A Language to Specify Mappings Between Ontologies

François Scharffe, Jos de Bruijn
Digital Enterprise Research Institute
University of Innsbruck, Austria
Email: francois.scharffe@deri.org, jos.debruijn@deri.org

Abstract— **The ontology mediation field aims at finding techniques and frameworks to allow interoperability between overlapping heterogeneous ontologies. One of the solutions is to designing mappings that link the corresponding entities. Many systems and algorithms have been built by different parties, but each system represents the mappings in its own format. To allow reusability of the mapping tools and of the results of the algorithms, we have designed a language to express these mappings. In this paper, we will present this mapping language.**

I. INTRODUCTION

Ontologies are the most prominent solution to integrate heterogeneous data sources. As structured objects they allow for a formal description of a domain. A domain consists of different data sources being linked to an ontology to enable interoperability between them. A same domain may be modelled differently depending on the contexts of use and interest. Ontology mediation is the research topic that aims to relate ontologies having an overlapping part to enable interoperability between them. In this paper we describe a language we have developed to express correspondences between ontologies. Such correspondences are then accessible in a mapping document which may be used to perform the mediation tasks.

The recommended Ontology Web Language OWL [1] gives a few constructs to express these mappings. These constructs are included as part of the ontology and coupled with it. In our approach we separate the mappings from the ontology. This allows us to define different mappings that may be partial or specific to a given context.

Different tools are already contributing to the mapping task; either by introducing different algorithms to automatize or semi-automatize the task [2],[3] or by providing a graphical user interface to relate the corresponding entities [4]. Current approaches often offer both functionalities. However, the tools and algorithms are generating mappings based on different formats. The reusability of these mappings is then limited. For example the mappings resulting of the Prompt algorithm and those generated by Euzenat's API are expressed differently. A common language to specify the mappings will facilitate their use.

In the following we describe the requirements for a language to express mappings between ontologies. We then give the specification of such a language we have designed based on these requirement. Finally we briefly present a Java API and its implementation as tools to manipulate this language and to build applications upon it.

II. REQUIREMENTS

We define an ontology O as a tuple $\langle C, R, I, A \rangle$ where C is a set of concepts, R is a set of relations, I is a set of instances and A is a set of axioms. This notion of an ontology is similar to the one in OKBC [5]. In the following we will call a *mapping rule* a single correspondence between a set of ontological entities, a *mapping* being the set of mapping rules between a source and a target ontologies.

An *ontology mapping* M is a (declarative) specification of the semantic overlap between two ontologies O_S the source and O_T the targeet. This mapping can be either unidirectional or bidirectional. In a unidirectional mapping we specify how to map terms in O_T using terms from O_S in a way that is not easily invertible. A bidirectional mapping works both ways, i.e. a term in O_T is expressed using terms of O_S and vice-versa.

A mapping language has to solve the mismatches that may happen when trying to align two ontologies. These requirements are based on an extensive survey of these mismatches [6]. We restrict the scope of these requirements to ontologies written in the same ontology language.

When mapping between two ontologies, one has to specify the correspondences between each type of entities of the ontology schema. Namely between the concepts, attributes and relations defined in the ontology. Simple mappings align one entity to another. for example a simple mapping will stand between the concept 'car' in an ontology O_S and the concept 'automobile' in an ontology O_T. The expression of such a simple mapping is given in an example later in this paper. The difference of granularity in the ontological descriptions as well as different understandings of the domain require more complex mappings than one to one. For example an ontology O_S may have the 'car' concept as a leaf in the concept hierarchy, while the ontology O_T is more "fine-grain" described and adds the subconcepts *normal-car* and *sport-car* under the *car* concept. In that case, the instances of *normal-car* and *sport-car* in O_T are both mapped under the concept *car* of O_S. Conversely, the instances of the *car* concept in O_S may be distributed between the two subconcepts *normal-car* and *sport-car* in O_S. A mapping language must then support mappings between both union and intersection of entities. Also, it may

happen that some mappings between classes are only effective considering the values of their attributes. For example a *car* is a *sport-car* if its *maxspeed* attribute value is greater than or equal to 200 Km/h and its *0-100time* attribute value (the time needed to go from 0Km/h to 100Km/h) is lower than 6 seconds. These conditional mappings must then be taken into account by the mapping language. It is also the case that some entities may be related but not strictly equivalent, ie one may be more general than the other. A mapping may be valid from the source ontology to the target but not from the target to the source. Thereby, directionality must be reflected by the language.

