Taxonomic Ontology Alignment - an Implementation

Carolina Howard Felicíssimo, Karin Koogan Breitman
PUC-RIO – Pontifical Catholic University of Rio de Janeiro, Department of Informatics,
R. Marquês de São Vicente 225, Rio de Janeiro, CEP 22453-900, RJ, Brasil
e-mail: [cfelicissimo, karin]@inf.puc-rio.br

Abstract. *With the evolution of the Web evolving towards the Semantic Web, where the information should be presented in a meaningful way for both humans and machines, arises the need for semantic interoperability. By interoperability we understand the use of meaningful representation mechanisms that allow for information exchange among agents. There is a consensus that ontologies will be this representation model. The information exchange process depends on our ability to engage two or more ontologies in conversation. In this paper the focus is on the compatibility of different ontologies. We propose a strategy for automatic alignment of ontologies, implemented by an ontology taxonomic alignment component - the CATO component.*

1 Introduction

The major drawback of today's web is the great volume of unstructured information. To face this problem, both industry and academia are exploring the possibilities of creating a semantic web, where information will be made available in such way that software agents will be able to process and integrate resources, allowing for more meaningful information retrieval [1].

Ontologies will play a major role in the semantic web. They provide a representation model that should become the standard for information exchange among agents. It is believed that in a near future most businesses on the web will provide the semantics of their pages by means of an ontology [2].

The most common definition for ontologies is that of a "*formal, explicit specification of a shared conceptualization*", proposed by Gruber [3]. For the purposes of this paper, we use Maedche's definition [4], for it details the ontology primitive building blocks, fundamental to our approach.

Ontology construction is supported by a number of methodologies like Methontology [5] and Tove [6], and methods proposed by Uschold and Gruninger in [7], Noy and McGuiness in [8] and Breitman and Leite in [9], but the communication between ontologies remains a problem. In the next section we discuss the ontology interoperability problem and in section 3 we propose our ontology alignment strategy. The implementation of this strategy is presented in section 4 and a case study is presented in section 5. We comment our approach and present our concluding remarks in section 6.

2 Ontology Interoperability

Ontologies represent important entities (also known as classes), their taxonomical relationships, properties and restrictions that apply to the entities of the domain. In some representations we can also use axioms to declare truths that are valid in the

Universe of Discourse of the ontology. In the semantic web context, the use of ontologies, rather than to provide a domain theory, has the goal of providing a standard representation in which to exchange information among different agents. Thus it is fundamental to provide mechanisms that allow information exchange among ontologies.

Currently there are a few approaches that deal with the ontology compatibility problem. The most salient ones are: (1) ontology merge [10], (2) ontology alignment [10], (3) ontology mapping [11] and (4) ontology integration [12].

Merging ontologies results in a unique ontology that contains all the terms from merged original ontologies, without indication of their former origin. Aligning ontologies results in separate ontologies with links among them. The links allow ontologies to share terms. Mapping ontologies results in a formal structure containing expressions that link concepts from one conceptual model to another. Finally, integrating ontologies results in a unique ontology created by assemblage, extension, specialization or adaptation of ontologies from different subject areas. When integrating ontologies it is possible to identify the relationships to the original ontology.

2.1 Our Understanding of Ontology Alignment

In this work, we aim at identifying common terms that exist among ontologies that complement one another in such way that we allow for negotiation, i.e., information exchange, among such representations. Therefore, we define ontology alignment as the process by which two ontologies[1] must undergo in order to establish an intermediate representation that guarantees communication (information sharing and exchange) between them. The result of the alignment process is a persistent model that indicates the links between ontologies allowing for information share, exchange and, possibly, reuse.

