topic Maps using the supermodel and write the translation rules at the metalevel.

Let us consider the simple RDF document that follows that illustrate the knowledge between two persons one of which with a homepage, using the foaf vocabulary.

```xml
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
><foaf:Person>
  <foaf:name>John Smith</foaf:name>
  <foaf:homepage rdf:resource="http://www.john.sm"/>
  <foaf:knows>
    <foaf:Person>
      <foaf:name>Fred Red</foaf:name>
    </foaf:Person>
  </foaf:knows>
</foaf:Person>
</rdf:RDF>
```

The supermodel representation of the above document, stored in our relational dictionary, is illustrated in Fig. 1.

![Figure 1. Supermodel representing an RDF document](image)

It can be noticed that while in the Statement table there is the specification of the actual names of the properties, the Abstract table stores generic RDF nodes and the Lexical table contains generic literals or resources. The values of the literals and URIs are not relevant during the translation and are represented at the instance level while the properties names are relevant for the translation at the schema level.

When translating the RDF Statement into Topic Maps, the Datalog rule generate the construct Name for the property foaf:name, the construct Occurrence for the property foaf:homepage and the construct Association for the property foaf:knows. For the last case we don’t have the information of the association roles in RDF while Topic Maps misses the direction of the statement. We choose to assign the roles values of rdf:subject and rdf:object to the association participants, in order to keep the semantic of the direction of the RDF arc. As result of the translation we obtain the following Topic Maps, the syntax we use is XTM:

```xml
<topic id="id18">
  <instanceOf>
    <subjectIndicatorRef xlink:href="#foaf:Person"/>
  </instanceOf>
  <baseName>
    <baseNameString>John Smith</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf>
      <subjectIndicatorRef xlink:href="#foaf:homepage"/>
    </instanceOf>
    <resourceRef xlink:href="http://www.john.sm"/>
  </occurrence>
</topic>
<topic id="id32">
  <instanceOf>
    <subjectIndicatorRef xlink:href="#foaf:Person"/>
  </instanceOf>
  <baseName>
    <baseNameString>Fred Red</baseNameString>
  </baseName>
</topic>
```
On the other side, considering the translation from Topic Maps to RDF, specifically the case of transforming an association, we have to transform n-ary relationships to binary statements. In Fig. 2 there is a topic map that represents the employment association between the company HiTech, with the role of employer and two employees.

**Figure 2. Topic Maps example**

To translate this map to RDF, we define a Datalog rule that creates an Abstract for each topic, with lexicals that represent the identification and the names. Other simple rules are devoted to the creation of an Abstract without lexicals (blank node) for each participant, that allows us to define BinaryAggregationOfAbstracts that link the participant topic with the role topic. We then create an Abstract for the association and an Abstract for the type that are linked by a Type meta-construct. Finally members are linked to the association by BinaryAggregationOfAbstracts. Exploiting the correspondence between the supermodel and RDF we can eventually generate the graph as illustrated in Fig. 3.

**Figure 3. Resulting RDF graph**

### 6 Conclusions and future works

We have illustrated how Semantic Web formalisms can be described by our Supermodel and how our framework can be used to define translations with a high-level, model independent approach. Data models and schemas are currently defined using the framework interface, we are developing a plug-in that allows to automatically import external documents of the source model inside the tool and to export the result of the translation in the target model. Moreover, we are developing a visual interface that allows users to query documents exploiting our logical organization.

### References