Such a language is used as the basis to write mappings. The mappings may be directly written by a domain expert using a text editor, or they may also be the result of an algorithm, or graphical tool. In such a context, the mapping specification must be easily human readable and its constructs intuitive to express. In addition it must be easy to include in an application (output of an algorithm or graphical tool). In the case the that mappings are produced as the result of an algorithm, the output may come with additional information about the nature of the generated mappings. For example, the alignment API [3] gives information about the algorithm used, the confidence level of each generated mapping rule and the nature of the relation between the mapped entities.

In order to be processed to realize the mediation task (query rewriting, instance transformation and unification) the mappings must be grounded to the formalisms in which the source and target ontologies are designed. Different formalisms are used apart from OWL , the Web Service Modeling Language WSML[1], used to design the Web Service Modeling Ontology WSMO [7] being one of them.

We base the definition of our language on this minimal set of requirements, adding useful features.

III. MAPPING SPECIFICATION LANGUAGE

The requirements presented above are provide the fundation to design a full-fledged language to support ontology mapping. In the following we present an overview of the syntax as well as a description of the API designed to manipulate the mapping constructs.

A. The Language

The reference of this language as well as its specification in Extended Backus-Naur Form (EBNF) are available on the Ontology Management Working Group web site http://www.omwg.org.

Answering to the requirements in the last section, the language contains constructs to express mappings between the different entities of two ontologies: from classes to classes, attributes to attributes, relations to relations, but also between any combination of entities like classes to attributes, etc.

[1]Informations about WSML is available at http://www.wsmo.org/wsml/wsml-syntax

Language Construct	Description
ClassMapping	Mapping between two classes
AttributeMapping	Mapping between two attributes
RelationMapping	Mapping between two relations
ClassAttributeMapping	Mapping between a class and an attribute
ClassRelationMapping	Mapping between a class and a relation
ClassInstanceMapping	Mapping between a class and an instance
IndividualMapping	Mapping between two instances

A set of operators associated to each type of entity give the possibility to combine them. The following tab represent the different operators for each type of entity.

Entity	Operator
Class	and, or, not, join
Attribute	and, or, not, inverse, symetric, reflexive, transitive closure, join
Relation	and, or, not, join

Each operator has a cardinality, an effect and some related semantics. The semantics are related to the logical formalism used to represent the ontologies. For example the semantics or the 'and' operator between two classes is linked to the semantic of 'and' in OWL if the mappings are grounded to OWL.

We model the conditions on which the mappings are valid by introducing a conditional field in the mapping rules. These conditions may be a class condition or an attribute condition. The class conditions are based on their nested attributes values,occurences or types while the attribute condition are based on their own values or types. We give in the following table the different conditions the mapping language can express.

Range	Name
Class conditions	attributeValueCondition
	attributeTypeCondition
	attributeOccurenceCondition
Attribute Conditions	valueCondition
	typeCondition

To get a clear but expressive language we define limited constructs for the most common cases of mappings, allowing the user to define arbitrary logical expressions to represent those which do not have constructs. These logical expression must be written according to the two ontology modelling language.

The syntax of this language has been designed to be intuitive and human readable. This results in a verbose syntax far from the often used XML syntaxes. We however plan to write XML and rdf syntaxes and to provide mappings between them. The language comes with a Java API that provides parsing and serializing methods to and from an object model of the mapping document. The following figure presents two simple ontologies representing the same domain but with a different modelling perspective. We will give the mapping having for a source the 'Living Thing' ontology and for a target the

'Creature' ontology.
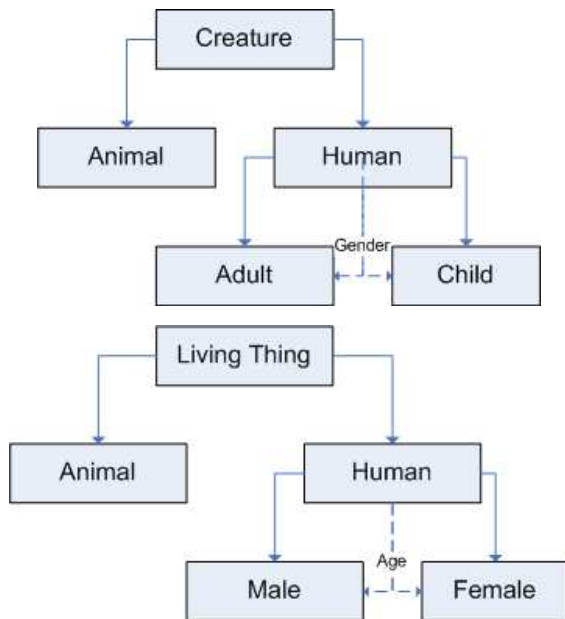
The top concepts 'living_thing' and 'creature' are presenting a terminological mismatch of synonymy. On both ontologies the concepts 'human' and 'animal' are modelled using the same label. These three cases are simple class to class mappings expressed in the mapping language. Here are the statements representing these mappings.

```
classMapping(
  annotation(<"rdfs:label">
             'Creature to LivingThing')
  annotation(<"http://purl.org/dc/
             elements/1.1/description">
    'Map the person concept to
             the livingThing concept')
  bidirectional
  <"http://ontologies.omwg.org/
          creature#creature">
  <"http://ontologies.omwg.org/
          livingThing#livingThing">)

classMapping(
  annotation(<"rdfs:label">
             'Animal to Animal')
  bidirectional
  <"http://ontologies.omwg.org/
          creature#animal">
  <"http://ontologies.omwg.org/
          livingThing#animal">)

classMapping(
  annotation(<"rdfs:label">
             'human to human')
  bidirectional
  <"http://ontologies.omwg.org/
```

```
          creature#human">
  <"http://ontologies.omwg.org/
          livingThing#human">)
```

