# AN APPROACH TO MATCH AND INTEGRATE ONTOLOGY USING ONTOLOGY REPOSITORY AND RULE BASE

Dan Wu and Anne Håkansson
*Software and Computer Systems, School of Information and Communication Technology,*
*KTH Royal Institute of Technology, Stockholm, Sweden*

Keywords: Ontology Matching, Integration, Reasoning, Knowledge Representation, Knowledge Base, Semantic Web.

Abstract: There exist a lot of ontologies that together can enrich knowledge within one or several related domains, thereby supporting the development of advanced services on the semantic web. This requires matching and integrating ontologies. This paper introduces an ontology matching process that handles the heterogeneities. The result is an intersection of the two original ontologies. An ontology repository stores the original ontologies and the matching results. A rule base is designed to integrate stored ontologies and the matching results with metadata, which is describing the interpretation of these ontologies and ontology matching results. The contribution of our approach is the semantic violation check which results in an ontology intersection that validates in the original ontologies. The metadata is applied with rules to integrate the ontologies so that the ontology and the matching results can be reused.

## 1 INTRODUCTION

When several ontologies are involved in reasoning, e.g., querying on the semantic web and combining ontologies to provide services on the distributed systems, the heterogeneousness of the ontologies becomes a problem. Ontology matching is the process of finding the correspondences between entities in heterogeneous ontologies (Euzenat and Shvaiko, 2007).

The ontology definition of Sowa (Sowa, 2011) illustrates some of the heterogeneities and the causes, i.e., an ontology is “a catalog of the types of things that is assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D.” (Sowa, 2011). To find correspondences across ontologies, we need to overcome the ontology heterogeneity on the syntactic, terminological, conceptual and semiotic levels (Bouguet et al, 2005).

By applying OWL 2 (W3C, 2009), syntactic heterogeneity is handled in this paper. The terminological, conceptual and semiotic heterogeneities remain. The proposed process matches two owl ontologies and produces an ontology intersection. The terminological differences is handled by the entity-string normalization and an external English lexical database. The ontology intersections are presented in ontology format and stored together with the original ontologies. Metadata describing the ontology conceptual and semiotic differences and are presented in rules. A rule base is designed for reusing the knowledge stored in the orignal ontologies and the ontology intersections.

## 2 RELATED WORK

Reviews over ontology matching techniques are found in (Euzenat and Shvaiko, 2007) .Below we present several other works.

Dou,*et al.* (Dou et al, 2005) developed a semi-automated process for semantic translations of the ontologies handling similar domains. This process includes developing bridging axioms to merge the related ontologies. The result of ontology merging is a merged ontology of the two input ontologies, and the merged ontologies can be used for further merging with other ontologies. This is an example of semantic approach.

The combination of matching techniques has also been tested in Ming Mao *et al.* (Mao et al, 2010). An automatic approach of matching two ontologies is


Wu D. and Håkansson A..
AN APPROACH TO MATCH AND INTEGRATE ONTOLOGY USING ONTOLOGY REPOSITORY AND RULE BASE.
DOI: 10.5220/0003937604340439
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 434-439
ISBN: 978-989-8565-08-2

described. There are three modules in the matching process, i.e., the IR-based similarity generator, the adaptive similarity filter and the weighted similarity aggregator. Linguistic and structural similarities are considered. The result is a set of statements that contains the semantic correspondences between similar elements with the associated relationship and the confidence.

In (Håkansson et al, 2010), an agent system with a knowledge base for comparing ontologies at the syntax level is explored. The research of this paper is a continuous of that work, but focuses on the semantic level.

Although the semantic ontology matching has been explored thoroughly, the technique of reusing the matching results by rules has been neglected. Our research fills this gap by ontology repository and metadata rules by describing how the ontology is interpreted.

# 3 MATCHING AND INTEGRATION

Only OWL 2 ontologies are handled in the process so far. Two ontologies are input. The information of entity labels, entity types and the expressions and axioms are considered in the process. An ontology intersection is produced after the matching process. Rules are applied to integrate the ontologies.

## 3.1 OWL 2 Ontology Examples

OWL 2 ontology contains expressions and axioms in the domain, which place constraints on sets of classes and individuals. However, in this paper the match is on the conceptual level, the information of individuals is not considered.

