Automatic Merging of Knowledge using Ontologies

Alma Delia Cuevas Rasgado and Adolfo Guzman Arenas
Center for Research in Computer Science, National Polytechnic Institute, Av. Juan de Dios Batiz, s/n, Zacatenco, 07738, Mexico City, Mexico
almadeliacuevas@gmail.com; aguzman@acm.org

Abstract. The fact that many people simultaneously construct the pages of the Web in an independent way, generates a great obstacle for the machines that track the information in it. Therefore, the concept of Semantic Network has been introduced. It provides a standardization of the information through markup languages (SGML, XML, etc.) where the user generates his own annotations, almost all of them as labels or syntactic rules. Relatively few of the languages have try to represent and to manipulate the knowledge with methods of Artificial Intelligence. This paper proposes a structure (an ontology) more suitable to represent the knowledge, with interesting contributions with respect to current languages (AML+OIL[5], RDF[8], OWL[12]). Also, this paper presents an automatic algorithm to match and merge two or more ontologies. This merging is important when it is desired to increase the knowledge in an ontology. In that way it is possible to accumulate the knowledge in an automatic way. The process of merging begins by obtaining the value of the similarity between each elements of the ontologies (through COM[1] Algorithm); later, the optimal matching is sought. Finally, the result defines the new ontology. This process is performed totally by the computer. That is to say, the user does not take part in this process, as it happens in current merging algorithms (OntoMerge[6], FCA-Merge[9], Chimaera[11], Prompt[13], If-Map[14]). In the merging, the OM Algorithm solves problems of contradiction and reorganization of the final ontology. The efficiency of the algorithm of fusion is demonstrated through several examples.

Keywords: Knowledge base, Internet, Ontology merging

1 Introduction

These days computers are not anymore isolated devices but they are important entry points in the world-wide network that interchanges knowledge and carry out business transactions. Nowadays, using Internet to get data, information and knowledge interchange is a business and academic need. Despite the facilities to have access to the Internet, people face the problem of heterogeneous sources because there are not

suitable standards in knowledge representation. This paper addresses this need of businesses and academia.

Many answers that people require involve acceding several sources in the Internet and then they merge manually the acquired information in a reasonable way. Merging the information is an important task and many languages and tools (DAML+OIL[5], RDF[8] and OWL[12]) have been developed to describe and process Internet content but the languages lack enough expressiveness to detail knowledge representation.

It is required that computer decipher the information (said, in a document written in a natural language) and convert it to a suitable notation (its knowledge base) that preserves relevant knowledge. This knowledge base can be an ontology. Ontology is an information technology that manages the knowledge through nodes that are joined with each other through relations, to describe a knowledge domain. Current works that merge ontologies (OntoMerge[6], FCA-Merge[9], Chimaera[11], Prompt[13], and If-Map[14]) rely on the user to solve the most important problems found in the process. This paper describes two important contributions to obtain better advantages of the Web resources:

1. A new notation to represent knowledge using ontologies, called OM (Ontology Merging) Notation and
2. An automatic algorithm to merge ontologies called OM Algorithm- That is, without human intervention

The OM notation provides several improvements to current languages of definition of ontologies. Two of them are: (a) the new type of relation called *Partition*; (b) a node or concept can also be defined as a relation.

Likewise, the merging algorithm that we will explain is totally automatic. This algorithm solves by itself all the problems found in the process. That is to say, the user does not take part in the process.

2. OM Notation

In the context of sharing knowledge, ontologies provide a clear, syntactic and formalized structuring of a set of nodes also called concepts that are related to each other, under a knowledge domain and that is common to many people and machines.

OM Notation represents ontologies through a structural design with labels similar to XML. Theses labels identify the description of the concepts and their relations. The labels and their descriptions are shown on table 1.

The binary and n-ary relations are described in OM Notation. That is, a relation can have more of one value and these could be concepts. For example, *Zebra* concept has a relation *Color* that is connected to two elements *White* and *Black*.