The annotation fields allow the input of annotations, for instance, title or description. This field is also used when the mappings are resulting from an algorithm, whereby the information like the confidence degree of the mapping and the algorithm used are here stated. In our example we use rdfs and Dublin Core namespace to indicate the nature of the descriptions. Another field express the directionality of the mappings. We consider by default a mapping as bidirectional, meaning that the source and target entities are equivalent. It may also be unidirectional, meaning that the target entity somehow subsumes the source one.

The complexities come when mapping the 'male' and 'female' concepts in the 'living Thing' ontology to the subconcepts of 'human', namely 'adult' and 'child' in the 'creature' ontology. A human male or female is an adult/child if his or her age is greater than or equal to/lower than 18. This kind of mapping is represented using a condition. The concepts are considered mapped only if the condition specified in the mapping rule is valid. Following is the representation of such a condition for this example.

```
classMapping(
  annotation(<"rdfs:label">
             'conditional female to adult')
  unidirectional
  <"http://ontologies.omwg.org/
          creature#female">
  <"http://ontologies.omwg.org/
          livingThing#adult">
  attributeValuecondition(
      <"http://ontologies.omwg.org/
             creature#age '>=18'))

classMapping(
  annotation(<"rdfs:label">
             'conditional female to child')
  unidirectional
  <"http://ontologies.omwg.org/
          creature#female">
  <"http://ontologies.omwg.org/
          livingThing#child">
  attributeValuecondition(
       <"http://ontologies.omwg.org/
              creature#age
      '<18'))
```

The same rules must then be written for the 'male' concept in order to realize a complete mapping. A mapping between the female/male concepts in the source ontology and the female/male gender attribute in the target one may also be created. This kind of mapping is saying: "The instances of the

female concept in the source ontology are equivalent to the instances having a gender attribute with the value 'female' in the target ontology". Here is the representation in terms of the mapping language.

```
classAttributeMapping(
  annotation(<"rdfs:label">
      'map female to gender:female')
  unidirectional
  <"http://ontologies.omwg.org/
          creature#female">
  <"http://ontologies.omwg.org/
          livingThing#gender:female">)

classAttributeMapping(
  annotation(<"rdfs:label">
      'map the male to the gender:male')
  unidirectional
  <"http://ontologies.omwg.org/
          creature#male">
  <"http://ontologies.omwg.org/
          livingThing#gender:male">)
```

B. The Programming Interface

When designing a language, different solutions on the syntax are offered. The current popular way is to base the syntax on a well-formed modeling language like XML, the structure being based on an XML schema. Using such a model allows the advantage that different tool are provided to parse and serialize the constructs defined. However, we have stated as a requirement that the user must be able to easily type and read mappings. This requires a human readable syntax, which is not really the case of XML syntax. We express the constructs of our language using an abstract syntax based on an EBNF grammar definition. In order to assist the tool developper who wants to deal with the mapping language in its application, we provide an Java API offering the following functionalities:

- **Parsing the mapping documents.** The parser has been written using SableCC and populates an object model representing the different expressions of the language. These expressions may the easily accessed via a set of classical methods.
- **Serializing the mappings.** The mappings in the model may currently be serialized in the mapping language abstract/human-readable syntax, and in the Web Services Modeling Language abstract/human-readable syntax. Our next steps are to propose an rdf/xml export for the language, as well as export methods to other ontology representation languages such as OWL and Flora.

