Chapter XXII
A Language and Algorithm for Automatic Merging of Ontologies

Alma-Delia Cuevas-Rasgado
Instituto Politécnico Nacional, Mexico

Adolfo Guzman-Arenas
Instituto Politécnico Nacional, Mexico

ABSTRACT

Ontologies are becoming important repositories of information useful for business transactions and operations since they are amenable to knowledge processing using artificial intelligence techniques. They offer the potential of amassing large contents of relevant information, but until now the fusion or merging of ontologies, needed for knowledge buildup and its exploitation by machine, was done manually or through computer-aided ontology editors. Thus, attaining large ontologies was expensive and slow. This chapter offers a new, automatic method of joining two ontologies to obtain a third one. The method works well in spite of inconsistencies, redundancies, and different granularity of information.

INTRODUCTION

Computers are no longer isolated devices but they are important to the world-wide network that interchanges knowledge for business transactions. Nowadays, using the Internet to get data, information, and knowledge is a business need.

Most of the important information resources that businessmen require are available through the Internet. Here, machines face the problem of heterogeneous sources. The computer has a hard time finding whether two data representations refer to the same object (a *bill* can be a bank tender or an invoice)[1] because there are no suitable standards in knowledge representa-

tion. This chapter addresses this need of businesses and academia.

When businessmen demand answers that require access to several Internet data sources, they have to manually or mentally merge the acquired information in a reasonable way. It would be nice if a computer program helped in this very useful but tedious task. This chapter solves this problem, which has important implications (see the section on "Commercial Areas Ready to Exploit OM").

The Problem to Solve[2]

To merge two data sources in such a way that its common knowledge could be represented and more easily used in further tasks.

Computers represent the information in files, databases, text documents, lists, and so forth. Computer merging of information in databases or in semistructured data, has its own challenges, and will not be addressed here. Merging information stored in documents is done manually, since the computer does not "understand" what a document says. If the information is stored in spreadsheets, merging can be done by a computer-aided person who understands the contents of different cells and their units. Information can also be stored in ontologies and thus be subject to merging. So far, merging of ontologies has been done manually (see the section on "Ontology Merging") using an ontology editor.

Ontology

An ontology is a data structure where information is stored as nodes (representing concepts such as `hammer`, `printer`, `document`, appearing in this chapter in `Courier font`) and relations (representing restrictions among nodes, such as cuts, transcribes, or hair color, appearing in this chapter in Arial Narrow font, as in (`hammer` cuts `wood`), (`printer` transcribes `document`), Figure 9. Usually, the information it stores is "high level" and it is known as *knowledge*. For working purposes, we further restrict this definition to those data structures compliant with OM notation (*quo vide*).

Ontologies are useful when arbitrary relations need to be represented; one has more freedom to represent different types of concepts.

Current notations to represent ontologies are DAML+OIL (Connoly et al., 2001), RDF (Manola & Miller, 2004) and OWL (Bechnofer et al., 2004). These languages are a notable accomplishment, but some lack certain features:

- A relation can not be a concept. For instance, if color is a relation, it is difficult to relate color to other concepts (such as shape) by using other relations.
- Partitions (subsets with additional properties, see the section on "Contributions of OM Notation") can not be represented.

This chapter offers the *OM notation* to represent ontologies that solves above problems and better represents the semantics involved.

Ontology Merging

Realizing the importance of the problem to solve, different scientists have approached it. Previous works incudes CYC (Lenat & Guha, 1989), whose goal was to represent common sense knowledge in a gigantic hand-built ontology. CYC does not do merging. Prompt (Noy & Musen, 2000), Chimaera (McGuinness, Fikes, Rice, & Wilder, 2000), OntoMerge (Stumme & Maedche, 2002) and ISI (Loom) rely on the user to solve the most important problems found in the process, and are considered non automatic methods. FCA-Merge (Dou, McDermott, & Qi, 2002) and IF-Map (Kalfoglou & Schorlemmer, 2002) require consistent ontologies that are expressed in a formal notation employed in Formal Concept Analysis (Ganter, Stumme, & Wille, 2005) which limits their use. Hcone (Kotis, Vouros, & Stergiou, 2006) uses WordNet and a formal approach to ontology merging. Cuevas-Rasgado (2006) mentions additional previous works.