The ontologies discussed in this paper follow the W3C specification (W3C, 2009). Figure 1 shows two ontologies that will be matched and integrated. The signature of ontology1 is for example a list entities that contains both classes and properties; classes are Root, Document, Journal, Publication, Book, Presentation, Report, Topic, Author and Literal; properties are has-topic, has-author, name and date-creation. The signature of ontology 2 is Source, Document, Website, Publication, Ontology and hasAuthor. To compare these ontologies' signatures is the first step of the matching process.

The open world assumption (Knorr et al, 2011) is applied for reasoning on ontologies. The open world assumption means that an ontology reasoner will not

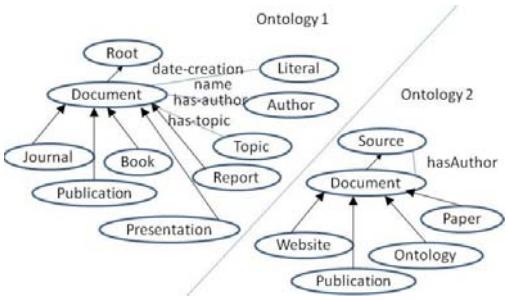


Figure 1: The example ontologies.

negate a statement unless it finds the explicit information in the ontology. This reasoning strategy is sound when several ontologies are integrated. In this paper, it means that if no explicit information of one ontology is found in another ontology, the absence of information brings no false of the statement in the other ontology.

## 3.2 Ontology Matching Process

Ontology matching process starts taking two ontologies, mentioned above, as input, and extracts the ontology signatures. The syntax comparison and the synonym comparison are carried on each entity of the signatures. Thereafter, entity candidates are generated, which are used for the semantic concept comparison. The result is an ontology also called ontology intersection. The ontology intersection is an ontology that is not violating with the original ontologies. For the whole ontology matching process, see figure 2.

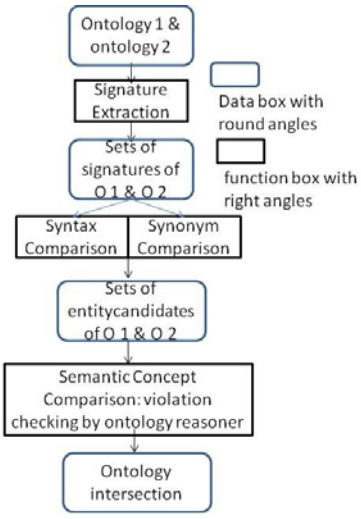


Figure 2: The matching process.

The labels, used in the ontologies, should give good index of similarities between them. Therefore;

the syntax comparison compares the label of each entity of one ontology with the other ontology. For example, the syntax comparison of ontology 1 and 2 generated a set of entities that were found in both ontologies, class of “document” and “publication” and object property of has-author (has Author). The following normalization strategies are implemented on syntax comparison:

1. *The letter cases are ignored*, i.e., “has Author” is the same as any combination of the upper and lower cases. For example, “HasAuthor” and “has author” are treated as equal;
2. *Only the letters are compared*, other special characters are excluded, e.g., “hasTopic” is the same as “has-topic” and “has_topic”;
3. *Grammatical forms are ignored*, i.e., singular and plural of nouns are equal and all the forms of verbs are ignored.

The result of the syntax comparison is a set of Class {document (document), publication (publication)} and a set of ObjectProperty {has-author (has Author)}.

Following the syntax comparison, the synonym comparison is carried on, i.e., each entities of one ontology is checked for synonym from the other ontology. The synonyms are checked and fetched from the online WordNet (Princeton University, 2011). Of the synonyms suggested by Wordnet, only those found in the other ontology are saved. For example, the class “document” in ontology 1, Wordnet gives several synonyms, such as “written document”, “papers” and “text file”. Among these synonyms, the class “paper” is found in ontology 2. Therefore, the “paper” is saved. After the synonyms comparison, the entity candidates are returned as: class {root (source), document (paper), report (paper), author (source)} and object property {has-author (has Author)}. The union of the results from the syntax comparison and the synonym comparison builds up the entity candidates: Class {root (source), document (document), document (paper), publication (publication), report (paper), author (source)}; ObjectProperty {has-author (has Author)}.

As shown above, the comparison is only made within the same entity types, i.e., class is compared with class and object property is compared with object property.