From a software engineering point of view, the alignment process must be quick, automatic and reliable, in order to support current interoperability requirements of multi-agent systems [13]. Speed is a requirement intrinsic to the web. We understand that, once a request is sent by an autonomous agent, its response should be given in execution time (the request may have been sent to multiple agents, each with its own ontology, making competition and time of response very important factors). The process must also be automatic, for that we should not count with user intervention. We must remember that while the designer of the ontology is a domain expert, the user of the service that requires semantic exchange via ontologies is not necessarily an expert.

The total alignment is desired but not necessary. We would rather guarantee a minimum level of reliability, which means that some of the terms will not be aligned, if it can be done within a reasonable time frame. We consider partial alignment good enough if it is attained quickly, automatically and within pre defined reliability limits.

[1] We implemented the alignment process for pairs of ontologies. If it is the case of aligning more than two, it can be done in progressive steps (two by two, always). We have conducted some experimentation in this sense and have noted that the order in which the ontologies are aligned does alter the final result significantly. Future experimentation will include the determination of heuristics to minimize the impacts of the order of choice in the process.

In this case, the aligned ontology does not need to be stored since we will compute the alignment when it is needed.

3 Proposed Strategy

We propose a three step ontology alignment strategy, illustrated in Figure 1. On the first step we compare concepts from two ontologies lexically using a trimming mechanism as a stop condition. The results of this step are the input ontologies enriched, i.e., with the alignments of this step. The second step is comprised of the structural comparison of the ontologies structures and it results the concepts classified as *equivalents*. On the third step, we classify terms according to a pre defined similarity measurements in order to determine whether they can be aligned or not.

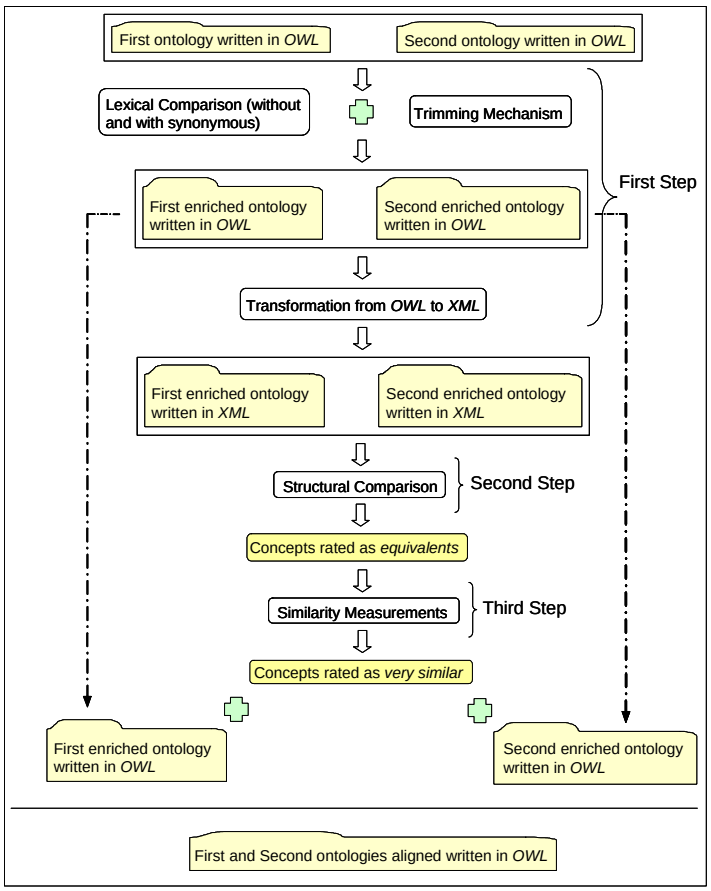

Fig. 1. Proposed ontology alignment strategy

Only the concepts classified as *very similar*[2] in this last step will be added in the enriched ontologies result of the first step. After this addition, the ontologies will be joined into a one single ontology.