<concept> *c* </concept>	Where *c* represents the name of the concept.
<language> *l* </language>	Where *l* represents the language in which the words are defined.
<word>$w_1,w_2 \ldots w_n$</word>	Where $w_1,w_2 \ldots w_n$ represent the words that describe the concept *c*.
<arity> *a* </arity>	Where *a* is a positive number that describes the arity

	of the concept *c*.
<relation> *n* = *v* </relation>	Where *n* represents the name and *v* represents the value of the relation. The value *n* and *v* are concepts. The *v* can be a list if the relation has more of a value.
<part> *c* </part>	Concept that contains this relation *is part of* the concept *c*.
<member> *c* </member>	Concept that contains this relation *is member of* the concept *c*.
<subset> *c* </subset>	Concept that contains this relation *is subset of* the concept *c*.
<type> *c* </type>	Concept that contains this relation *is a type of* the concept *c*.

Table. 1. Labels used in the OM Notation.

The relations are properties or characteristics of the node or concept where they are defined. An example of this called relation *Eat* is shown in figure 1.
Other relations exist, such as hyponymous relation, that are expressed through concepts nested. For example, *plant* is a subset of *physical_object*.

```
<concept>thing
    <language> English <word>thing, something, object, entity </word> </Language>
    <concept>physical_object
        <language> English <word> concrete_object, physical_object</word> </Language>
        <concept>plant
            <language> English <word>plant, tree</word> </Language>
            <concept>fruit
                <language> English <word>fruit, citric</word> </Language>
            </concept>
        </concept>
        <concept>man
            <language> English
            <word>man </word></Language>
            <relation>eats=tropical_fruit, citrus</relation>
            <relation>Partition=age {0<age<=1 : baby; 1<age<=10 : child;10<age<=17 : puberty; 17<age<=29 :
            young; 29<age<=59 : mature; age>59 : old;}</relation>
        </concept>
    </concept>
    <concept>abstract_object
        <language> English <word>imaginary object, abstract thing</word> </Language>
        <concept>soul
            <language> English <word>soul, spirit</word> </Language>
        </concept>
    </concept>
</concept>
```

Figure 2 Representation of an ontology in OM Notation.

Relations are Implicit and Explicit. The Implicit relation indicates a structural relation (parent-son). For example, the relation "part of" exists between holonymous and meronymous sets. Ontology with nodes and relations is shown in the figure 2. The circles and arrows denote nodes and relations respectively. A set is holonymous of another when its semantic notion represents the whole of an object; therefore *bicycle* is holonymous of *handle-bar*. A set is Meronymous when it represents a part of an object; therefore *handle-bar* is meronymous of *bicycle*.

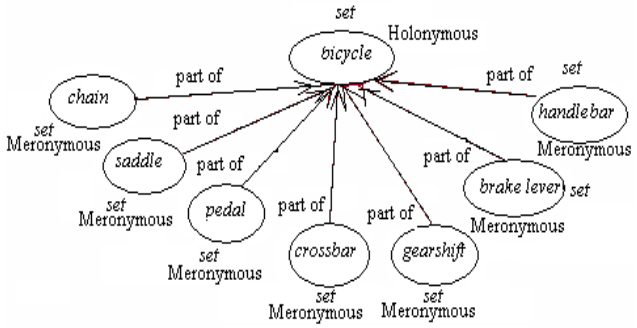

Figure 2. Graphical representation of ontology with "part of" relation.

Other implicit relations exist, such as: Hyperonymous and Hyponymous, where a term is hyperonymous of another if meaning of the first one includes the second one. The set *Port* as hyperonymous of *Port of Mexico* is an example of this, because the meaning of *Port of Mexico* is included in *Port*; the relation is represented as <subset>.
Other implicit relation is "type of". This is the same that "subset". It is not shown in this paper.