The API is freely available under the licence of the Distributed Ontology Management Environment at the following address: http://www.omwg.org/tools.html

IV. FUTURE WORK

We are currently working on further improvements to obtain a workable language with enough expressivity to represent the complexity of some correspondences between two different ontologies. Our future work will consist of the following:

A. Relating Axioms

We have not yet considered relating axioms in different ontologies to each other. We believe that this would not occur very often in an ontology mapping scenario. However, in an ontology merging scenario where certain constraints must be merged, we predict the necessity of relating axioms.

B. Aligning Ontologies Written in Different Languages

Aligning two ontologies written in different languages may be possible at a certain level. Depending on their degree of expressivity.[8],[9]. Information integration frameworks based on the mapping of different logical languages is nowadays a prominent research area which attempt to solve the problem of integrating different modeling paradigms having different semantics. It is however possible to perform translations between languages at the ontology schema level. These translations are purely syntactic and are expressed using transformation languages like XSLT.

C. Extend the Mapping Language to Deal with Ontology Evolution

Ontologies are dynamic objects that evolve through time, following the domain they model. An important aspect of ontology management consists in keeping track of the evolutions of an ontology to allow backward compatibility between the different versions. Expressing the differences between two versions of an ontology means expressing the different conceptualization between them. In that scope, an ontology evolution specification may be seen as a mapping betwee two versions of the same ontology. We plan in our future work to extend the mapping language, adding constructs to relect the evolution of two given versions of the same ontology.

D. Functional mappings

As already stressed, some complex mappings may require the use of aggregate functions to reflect the modeling differences. These functions will not only be used to convert units, currenceis and measurements from one ontology to the other, but also to perform string manipulations. For example splitting a string into two parts and thus creating two instances from one, or merging two strings to create a new instance from two given ones. These functions are static in the string manipulation case but may also have to be dynamic in the currency transformation case. The mapping language must be able to express or invoke such functions when necessary.

E. XML and RDF syntaxes

The current syntax is verbose and human-readable. A special parser has then been defined for it. We are currently writing a XML syntax and will begin to write a RDF one in the near future, these parsers are generic for these syntaxes, so one is not obliged to use our API to deal with the mapping language but can use its own parser.

V. CONCLUSION

In this paper we have presented requirements for an ontology mapping specification language for the Semantic Web. We have presented the language we developed following these requirements and the API needed to manipulate its constructs. We have then given future directions for the extension of this language in order to cope with all the parts of the ontology mapping task. This language is currently being used by different mapping tools as part of the SDK European projects cluster. Its specification and the associated Java API may be found on the Ontology Management Working Group web site: http://www.omwg.org/tools/

VI. ACKNOWLEDGEMENTS

REFERENCES

[1] M. Dean and G. Schreiber, Eds., *OWL Web Ontology Language Reference*, 2004, w3C Recommendation 10 February 2004.
[2] N. F. Noy and M. A. Musen, "Prompt: Algorithm and tool for automated ontology merging and alignment," in *Proc. 17th Natl. Conf. On Artificial Intelligence (AAAI2000)*, Austin, Texas, USA, July/August 2000.
[3] J. Euzenat, "An api for ontology alignment," in *Proc. 3rd conference on international semantic web conference (ISWC)*, 2004.
[4] N. F. Noy and M. A. Musen, "The PROMPT suite: Interactive tools for ontology merging and mapping," *International Journal of Human-Computer Studies*, vol. 59, no. 6, pp. 983–1024, 2003.
[5] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, and J. P. Rice, "OKBC: A programmatic foundation for knowledge base interoperability," in *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. Madison, Wisconsin, USA: MIT Press, 1998, pp. 600–607.
[6] J. de Bruijn, F. Martín-Recuerda, D. Manov, and M. Ehrig, "State-of-the-art survey on ontology merging and aligning v1," SEKT, Deliverable D4.2.1, 2004.
[7] D. Roman, H. Lausen, and U. Keller, "Web service modeling ontology standard (WSMO-standard)," WSMO, Working Draft D2v0.2, 2004.
[8] M. Schorlemmer, "On the mathematical foundations of semantic interoperability and integration," in *Semantic Interoperability and Integration*, ser. Dagstuhl Seminar Proceedings, Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, and M. Uschold, Eds., no. 04391. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005, <http://drops.dagstuhl.de/opus/volltexte/2005/44> [date of citation: 2005-01-01].
[9] C. Menzel, "Basic semantic integration," in *Semantic Interoperability and Integration*, ser. Dagstuhl Seminar Proceedings, Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, and M. Uschold, Eds., no. 04391. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005, <http://drops.dagstuhl.de/opus/volltexte/2005/42> [date of citation: 2005-01-01].