4 An Implementation for the Proposed Ontology Alignment Strategy

To validate the proposed strategy we built an ontology taxonomic alignment component called CATO. This component will mediate services provided by agents in the framework proposed in [13, 14]. CATO uses two ontologies as input. The current implementation accepts ontologies written in the *OWL* ontology language [15], recommended by the W3C consortium [16]. The online version of CATO is available at the following address: http://cato.les.inf.puc-rio.br.

CATO was fully implemented in JAVA [17] and it uses a specific API (Application Programming Interface) for Java built to deal with ontologies, the Jena's API [18]. Jena functions as a framework in which to anchor the construction of semantic web applications. It provides a programming environment for semantic web ontology languages including a rule based inference mechanism.

By using the Jena API it is possible to transform an ontology into an abstract model that can be dealt with using Java Object Oriented programs. A series of methods is also provided, that make it possible to easily retrieve concepts related information. The use of this API helped us focus on the alignment process, for it made ontology manipulation transparent. It is important however, that Jena is not an inference mechanism, it does not deduct information. It rather reads and filters information from the tags of files written in an ontology language.

In the next sections, we describe the implementation of each step of the proposed strategy.

4.1 First Step: Lexical Comparison with the Use of Synonymous and Trimming Mechanism as a Stop Condition

The initial step of the proposed strategy takes two ontologies as input. Those ontologies are transformed to models using the Jena API, with the goal of manipulating those as objects. The goal of this step is to identify lexically equivalent concepts and sub concepts, i.e., the ones with the same semantic content.

Each concept and sub concept of the first model is compared, one by one, to every concept and sub concept present in the second model, using name similarity as the criteria. Besides using the label, synonyms are also used.

The use of synonyms enriches the comparison process, because it provides more refined information. Automatic extraction of valid synonyms from existing dictionaries is not a trivial task, so we were forced to create a synonym database to aid in this process.

One ontology may contain concepts imported from other ontologies. At the time of the creation of the ontology model by the Jena API, all the imported terms will be present in the model, as if they were part of the original ontology. Each concept in an ontology has a unique identifier formed by the set of its name and the namespace of

[2] similarity rate over than 75%

the ontology it belongs to. The namespace verification procedure allows us to identify imported concepts from those original to the ontology. For the sake of speed, we do not take into consideration concepts that have been imported from other ontologies. Results from our experimentation with the CATO component have shown that it would slow down the process excessively. Besides this, the information of those concepts would not help in the following steps of the strategy.

Once lexical similarity has been detected between concepts, it is necessary to compare the sub trees of the pair to determine if they are in fact compatible (and thus reliable alignment). It is important to note that this step's alignment strategy is restricted to the concepts, sub concepts and instances of the ontology. We are not considering properties at this time. A concept instance is represented by the pair name and namespace. Because we compare concepts from different ontologies, its namespace are different. Our trimming strategy aims at the identification of the instances, which share the same label in the compared ontologies, from the concepts that were identified as equivalent by the lexical analysis. In this case, we had to filter the input and compare the instances' name of the concepts separately from its namespaces.

The result of the proposed strategy's first stage is the models of both original ontologies enriched with links that point to concepts and sub concepts that were identified as equivalent. The links are added to both models. The enriched models are stored in *OWL* format and will be the input to the next step of the proposed strategy.

4.2 Second Step: Structural Comparison Using an Implementation of the *TreeDiff* Algorithm

The second step of the proposed ontology alignment strategy is the comparison between the structure hierarchies of the ontologies. This comparison is based on the subsumption relationship held among ontology concepts. Ontology properties (defined as *function* that relates the concepts non taxonomically, in Maedche's definition in section 1) are not considered.

Our approach is thus more restricted than the one proposed in [19], that analyses the ontologies as graphs, taking into consideration both taxonomic and non taxonomic relationships among terms. Because we only consider the taxonomic relationships at this point, we are able to make use of well known tree comparison algorithms. We are currently using the *TreeDiff* [20] algorithm proposed in [21]. Our choice was based on the ability to identify structural similarities between trees in reasonable computing time.