The explicit relation provide more semantic to the nodes, describing properties, characteristics or actions that identify a concept of others. For example, the relation *activity* between *Port of Salina Cruz* and *Commercial activity*, *Tourist activity* and *Fishing activity*. Other examples are presented here:

1. *Apple* Color *Yellow*
2. *Apple* Form *Round*
3. *Cat* Drinks *Milk*
4. *Turtle* Lives *Intertropical zone*
5. *Oaxaca* Economy *Economy of Oaxaca*

2.1. Relation of type Partition

A partition is a collection of sets such that whatever two elements of this collection are mutually exclusive and all of them are collectively exhaustive.
Nowadays, partitions are not represented in languages [4], [10] and [12]. Partitions are represented of the following way:
Partition=$nomPart\{range_1{:}value_1;range_2{:}value_n range_1{:}value_n\}$
Where *nomPart* represents the name of partition, *range* is the characteristic that distinguishes this set of the other sets of the partition. This element can be an interval, a

list of elements or simply a character. The *value* represents the value of the range, the name of interval, list or character. This can be a node or concept.
For example:
<relation>
Partition = *age* {*0<age<=1*: *baby*;*1<age<=10*: *child*;*10<age<=13*: *tennager*;*13<age<18*: *young*;*18<=age<40*: *adult*;*40<=age<60*: *mature*;*60<=age*: *old*;}
</relation>
The graphic representation of a partition is shown in figure 3; the small, black circle represents the partition.

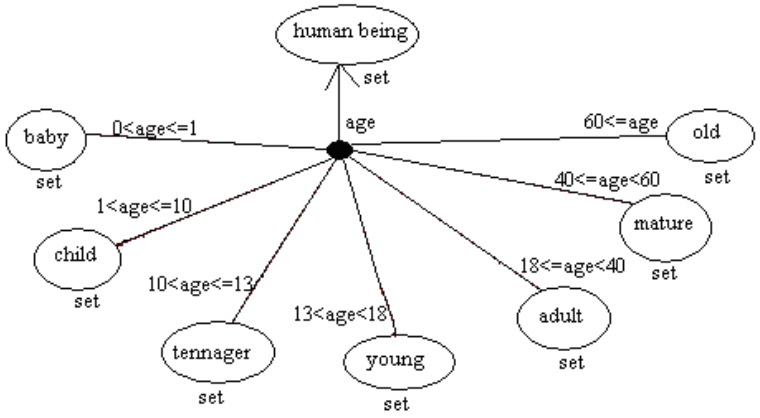

Figure 3 Graphical representation of a partition

Partitions are a form of classifying a concept, to be able to infer on this later. The inference of partitions is not included in this paper.

2.2. A concept can be a relation

The relations are represented in following form: r (C_{name}, C_{value})
Where, r represents the name of relation, C_{name} represents the name of the concept of the relation, C_{value} represents the concept value of the relation. An example is:
Mother (*Mary Ball Washington*, *George Washington*) *Mary Ball Washington* is *Mother* of *George Washington*, but *Mother* can be a concept that contains more information of the meaning of *Mother* and other concepts related to this. Other contributions exist but will no be explained in this brief space.

3. OM Algorithm for automatic merging of ontologies

Nowadays, several works that merging ontologies need the intervention the user for this important process, some of them are: [6], [9], [11], [13] and [14]. This algorithm is the unique (up to this days) that merges ontologies in an automatic form. The pro-

cess to merge two ontologies, consists of the following general steps: Given 3 ontologies A, B and C, given concepts a, b and c that belongs to A, B and C respectively.

1. $a \in$ A, to obtain $com(a,B)$
2. if $com(a,B) > 0$
   $b \in B$ with better $com(a,B)$
   to merge a, b obtaining $c = ext(a,b_{relation})$

to obtain $c \in C$ to each pair (a,b) the resulting ontology is: $C = \{c\} \cup \{a: com(a,B) = 0\} \cup \{b:com(b,A) = 0\}$

The function *com*(*concept*, *ontology*) of the algorithm COM [1] is a similarity search function that takes the *concept* and looks for its more similar concept in the *ontology*, giving back the most similar *concept* and a *sv* (similar value) with value between 0 and 1.
The function *ext*(*concept*, $concept_{relation}$) of the OM Algorithm is the extension of concept that is obtained adding to this, new relations of $concept_{relation}$ to *concept* in *B* and those relations that are synonymous. In this step, the inconsistencies are detected between names and values of a relation. An inconsistency is a fact of the ontology *B* that contradicts a fact of the ontology *A*.
In the process of merging ontologies the following cases appear.