Our solution to the above problem is the *OM algorithm,* which performs the fusion in a robust[3] consistent,[4] complete,[5] and automatic[6] manner. When compared with fusion done by hand and with current computer-assisted methods, OM does "very good" (≈ 96%, Table 1), but manual methods may achieve 100% accuracy, depending on the user or expert that makes the correct choices, solves contradictions, and eliminates redundancies. OM also fused some ontologies expressed in current ontology languages, hand-translated to OM notation. The results are good (100%, Table 1) but care should be exercised: the ontologies merged contain only shallow information,[9] most are merely a taxonomy.

The section on "OM Notation" explains the OM notation, and the section on "OM Algorithm for Automatic Merging of Ontologies" the OM algorithm. The chapter concludes with examples.

Increased Yield Through Better Processing the Web Resources

This chapter describes important contributions towards the task of obtaining more benefits from Web resources: (1) the OM notation, (2) the OM algorithm, which automatically merges two ontologies, (3) a mapping algorithm among ontologies, called COM (see the section on "The Comparison Function COM"), that finds similarity among concepts belonging to different ontologies, and (4) the use and exploitation of a theory that measures the *confusion* (see the section on "Confusion") in using a symbolic value instead of another (the intended value). This theory solves some inconsistencies arising during the union of ontologies and lets the process proceed further.

In addition to being useful for businesses, ontology merging is an Artificial Intelligence (AI) tool that could harvest the knowledge (in a given area, say, oil production) available in the Web from documents in English and other natural languages, and (if they were translated to our ontology format)[7] automatically produce a new ontology that captures the (total, joint) knowledge available in all these documents. How? By joining consecutively ontology after ontology from those documents. See the section on "Suggestions for Further Work" for uses of this joint knowledge.

Issues, Problems, and Trends

One of the hard problems keeping AI people busy is how to provide the computer with a "deep" or "semantic" understanding of the information it is processing. In order to give it, for instance, the ability to answer complex, nontrivial queries about the information it has. One way is to construct a large ontology, understandable (processable) by machine, where mechanical reasoning could be achieved. Initially, a 10-year project (Lenat & Guha, 1989) was going to build by hand the common sense ontology. As time passed, numerous groups hand-crafted their own ontologies. People wondered how to map a concept from one of these ontologies to the closest concept in another ontology, and Guzman and Olivares (2004) were the first to solve this. OM uses and has improved their COM algorithm. See the section on "The Comparison Function COM". Inspired in COM, Cuevas-Rasgado (2006) reflected that automatic ontology merging was possible and desirable. This chapter presents her work. Until now, merging of ontologies was accomplished with the help of a user that resolved differences and made important decisions.

The trend is now clear: keep improving the merging algorithms, giving them access to "semantic sources of knowledge" (see the section, "Discussion"), and to knowledge previously processed (see the section on "Suggestions for Further Work"), in order to continue adding pieces of knowledge to growing ontology, which could be one day "the ontology of knowledge," much as Wikipedia is now the encyclopedia of knowledge.

Knowledge Support for OM

OM uses some built-in knowledge resources, which help to detect contradictions, find synonyms, and the like. These are:

1. Stop words (in, the, for, this, those, it, and, or…) are ignored form word phrases;

2. It takes into account words that change the meaning of a relation (without, except…);
3. Hierarchies (simplified ontologies, merely trees of concepts where each node is a concept or, if it is a set, its descendants must form a partition of it) represent a taxonomy of related terms, and are used to measure confusion (See the section on "Confusion"), and later can be used for synonym detection. Guzman and Levachkine (2004) explain how to build these hierarchies.