Semantic concept comparison checks violations of the ontology definition of the entity candidates. For each ontology, the definitions of the entity candidates are extracted; and the labels of the entities are swopped, i.e., the definition in ontology 1 with labels of ontology 2 is checked in ontology 2, as well the definition in ontology 2 with labels of ontology 1 is checked in ontology 1. For example, the definition of entity “root” is extracted from ontology 1; and the label “root” is swopped for “source”. Then, the axioms of “root” defined in ontology 1, now labelled “source”, are checked for violation in ontology 2. This process takes care of all the entities in entity candidates at the same time.

In our example, the definitions of all the entity candidates in ontology 1 are extracted. However, it happens that the whole ontology is involved and, then, the labels are swopped for the synonyms. The entities that have no synonyms are excluded from the axioms. The result is shown below:

```
Declare (Class (Source))
Declare (Class (Document))
Declare (Class (Publication))
Declare (Class (Paper))
Declare (Class (Source))
Declare (ObjectProperty (hasAuthor))
SubClassOf (Document, Source)
SubClassOf (Source, Document)
SubClassof (Publication, Document)
SubClassOf (Paper, Document)
ObjectPropertyDomain (hasAuthor,
Doument)
ObjectPropertyRange (hasAuthor,
Source)
```

One violation is found directly from the above description, i.e., two classes of Sources are found, because both Root and Author have Source as synonyms. Source is a more general conception than both Root and Author, since it is synonyms to both. The minimum action is to add Root and Author as two subclasses, and hence, the result has reformed as below:

```
Declare (Class (Source))
Declare (Class (Root))
Declare (Class (Author))
Declare (Class (Document))
Declare (Class (Publication))
Declare (Class (Paper))
Declare (ObjectProperty (hasAuthor))
SubClassOf (Root, Source)
SubClassOf (Author, Source)
SubClassOf (Document, Root)
SubClassOf (Paper, Document)
SubClassof (Publication, Document)
ObjectPropertyDomain (hasAuthor,
Doument)
ObjectPropertyRange (hasAuthor,
Author)
```

The open world reasoning is applied here, i.e., if the definition is not found in ontology 2, the statement is seen as not violating and saved in the ontology intersection. If the violation is found in

ontology 2, the related information will be excluded from the ontology intersection. In this example, the violation is solved by adding two subclasses to Class Source.

The similar violation checking runs from ontology 2 to ontology 1. Since entity Source has two synonyms Root and Author, the checking needs to be done twice, Source swop for Root and Source swop for Author. The result, below, shows the swop of Root:

```
Declare (Class (Root))
Declare (Class (Document))
Declare (Class (Publication))
Declare (Class (Report))
Declare (Class (Document))
Declare (ObjectProperty (has-
author))
SubClassOf (Document, Root)
SubClassOf (Report, Document)
SubClassof (Publication, Document)
SubClassof (Document, Document)
ObjectPropertyDomain (has-author,
Doument)
ObjectPropertyRange (has-author,
Root)
```

The violations here are the relationship "has-author" of Document and Root, and the hierarchy between Document in ontology 1 and Document in ontology 2. Therefore, these two expressions are excluded.

The result of the swopping of Source and Author is as following:

```
Declare (Class (Author))
Declare (Class (Document))
Declare (Class (Publication))
Declare (Class (Report))
Declare (Class (Document))
Declare (ObjectProperty (has-
author))
SubClassOf (Document, Author)
SubClassOf (Report, Document)
SubClassof (Publication, Document)
SubClassof (Document, Document)
ObjectPropertyDomain (has-author,
Doument)
ObjectPropertyRange (has-author,
Author)
```

The violations here are the subsumption relations of the Document and Author and the Document and Document. They are, therefore, excluded. The union of these two results return an intersection of a pre-result as shown below:

```
Declare (Class (Root))
Declare (Class (Document))
Declare (Class (Publication))
Declare (Class (Report))
Declare (Class (Author))
Declare (ObjectProperty (has-author))
SubClassOf (Document, Root)
SubClassOf (Report, Document)
SubClassOf (Publication, Document)
ObjectPropertyDomain (has-author,
Doument)
ObjectPropertyRange (has-author,
Author)
```