3.1. Verification of the arity in concept

The arity of a concept represents the number of values that the concept can take. If the concept takes only a value it is said that it is mono-valuated arity. For example, the arity of the concepts *Mother* and *Father* is mono-valuated; because a person can have a *Mother* and *Father* simultaneously.
A concept is a multi-valuated arity if this can take several values. For example, the *political position* that a person can carry out. OM Algorithm verifies the arity of concepts before copying the new relation in the resulting ontology. If this concept is a multi-valuated arity receives the new value; or else, tries to solve the problem using the Confusion [2] algorithm.

3.2. Union of a new relation

The union of a new relation in the resulting ontology implies the following:

a) The name and value of the relation in A are different from the name and value of the relation in B, that is to say; they are totally different concepts and they aren't synonymous.

b) The name and value of the relation in B are different from the name and value of the relation in A; that is to say, they are totally different concepts, they aren't synonymous.

3.3. Union of a relation with elements that are synonymous

In order to know if two concepts are synonymous, OM applies COM [1] Algorithm. This it gives back a message, a concept and a value of similarity. If the message is "Case B", the given back concept is considered synonymous; the value of the similarity must be bigger or equal to 0.8 and minor or equal to 1.
Given two ontologies A and B to form one third C ontology, give the relation in A: *it escaped with* (*José Arcadio*, *a gypsy*) and the relation in B: *fled with* (*José Arcadio*, *a gypsy*).
The function *com* (*it escaped with, B*) of COM [1] is applied. This function gives back "Case B" with the concept *fled with* and the value of similarity is 1 and then OM does not fuse both relations but it enriches the relation in A (because *it escaped with* and *fled with* are synonymous) with the new words and properties of B, copying the relation enriched to the resulting ontology C.

3.4. Confusion in the name of relations

During the copy of the relations of a concept, it's possible that the name of the relation in A was different from one in B more not the value from this. The confusion arises when both relations share the same value. The OM Algorithm looks for the synonymy between the names of relations; this is, applies COM [1] to the names of the involved concepts. This step is applied when names of relations are concepts. If COM [1] returns "Case B" then they are synonymous, otherwise they are not. There are other forms to find the synonymy between the names of relations, but because of lack of space they are not explained in this paper. If they are not synonymous OM solves the problem using Confusion [2]. For example:

Given a relation *r* in *A* with values: *Hydrology* (*Oaxaca*, *Main river of Oaxaca*).
Given a relation *r* in *B* with values: *River*(*Oaxaca*, *Main river of Oaxaca*)

A hierarchy of concepts is used where the names of the relations are represented. The figure 4 shows this hierarchy. In the hierarchy the number of levels is obtained. It is to say, the height of the tree is 2. The value of the Confusion [2] of using *River* instead of *Hydrology* is calculated, starting from the concept *River* and following a route up to *Hydrology*, counting the descendent levels and dividing the sum between the number of levels. In order to obtain the value of the confusion of using *Hydrology* instead of *River*, the descendent levels are added; that is to say, 1 is divided between 2. The result is 0.5.

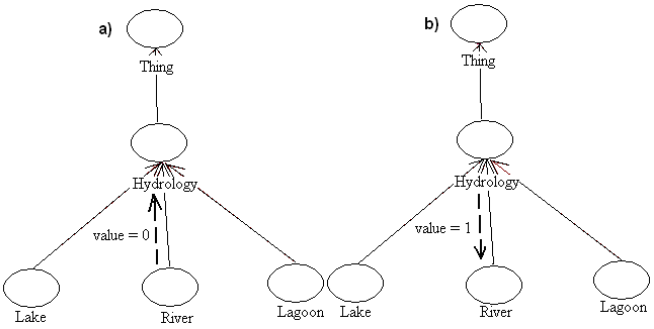

Figure 4 The Confusion of using *River* instead of *Hydrology* is 0. This is shown in a), and the Confusion of using *Hydrology* instead of *River* is 0.5. This is shown in b).