Future additions include using a stemmer, to find the root of words (love, lover, lovingly…), reliance on linguistic resources such as WordNet, use of a dictionary to find synonyms, homonyms, and so forth. The result of previous fusions could also be part of the built-in knowledge base for OM.

OM NOTATION

OM Notation represents ontologies through a structural design with XML-like labels, identifying the concepts and their relations. See Figure 1.

The label of each concept (such as thing) comes after <concept>; the language of the concept's definition (such as English) goes between <language> and </language>; the definition of the concept (such as concrete_object, physical_object) goes between <word> and </word>; the relations of the concept (such as eats) go between <relation> and </relation>. The description of a concept ends in </concept>. Nested concepts (such as physical_object within thing) indicate that physical_object is subordinate (or hyponym) of thing, the precise meaning of this subordination is indicated by <subset> thing </subset> (physical_object ⊂ thing)

Figure 1. Representation of an ontology in OM notation

```
<concept>thing
        <language> English <word>thing, something, object, entity </word> </Language>
        <concept>physical_object
                <language> English <word> concrete_object, physical_object</word> </Language>
                <subset> thing </subset>
                <concept>plant
                        <language> English <word>plant, tree</word> </Language>
                        <subset> physical_object </subset>
                        <concept>fruit
                                <language> English<word>fruit, citric</word> </Language>
                        </concept>
                </concept>
                <concept> human being,
                        <language> English
                        <subset> physical_object </subset>
                        <word> person, people, human being </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>
                <subset> thing </subset>
                <concept>soul
                    <language> English <word>soul, spirit</word> </Language>
                    <subset> abstract_object </subset>
                </concept>
        </concept>
</concept>
```

The relations expressed by nesting are called implicit relations. Currently, they are member of, part of, subset (represented in this chapter as ⊂), and part* ("one of my domain elements is part of one of my codomain elements," as in country part* continent). The other relations, such as eats, are called explicit. These are known elsewhere as properties or attributes of the concept. Cuevas-Resgado (2006) gives a complete description of the OM notation

In OM Notation, a relation can be n-ary; a relation relates nodes (concepts); a relation can be a concept (a node), too. For example, the `Zebra` concept has a Color relation that connects to two elements `White` and `Black`. Relations can be considered as properties or characteristics of the node or concept where they are defined. Nested concepts imply subordinate relations (see the caption of Figure 1).

Contributions of OM Notation

Most important are:

a. Ability to represent partitions. A *partition* of a set is a collection of subsets such that any two of them are mutually exclusive, and all are collectively exhaustive. OM can represent partitions, while current ontology languages (DAML, RDF, OWL) can not. For instance, not only `male _ person` and `female _ person` are subsets of `person`, they are a partition of `person`. Alternatively, the gender of a `person` will tell us to which of the partitions `male _ person` or `female _ person` the person belongs.

b. A concept also can be a relation**.** Often, ontologies are represented as a graph $O = (C, R)$ consisting of two *disjoint* sets: C (nodes, or concepts) and R (edges, or relations).[8] Two disadvantages of this visually oriented approach are: all the relations are binary and a concept can not be a relation. In OM, it is possible[9] to add relations *to a relation*, to provide more semantics. For example, one can say `Mary Washington` mother of `George Washington` to indicate that Mary is Washington's mother, but mother_of can be a concept that contains more information, for instance, related to child_of by the relation inverse.[10]

c. OM's graphs are hypergraphs, since relations are n-ary.

OM ALGORITHM FOR AUTOMATIC MERGING OF ONTOLOGIES

This algorithm fuses two ontologies A and B, building a third ontology $C = A \cup B$ containing the information in A, plus the information in B not contained in A, without repetitions (redundancies) or contradictions.

The information in B not contained in A can be: (1) new nodes, for instance B contains information about dinosaurs, which A lacks; (2) new relations, for instance, B knows that Gabriel García Márquez wrote *The Colonel has Nobody to write to him*, in addition to *One Hundred Years of Loneliness*, already known to A; (3) improved or more precise relations, for instance A knows that Abraham Lincoln was born in United States, while B knows that Lincoln was born in Kentucky; (4) new synonyms in B for current nodes in A enrich A; and (5) relations can be better defined in B, for instance B has a better description of lend money to than A. Thus, the addition of B to A is "carefully done" by OM.