The goal of the *TreeDiff* algorithm is to identify the largest common substructure between trees, described using the DOM (*Document Object Model*) model. This algorithm was first proposed to help detect the needed steps, including renaming, removing and addition of tree nodes that were necessary to migrate one tree to another one (both trees are the inputs to the algorithm). We used the implementation of the *TreeDiff* algorithm in [21], which focuses on scenario evolution. With few modifications to the original implementation it was possible to configure the editing operations to rename only (not anymore insertion and removal operations). Those modifications in the algorithm were needed because, in our case, one ontology is not

the evolution of a second, but separate artifacts that need to be compared in order to identify shared conceptualizations.

Our first idea was to adapt the algorithm to accept entries in the *OWL* file format. Today's DOM parser implementation accepts only the specific Document Object Model provided by the *XML* format. Because at this stage we use only the taxonomical structure of the ontology, it is acceptable to use *XML*. This approach would not be acceptable if additional information, e.g., concept restrictions or axioms were needed, because they are lost in the conversion process. Figure 2 illustrates the *TreeDiff* algorithm with its entries and exits.

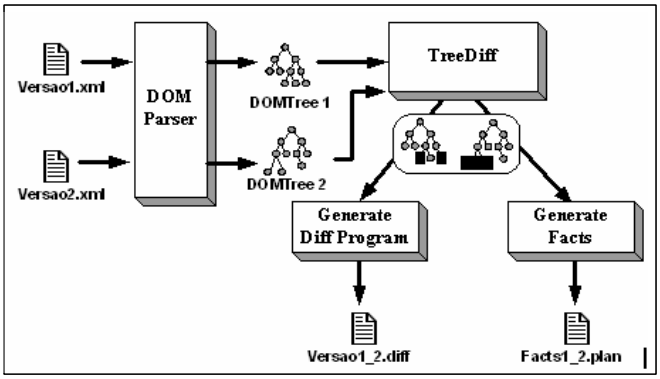

Fig. 2. The *TreeDiff* algorithm's entries and exits [21]

4.2.1 Equivalence Information

Before the transformation of the OWL ontologies to a hierarchical model representation, it was necessary to choose the names or synonyms of equivalent concepts that were going to be used by the comparison process. The structural comparison process is based in lexical equality (identified in the previous step); synonyms are no longer used in this step. The equivalence information is linked to the concepts of both ontologies, as described in section 4.1. One of the concepts has to be chosen as preferred label and the substitution in the other ontology is made.

There is, however, the possibility that the substitution will incur in the addition of a label that is already in use by another concept in the ontology, probably with another meaning. This would be an inconsistency (the pair name and namespace has to be unique) so the substitution process is aborted and the information of the equivalence pair is left out of the structural comparison process.

4.2.2 Equivalence Groups

We make use of equivalence groups, as proposed by [19], during structural comparison. Equivalence groups are identified both by lexical and structural comparison. First concepts with identical labels are identified and, then, we compare their structure, i.e., number of sons and number of equivalent concepts in those.

Another factor that is taken into account is the order in which the descendants of equivalent concepts appear.

We implemented two different modules. One that contains the files for structural comparison alphabetically ordered and the other in the original order of appearance of the concepts and sub concepts in the ontology. For ontologies that make use of identical labels, the use of the alphabetically ordered structural comparison files brought significantly better results. This was due to the fact that after the alphabetical sort, the concepts present in an equivalence group will be closely located in the structure. However, ontologies that make use of identical labels are rarely the case in practice. We tested the performance of both files on ontologies that had few identically labeled concepts. The ordered file did not bring differences in the results and there were some cases when it made the results worse than using the unordered file.

4.3 Third Step: Use of Similarity Measurements for Fine Adjustments

The third and last step is the use of similarity measurements. Concepts are rated as *very similar* or *little similar* based on pre defined similarity thresholds. We only align concepts that were both identified as equivalent, in the previous step, and rated very similar. Thus the similarity measurement is the deciding factor responsible for fine tuning our strategy.