Finally OM chooses the smaller value of the confusion. In the example this is the name of relation in B; it is to say *River* (*Oaxaca*, *Main river of Oaxaca*).

3.5 Confusion in the value of the relations

If the Confusion [2] arises in the value of the relations, the arity of the name of relation is verified. For example:

Given the relation *r* in *A Birthplace* (*Benito Juárez*, *San Pablo Guelatao*)
Given the relation *r* in B: *Birthplace* (*Benito Juárez*, *México*)

The arity of *Birthplace* is mono-valuated, because it's not possible to be born at the same time in two different places, but the place can be specified. It's to say *San Pablo Guelatao* belongs to *Mexico*. Therefore, OM looks the synonymy of *San Pablo Guelatao* and *Mexico*. If this doesn't exist, OM apply Confusion [2], calculating firstly the height of the tree, the result is 5 according to what it shown in figure 5.

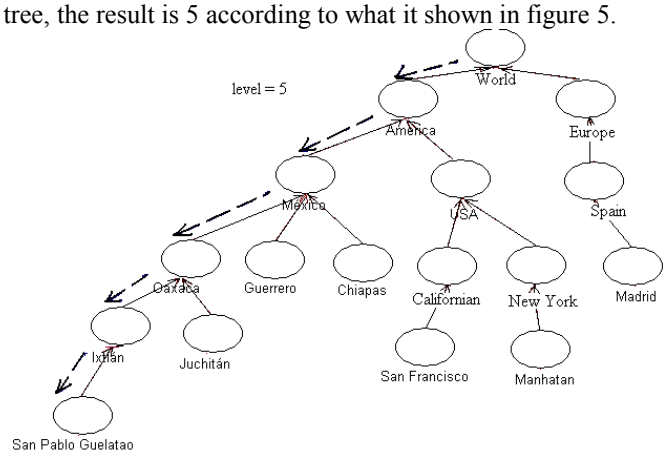

Figure 5. Graphical representation of the hierarchy that indicates the height of the tree.

Later, the value of Confusion using *Mexico* instead of *San Pablo Guelatao* is calculated, counting the number of descendent levels. The result is 3 divided between 5, obtaining the value of confusion 0.6. The value of the confusion of using *San Pablo Guelatao* instead of *Mexico* is obtained. This value is 0. Therefore OM decides to conserve the relation in A.

3.6. Union of partitions

The relations of type partition are an important contribution in OM Notation. The case that can appear is: in another ontology, OM finds a partition with the same name but with different classifications. OM analyzes the ranges and values of these. If they are different, OM adds the new partition in C like a new one.

3.7. Values of a relation

Given each list of values in a relation, OM verifies the presence of predecessors who cause redundancy in the data. For example, the following relation:
<relation> visited = `Istmo`, `Salina Cruz`, `Paris`, `France`, `Africa`</relation>
The analysis consists of verifying of sequential form each one of the values of the relation eliminating the predecessors of each concept in list. In this example `Istmo` is eliminated because `Salina Cruz` is member of the `Istmo` set. It's understood that if visited `Salina Cruz` then it visited `Istmo`.
If it's wanted to fuse two relations:
<relation> visited = Istmo, Francia, Frankford</relation> in O_A.
<relation> visited = Salina Cruz, Paris, Germany</relation> in O_B.
We would think that the result of the fusion would be:
<relation> visited = Istmo, Francia, Frankford, Salina Cruz, Paris, Germany</relation>
But OM would not return that fusion, since it compares the words of each one of the values of relation, if they are different compares the synonymy, if it does not exist then applies the algorithm of the Confusion to each one of the values of the relation and chooses the minus value of the confusion to fuse the relations. In such a way that the result would be:
<relation> visited = Frankford, Salina Cruz, Paris</relation>