OM proceeds and Cuevas-Rasgado (2006) gives more details:

1. **C ← A**. Ontology A is copied into C.
2. **Search in B each concept C_C of C**.[11] This step describes the *deep copy* of a concept.[12] At the start of the search, concept C_C is the root of ontology C. Then, C_C will be each of the descendants of C_C, and so on, so that each of the nodes of ontology C will be visited by C_C.[13] For each C_C, COM looks for the concept that best resembles C_C in B, such concept is called the *most similar concept* in B to C_C, or *cms*. Two cases exist:
   a. If **C_C has a most similar concept *cms*** in B, then:

i. Relations that are synonyms (see the section on "Knowledge Support for OM") are enriched. To enrich a concept C_C is to add to its definition the new words that are in the definition of *cms,* when C_C and *cms* are synonyms. [14]

ii. New relations (including partitions) that *cms* has in *B*, are added to C_C.
   1. For each added relation, concepts related by that relation and not present in *C* are copied to *C*. Example: if *cms* color `red` and concept `red` is not in *C*, it is copied to *C*, together with its ascendants who are not present in *C*. In this step we copy partitions of C_C, if they exist, since they are relations, too.

iii. Inconsistencies between the relations of C_C and those of *cms* are detected.
   1. If it is possible, by using confusion (see the section on "Confusion"), to resolve the inconsistency, the correct concepts are added to *C*. For instance, in Contribution g, ontology *A* says `AcmeCorp` incorporated_in `Maryland` and *B* says `AcmeCorp` incorporated_in `USA`. Since *incorporated_in* can only have a single value, a contradiction is detected and solved, thus AcmeCorp incorporated_in Maryland is added to C_C.
   2. When the inconsistency can not be solved, OM rejects the contradicting information in *B*, and C_C keeps the original relation coming from *A*.[21]

iv. Concepts that are descendants of *cms* not present in *C* are copied to *C*, in a superficial manner.[11]

b. **C_C can not find in B a good resemblance**. That is, *B* contains no object *cms* resembling C_C.
   i. Take the next descendant of C_C, which will become the new C_C.
   ii. Go to step 2 until all the nodes of *C* are visited (including the new nodes that were being superficially added by OM).

The Comparison Function COM

Four cases are used to find *cms* = COM(C_C, *B*), the most similar concept in ontology *B* to the concept C_C in ontology *C*. Guzman and Olivares (2004) explain COM in detail.

CASE A. A concept C_B having a definition similar to the definition of C_C is found in *B*, and the parent[o] of C_B has a definition matching the definition of the parent[15] of C_C. In this case, COM returns *cms* = C_B. See figure 2.

CASE B. C_C does not find a similar concept in *B* matching C_C, but the parent (let us call it P_C) of C_C finds a match with a node P_B in *B*. Then, we search for a son (or grandson, or nephew) of P_B having most of its relations match (using COM) with those of C_C. If such candidate has also descendants, do they coincide with the descendants of C_C? The best candidate becomes *cms.* If no candidate is good enough, COM returns *cms* = "son of P_B" (meaning that C_C must be some son of node P_B, unknown to *B*). In this case, OM will try to merge P_B with P_C.

CASE C. C_C finds a match C_B in *B*, but the parents P_C and P_B (of C_C and C_B) do not match. COM verifies if most of the relations of C_C correspond to those in the candidate, and if most of the descendants of C_C match those of the candidate C_B. That being the case, it returns the C_B with the best match as

cms. If only some properties of C_C and C_B match, COM returns *cms* = "probably C_B." OM treats this (arbitrarily) as a match between C_C and C_B.[16] If few or no properties of any candidate match, COM returns "no match" (Figure 3).