We adopt the similarity measurement strategy proposed in [21]. Experimentation showed that it provided good results, when compared to manual concept alignment.

Our similarity threshold was empirically defined. We are currently working with 75% threshold, i.e., concepts whose similarity rate is equal or over 75% are very similar whereas those with less than 75% similarity are rated little similar and therefore not aligned (despite the fact that they may have been identified as equivalent to some other concept in the first step of the strategy).

If new equivalences are detected during this step, those are annotated in the corresponding ontology model.

5 Case Study

Two independent ontologies, created by different groups, were chosen for the case study that will be presented. The first group chosen is one the responsible for the project Agent Transaction Language for Advertising Services (ATLAS) [22] from the Carnegie Mellon University [23]. Their ontology called CMU RI Publications [24] is the first ontology chosen[3]. The second group chosen is represented by the French company Mondeca SA [25], one of the companies of the W3C Web-Ontology's work group [26]. Their ontology called General University Ontology [27] is the second ontology chosen.

The two chosen ontologies have differences in their taxonomic organization and in the number of their terms. Figure 3 shows the compared ontologies' hierarchy, with O1 as an abbreviation for the first ontology described and O2 as an abbreviation for

[3] As CATO receives ontologies written in the OWL language and the ontology chosen is written in DAML language, the transformation from OWL to DAML is needed. An ontology editor can do that, like the OilEd [28] editor used.

the second one. The first ontology has 25 concepts and the another one has 225 concepts, in total.

From all the concepts of the compared ontologies just 8 from each one can be aligned. The arrows in the Figure 4 point to the concepts that can be aligned in the compared ontologies, with the thicker arrows meaning concepts that will be aligned by CATO.

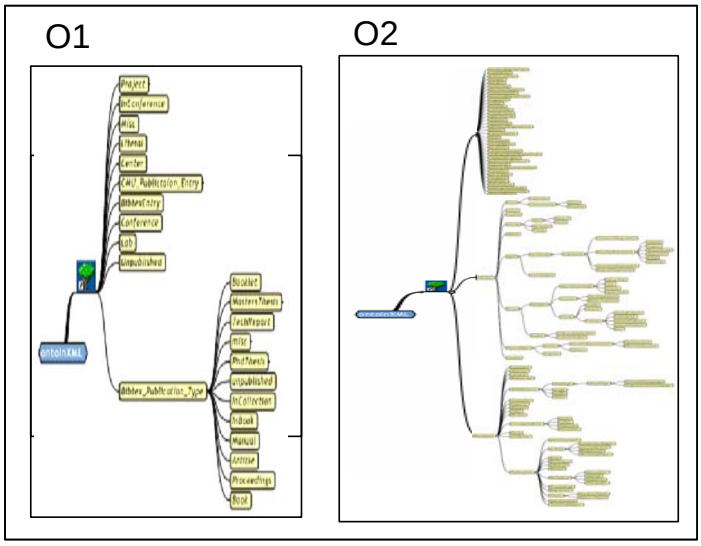

Fig. 3. The compared ontologies' hierarchy of the case study presented

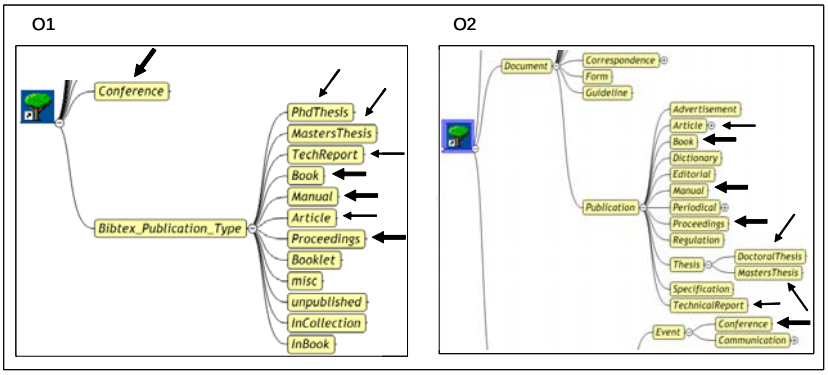

Fig. 4. Concepts that can be aligned in the compared ontologies

5.1 First Step

This step of the strategy compares concepts from ontologies lexically, also using synonymous. In the beginning, the data base used to store the synonymous is empty.