3.8. Verification of redundant relations

During of fusion of ontologies, redundant relations are also copied. OM avoids that in the resulting ontology, redundant relations from a concept to another are made. The redundant relations arise when three concepts in C exist. For example, $c1_c$, $c2_c$ y $c3_c$ (c is the concept and the sub-index is the ontology to which it belongs) whose relations are the following: $c1_c$ is subset of $c2_c$, $c2_c$ is subset of $c3_c$ and $c1_c$ is subset of $c3_c$; the nested relation arises in: $c1_c$ is subset of $c3_c$ and OM eliminates it of ontology

C. The nested relations do not only exist in those of type <subset>, also in those of type <part> and <member>. Figure 6 shows 2 ontologies A and B that merge with each other to obtain C. The lines represent the similarity between the concept origin in A (where it leaves) towards the concept destiny in B (where it point the arrow). In the figure 6 it's possible to observe that in A the concept *Seed* have as preceding *Poppy* and in B this predecessor its *great-grandfather*.

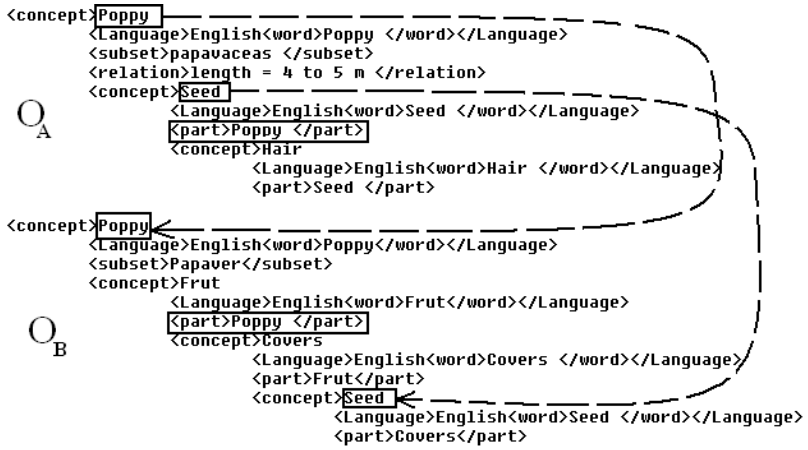

Figure 6 A and B ontologies with the relations in *Poppy* that it will generate nested relation.

In figure 7 the result of the merge in ontology C appears, where the relation nested between the concepts has been eliminated.

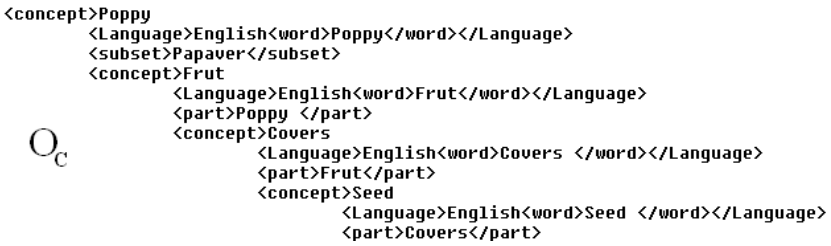

Figure 7 Representation of an ontology without the redundant relation.

3.9. Contributions of the OM Algorithm

1. Totally automatic, requires no human intervention.
2. It handles partitions as well as subsets.
3. It handles nodes (concepts) in an ontology that is described "shallowly" by just a word, a word phrase or a set of them.
4. Relation among nodes can also be concepts (nodes, that is).

5. It detects inconsistencies (contradictions) in the knowledge in ontology A versus the knowledge in B, using inconsistency measurements [7] and confusion [2].
6. It solves some of the contradictions detected in (5), through inconsistency measurement [7].