The conjunction of the violation checking of these two ontolgies is the following:

```
Declare (Class (Source))
Declare (Class (Root))
Declare (Class (Author))
Declare (Class (Document))
Declare (Class (Publication))
Declare (Class (Paper))
Declare (Class (Report))
Declare (ObjectProperty (has-author))
Declare (ObjectProperty (hasAuthor))
SubClassOf (Root, Source)
SubClassOf (Author, Source)
SubClassOf (Document, Root)
SubClassOf (Paper, Document)
SubClassOf (Report, Document)
SubClassof (Publication, Document)
ObjectPropertyDomain (hasAuthor,
Doument)
ObjectPropertyRange (hasAuthor,
Author)
ObjectPropertyDomain (has-author,
Doument)
ObjectPropertyRange (has-author,
Author)
EquivalentObjectProperties (has
Author, has-author)
```

The axioms above are the result of the semantic concept comparison, which is called the ontology intersection of ontology 1 and ontology 2. It does not conflict with the original ontologies with the open world assumption reasoning. The intersection of ontology is expressed in Figure 3.

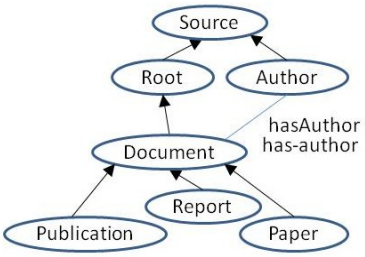


Figure 3: Ontology matching result: Ontology intersection.

## 3.3 Ontology Integration with Metadata and Rules

By defining metadata, the definitions from ontologies and the ontology matching results (ontology intersections) are expressed in rules to

enrich the knowledge based on the heterogeneous ontologies. The metadata and rules describe the context of these ontologies, which is the conceptual ontology integration, see figure 4. Ontology is composed of entities. The set of the entities of the ontology is the signature. Each entity has an entity type that is given according to owl 2. Ontology can be described with its domain, purpose, creator and date, which can usually be found in ontology annotation. An ontology describes one domain. Domain can contain several sub-domains. One ontology is created for one purpose, by one creator at one date. One domain can be described by one or several ontologies. The ontology intersection is the result of ontology matching and is an ontology itself. One ontology intersection is connected with at least two ontologies.

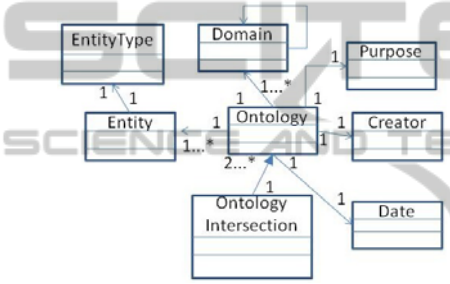


Figure 4: Ontology metadata model.

Often the metadata information of domain, purpose, creator and date is available in the ontology definitions and can be extracted automatically. Nevertheless, it is possible to manually add these data if they are missing from the ontology. For example, the two example ontologies integrated belong to the domain of university; a rule can be expressed as

```
If Domain (university) Then
identificationOfOntologyIntersection
```

The identificationOfOntologyIntersection is a link to the declarations of the result.

One creator may write several ontologies during different time, and, then, a rule can be expressed as:

```
If Creator (x)
   If Date (d1) Then
   indentificationOfOntology
  If Date (d2) Then
identificationOfOntology
End
```

The relationship between the metadata can be expressed in rules for reusing as well. For example, certain entities used in a Domain are always interpreted with an ontology definition is shown:

```
If Entity (x,y,.., z) and Domain (d)
Then
identificationOfOntologyIntersection
```

One example of this is the integration done in the previous part:

```
If Entity (document, publication,
paper) and Domain (university)
Then
identificationOfOntologyIntersection
```

Another rule can be defined from the previous example, that all the ontology definitions can be integrated. It interprets each ontology definition is a perspective view.

```
If Ontology 1 Then
identificationOfOntology1
If Ontology 2 Then
identificationOfOntology2
If Ontology 1 and 2 Then
identificationOfOntologyIntersection
```

The context for the ontologis can, together, give more knowledge about a service and can, if combined, provide advanced services to the users. The context rules provide the reasoning and integration for the heterogeneous ontologies.

With the rules, the close world assumption (Knorr, et al, 2011) is applied, i.e., the antecedent must be satisfied in order to conclude the concequent.

The rules should not be impeded by definitions of ontologies. In another word, these rules have a higher priority than individual ontologies.