Because of that, for this example, the following information were registered there: “TechReport” as synonymous of “TechnicalReport”, “TechnicalReport” as synonymous of “TechReport”, “PhdThesis” as synonymous of “DoctoralThesis” and “DoctoralThesis” as synonymous of “PhdThesis”.

CATO executes and recognizes the concepts with identical labels in both ontologies (“Conference”, “MastersThesis”, “Book”, “Manual”, “Article”, “Proceedings”) and the ones with their synonymous but, as none of them satisfy the trimming condition of this step of the strategy because those concepts have different labels for their super-concepts, they will not be aligned.

At the end of this step, the ontologies’ hierarchy are represented in XML files, with the exactly organization of the ontologies’ concepts and their sub-concepts. That is because the next step of the strategy just compares the ontologies’ hierarchy, i.e., how the concepts are organized in the compared ontologies. Figure 5 shows part of the representation in XML of the ontologies’ hierarchy. Those XML files will be the entry for the next step of the strategy.

5.2 Second Step

This step of the strategy compares the ontologies structures. Figure 5 shows the ontologies’ hierarchy compared and the identified equivalence group, described in section 4.2.2, represented by circles in the figure.

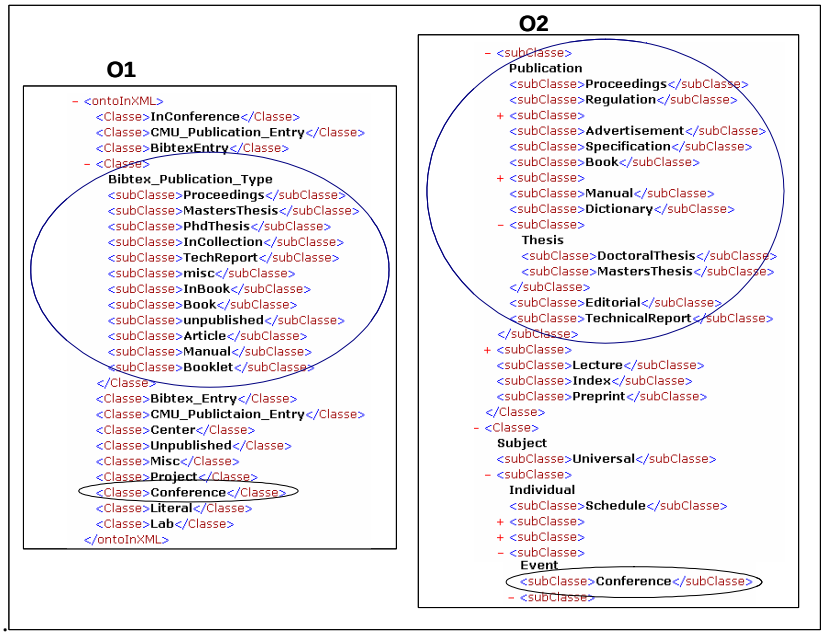

Fig. 5. Part of the ontologies’ hierarchy represented in XML files

The first equivalence group, represented by the superior circles, is formed because of the lexical equality between the concepts named “Proceedings” in the two compared ontologies and the structural similarity between their super-concepts. After the equivalence group is formed, all of the concepts inside it will be compared, looking for concepts that have lexical equality and structural similarity. With these requirements, the concepts named “Book” and “Manual” from the compared ontologies will be recognized as equivalents.