4. Tests

Tests have been merging ontologies in the domain of geographic zones, description of animals, biographies and description of tools, products and novels such as *Cien Años de Soledad* of Gabriel García Márquez. The ontologies were obtained manually from several documents describing that described, the same topic. The obtained ontologies were merged (automatically) by OM.
The validation of results has been made manually, although we are designing an automatic validation tool.

The work to be reported is a summary of the Ph D. thesis [3] of one of the authors, and uses COM, a software [1] that, given a concept *ca* in ontology *A* finds the most similar concept *cb* in ontology *B*, as well as its *similar value*.

Conclusions

A notation has been created to design ontologies. This notation presents some improvements made to languages of ontologies that exist in the Web. We also implemented OM, an algorithm to fuse ontologies; this algorithm does not process texts but it takes care to preserve the semantic of the merged ontologies. It detects the inconsistencies during the merge and it solves them. OM makes the fusion totally automatic. This is a great improvement to the fusion algorithms that are in the Web, since they perform the fusion in a semi-automatic form. It is to say, the user in them takes part in the important points of the fusion. OM notation and OM algorithm are part of the answer to the great necessity to make that the computer, as important entry point in the Web, can accumulate knowledge and make transactions of business without human intervention.

References

1. Adolfo Guzmán and Jesus Olivares, “Finding the Most Similar Concepts in two Different Ontologies”, *Lecture Notes in Artificial Intelligence* LNAI 2972, Springer-Verlag. 129-138. ISSN 0302-9743, 2004
2. Adolfo Guzmán and Sergey Levachkine, “Hierarchies Measuring Qualitative Variables”, Lecture Notes in Computer Science LNCS 2945, *Computational Linguistics and Intelligent Text Processing*, Springer-Verlag. 262-274, ISSN 0372-9743, 2004

3. Alma-Delia Cuevas-Rasgado. *Ontology Merging using semantic properties* Ph. D. thesis in progress. CIC-IPN, Mexico.
4. Asunción. Pérez , and Mary Carmen Suárez, *Evaluation of RDF[S] and DAML+OIL Import/Export Services within Ontology Platforms*. LNAI 2972, 109-118. 2004
5. D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinnes, P. F. Patel-Schneider, L. Andrea Stein, *DAML+OIL Reference description*, March 2001, W3C Note 18 December 2001, http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218
6. D. Dou, D. McDermott, and Peichen Qi. *Ontology Translation by Ontology Merging and Automated Reasing*, Yale University, Computer Science Departament New Haven, CT 06520.
7. Edith Adriana Jimenez Contreras. *Quantifying inconsistencies in sentences (facts) with symbolic values*. Ph. D. thesis in progress. CIC-IPN, México.
8. F. Manola, E. Miller. *RDF Primer. W3C Recommendation*. 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
9. G. Stumme, A. Maedche. *Ontology Merging for Federated ontologies on the semantic web*. Institute for Applied Computer Science and Formal Description Method [AIFB] University of Karlsruhe D-76128 Karlsruhe, Germany.
10. Knowledge Interchange Format. *Draft proposed*, American National Standard [dpANS] NCITS. T2/98-004.
11. L. Deborah McGuinness, R. Fikes, J. Rice and S. Wilder, *The Chimaera Ontology Environment Knowledge*, System Laboratory CommerceOne Stanford University, Stanford, CA Mountain View, CA.
12. M. K. Smith, Electronic Data System, C. Welty, IBM Research, D. L. McGuinnes, Stanford University, *OWL Web Ontology Language Guide*, W3C Recommendation 10 February 2004.
13. N. Fridman Now and A. Mark. *PROMPT: Algoritm and Tool for Automated Ontology Merging and Alignment*, Stanford Medical Informatics, Standford University, CA.
14. Y. Kalfoglou and M. Schorlemmer. *Information-Flow-based Ontology Mapping.* Advanced Knowledge Technologies [AKT] Departament of Electronics and Computer ScienceUniversity of Southamptom. Advanced Knowledge Technologies [AKT] Centre for Intelligent System and their Applications The University of Edinburgh.