## 4 THE ARCHITECTURE

To keep track of the ontologies, the ontology metadata models and the rules, a modular architecture is proposed, see figure 5. The ontologies are stored in the ontology database as owl files. The ontology repository stores the metadata models to manage the ontologies. The rule base stores the context integration rules that are used for context comparison and the Owl reasoner is applied on ontologies and for testing and querying on ontologies, for example, for the concept violation checking. The rule engine determines rules to be fired and generates results according to rules.

The ontology loader load ontologies in ontology database. It has a parser, which parses the owl files stored in the ontology database and extracts the signatures from the ontologies and stores them in the ontology repository, together with the definition of entity type and the links to the ontologies.

The ontology violation detector checks the semantic conceptual violations between two

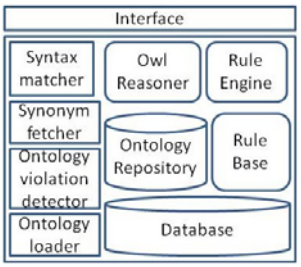


Figure 5: The architecture.

ontologies. The process handles semantic conceptual comparison. The process of synonym comparison works with synonyms which the synonym fetcher finds by searching in the WordNet and return entities synonyms.

The syntax matcher provides a function for terminology syntax matching. The process is the syntax comparison that normalizes entities and return equal entities.

Interface handles the input from users and brings these modules together with user-friendly interfaces. An input is needed when conflicts have been found by the ontology violation detector. To solve the conflict, the users can choose to provide manual mappings of the conflicted definitions or can confirm or reject the suggestions of the matching process. The users can also write rules using metadata to integrate ontologies.

## 5 CONCLUSIONS

The main contribution of this paper is the conceptual ontology intersections that generated in two steps. The first step handles the label strings and the ontology type and taking help from WordNet with synonyms. Matching candidates are generated from this step and then semantic violation checking is performed on only the candidates. Since we believe that the labels give quite good index to what they mean; the similarity of label strings give good candidates for the semantic checking. The semantic violation checking saves lots of reasoning by restricting to candidate checking.

The ontologies and the matching results are stored in the repository. Repository applies metadata not only managing the ontologies but also providing contexts of ontologies. The rules in the rule base apply metadata to interpret the ontology and are reused to integrate the ontologies. With the ontology intersections and the rules, the ontology repository is functioning as integrated knowledge across these ontologies.

However, this work is in the first stage. The approach needs to be applied and tested with more ontologies. The result is expected to be more effective in restricted domains.

## REFERENCES

Aumueller, D., Do, H., Massmann, S., Rahm, E., Schema and Ontology Matching with COMA++, SIGMOD 2005, June 14-16, 2005, Baltimore, Maryland, USA.

Bouquet, P., Ehrig, M., Euzenat, J., Franconi, E., Hitzler, P., Krötzsch, M., Serafini, L. Stamou, G., Sure, Y., Tessaris, S., 2005. ”D2.2.1 Specification of a common framework for characterizing alignment”, Knowledgeweb – realizing the semantic web.

Dou, D., Mc Dermott, D., and Qi, P., 2005. “Ontology Translation on the Semantic Web”, Journal on Data Semantics II, Springer, Berlin/Heidelberg, volume 3360/2005, p.35-37, January 2005.

Euzenat, J., Shvaiko, P., “Ontology Matching”, Springer Berlin Heidelberg, 2007.

Knorr, M., Alferes, J., Hitzler, P., 2011. “Local closed world reasoning with description logics under the well-founded semantics”, Artificial Intelligence doi: 10.1016/j.artint. 2011.01.007.

Mao, M., Peng, F., Spring, M., 2009. “An adaptive ontology mapping approach with neural network based constraint satisfaction”, Web Semantics: Science, Services and Agents on the World Wide Web, Elsevier, 8 (2010) 14-25, 2009.

Princeton University, 2011. Use Wordnet online, http:// wordnetweb.princeton.edu/perl/webwn, accessed on 6th December, 2011.

Sowa, J. F., 2011. “Ontology”, http://www.jfsowa.com /ontology/index.htm, accessed 8th November 2011.

W3C, 2009. “OWL 2 Web Ontology Language, Structural Specification and Functional-Style Syntax, W3C Recommendation 27 October, 2009.