Because of the structure similarity (a single concept, i.e., a concept without sub-concepts, in O1 and a single sub-concept in O2) and the lexical equivalence between the concepts named “Conference”, the second equivalence group, represented by the inferior circles in the figure, is formed.

However, others concepts with identical labels, like those named “MastersThesis” and “Article” in both ontologies, will not be recognized as equivalent because their structures have differences. At the end of this step, the equivalent concepts “Proceedings”, “Book”, “Manual” and “Conference”, from the compared ontologies, were recognized as equivalent concepts.

5.3 Third Step

This step classifies the equivalent concepts, recognized in the previous step, as very or little similar according to a pre defined similarity measurements, in order to determine whether they can be aligned or not. CATO will align only the very similar concepts. It will happen with the concepts “Conference”, “Proceedings”, “Book” and “Manual”, from both ontologies. Those concepts are presented inside the identified equivalence groups, written with same names and with similar hierarchical structure.

Figure 6 shows the calculated percentages of similarity of the equivalent concepts recognized.

```
Similarities Level:
Bibtex_Publication_Type -> Publication   *** Similarity Level: 23.076923%
Conference -> Conference   *** Similarity Level: 100.0%
Proceedings -> Proceedings   *** Similarity Level: 100.0%
Book -> Book   *** Similarity Level: 100.0%
Manual -> Manual   *** Similarity Level: 100.0%
```

Fig. 6. Percentages of similarity calculated by CATO

In the end of this step, the equivalent concepts “Proceedings”, “Book”, “Manual” and “Conference”, from the compared ontologies, were recognized as very similar concepts (similarity rate over than 75%) and are aligned by CATO.

6 Conclusion

In this paper we discussed the implementation of a software component responsible for the automatic taxonomical alignment of ontologies. Our strategy is based on the sequential application of well known software engineering strategies, such as lexical analysis, tree comparison and the use of similarity measurements to the problem of ontology alignment.

For the sake of speed, we are only taking in consideration taxonomical relationships in the proposed alignment strategy. However, this limitation of the

strategy can be overcome by the adaptation of the second step to take into consideration other ontology primitives, such as properties (the strategy could work with graphs instead of trees) and axioms. For sure this adaptation will increase the total computation time because of the added complexity.

The worst case scenario in terms of completeness is not being able to align any concept. This happens when the input ontologies are from disjoint domains. The worst case scenario in terms of inconsistency is aligning two concepts that have identical labels, but are semantically different. This would only happen if, and only if, both share identical labels and possess structural similarity (equivalent concepts must be present in the descendants and the number of descendants must also be compatible). This case is very unlikely to happen, if the ontologies were created using some conceptual modeling.

We believe that addition of new functionalities to the current implementation will result in more precision in the results. The use of thesauri and reference ontologies, for example, will enhance the identification of semantically similar concepts that use different labels.

The *TreeDiff* algorithm is unidirectional in the sense that its goal is to determine the transformation needed to go from the first input tree to the second input tree. We are currently experimenting with double runs of the algorithm in which we invert the order of the input (today we are using the biggest ontology as the first input). When compared, the results present some differences. We believe we can refine the results from this algorithm by providing the combination of the two runs. Future plans include continuing validation of the approach by experimentation and the elaboration of more case studies.

7 References

1. Berners-Lee, T., Lassila, O. Hendler, J., The Semantic Web, available in: http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html, access in May, 2004.
2. Fensel, D., Ontology: A Silver Bullet for Knowledge Management and Electronic Commerce, *SpringerVerlag*, 2001.
3. Gruber, T.R., A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition, 5*, pp. 199-220.
4. Maedche, A., Ontology Learning for the Sematic Web, *Kluwer Academic Publishers*, 2002.
5. Gómez-Pérez, A.; Fernadéz-Peréz, M.; Corcho, O., Ontological Engineering, Springer Verlag, 2004.
6. Gruninger, M.; Fox, M., Methodology for the Design and Evaluation of Ontologies, *Workshop on Basic Ontological Issues in Knowledge Sharing at International Joint Conferences on Artificial Intelligence*, Montreal, Canada, 1995.
7. Ushold, M.; Gruninger, M., Ontologies: Principles, Methods and Applications, *Knowledge Engineering Review, Volume 11, Number. 2*, 1996, pp. 93-136.
8. Noy, N.; McGuiness, D., Ontology Development 101 – A guide to creating your first ontology, *Technical Report*, Knowledge Systems Lab, Stanford University, 2001.
9. Breitman, K. K., Leite, J. C. S. P. - Ontology as a Requirement Engineering Product, *11th IEEE International Requirements Engineering Conference*, Monterey Bay, California, USA, September 2003, pp. 309-319.

10. Noy, N. F., Musen, M. A., SMART: Automated Support for Ontology Merging and Alignment, *Workshop on Knowledge Acquisition, Modeling, and Management*, Banff, Alberta, Canada, 1999.
11. Noy, N. F., Musen, M. A., The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping, *International Journal of Human-Computer Studies*, 2003.
12. Pinto, S. H., Gómez-Pérez, A., Martins, J. P., Some Issues on Ontology Integration, *International Joint Conference on Artificial Intelligence*, Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends, Stockholm, Sweden, 1999.
13. Breitman, K.K., Haendchen, A.F., Staa, A., Haeusler, H., Using Ontologies to Formalize Services Specifications in Multi-Agent Systems, *Third NASA - Goddard/ IEEE Workshop FAABS III - Formal Approaches to Agent-Based Systems*, Greenbelt, Maryland, USA, 2004.
14. Haendchen, F., A.; Staa, A.v.; Lucena, C.J.P, A Component-Based Model for Building Reliable Multi-Agent Systems, *Proceedings of 28th SEW - NASA/IEEE Software Engineering Workshop*, Greenbelt, MD, IEEE Computer Society Press, Los Alamitos, CA, 2003.
15. Dean, M., Schreiber, G., Bechhofer, S., Harmelen, F. v., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L. A., OWL Web Ontology Language Reference, available in: http://www.w3.org/TR/owl-ref/, access in May, 2004.
16. W3C: World Wide Web Consortium, available in: http://www.w3.org/, access in May, 2004.
17. The Java Technology, available in: http://java.sun.com, access in May, 2004.
18. Jena, the Semantic Web Framework, available in: http://jena.sourceforge.net/, access in May, 2004.
19. Noy, N. F., Musen, M. A., Anchor-PROMPT: Using Non-Local Context for Semantic Matching, *Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence*, IJCAI, 2001.
20. Wang, J., An Algorithm for Finding the Largest Approximately Common Substructures of Two Trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 20, Number 8, pp. 889-895, 1998.
21. Bergmann, U., "Evolução de Cenários Através de um Mecanismo de Rastreamento Baseado em Transformações", *PhD Thesis* of the Department of Informatics of PUC-Rio, 2002.
22. ATLAS. Agent Transaction Language for Advertising Services, available in: http://www.daml.ri.cmu.edu, access in June, 2004.
23. Carnegie Mellon University, available in: http://www.cmu.edu/, access in June, 2004.
24. CMU RI Publications, available in: http://www.daml.ri.cmu.edu/ont/homework/cmu-ri-publications-ont.daml, access in June, 2004.
25. Mondeca SA, A Semantic Knowledge Company, available in: http://www.mondeca.com, access in June, 2004.
26. W3C Web Ontology (WebOnt) Working Group, available in: http://www.w3.org/2001/sw/WebOnt/, access in: June, 2004.
27. General University Ontology, available in: http://www.mondeca.com/owl/moses/univ.owl, access in: June, 2004.
28. OilEd - an Ontology Editor, available in: http://oiled.man.ac.uk/, access in June, 2004.