CASE D. C_C does not find a match in *B*, and neither its parent P_C does. COM returns "no match."

Confusion

I ask for a *European car*, and I get a *German car*. Is there an error? Now, I ask for a *German car*, and a *European car* comes. Can we measure this error? Can we systematize or organize these values? Hierarchies of symbolic values allow measuring the similarity between these values, and the error when one is used instead of another (the intended or real value). This measurement is accomplished by the theory of confusion (Guzman & Levachkine, 2004) and the function *conf*, which is used by OM to solve some inconsistencies.

Confusion, contradiction, or inconsistency arise when a concept in *A* has a relation that is incompatible, contradicts or negates other relation of the same concept in *B*. For instance, `Earth` in *A* has shape `flat`; and in *B* `Earth` has the relation shape `round`. Contradiction arises from two relations: in our example, the shapes are not the same, are inconsistent since shape can only have a single value.

Because OM must copy concepts keeping the semantics of the sources in the result, and both semantics are incompatible, a contradiction is detected. It is not possible to keep both meanings because they are inconsistent.[17] To solve some of these inconsistencies, OM uses the theory of confusion.

Function CONF(*r*, *s*), called the *absolute confusion,* computes the confusion that occurs when object *r* is used instead of object *s*, as follows:

CONF(*r*, *r*) = CONF(*r*, *s*) = 0 when *s* is some ascendant of *r*;
CONF(*r*, *s*) = 1 + CONF(*r*, father_of(*s*)) otherwise.

CONF is the number of descending links when one travels from *r* (the used value) to *s* (the intended or real value), in the hierarchy to which *r* and *s* belong.

Absolute confusion CONF returns an integer between 0 and *h*, where *h* is the height of the hierarchy (Figure 4). CONF is granularity-dependant, since its value changes merely by adding nodes between the root of the hierarchy and *s*. To make it insensitive to this, we normalize it by dividing into *h*, the height of the hierarchy, thus:

Definition.
conf(*r*, *s*), the confusion when using *r* instead of *s,* is:
conf(*r*, *s*) = CONF(*r*, s) / *h*
conf returns a number between 0 and 1.
Example: conf(Hydrology, river) = 0.2 (Figure 4).

OM uses conf, whereas Guzman and Levachkine (2004) describe CONF. Confusion is not a distance. In general, conf (*a, b*) ≠ conf(*b, a*). conf(*r*, *s*) is domain-dependant, as reflected by the hierarchy used to compute it.

Besides *confusion,* there are many forms to measure similarity or likeness between qualitative values *r* and *s*. For instance, seeing how far apart in Wordnet (wordnet.princeton.edu/) are the synsets where *r* and *s* lie, or comparing their definitions (or glosses) in a dictionary. OM uses confusion due to its asymmetry, but it could easily adapt or add some other similarity functions. A complete discussion of similarity is in Guzman and Levachkine (2004).

Contributions of the OM (Ontology Merging) Algorithm:

a. *It is totally automatic*, requiring no human intervention.
b. *It handles partitions* as well as subsets (explained in "Contributions of OM Notation").
c. *It handles concepts in an ontology that are described "shallowly"* by just a word, a word phrase or a set of them (see footnote 9).

d. *Relations among nodes can also be concepts*, as explained in "Contributions of OM Notation."

e. With the help of COM, OM takes into account:

1. *Synonyms.* Example: If *A* contains `boat` ("boat", "ship") ⊂ `vessel`, and *B* contains `dinghy` ("skipper", "boat") ⊂ `vessel`, then *C* will contain `boat` ("boat", "ship", "skipper") ⊂ `vessel`. Other example: In figure 2, `method` in *A* matches `procedure` in *B* and the parent (of `method`) `technique` in *A* matches the parent of `procedure` in *B*. Thus, this is case A of COM. Other example is found in part c of example 3; see Figure 9.
2. *Homonyms*. If *A* contains fly ⊂ `insect` ⊂ `animal` and *B* contains fly ⊂ `navigate`, then *C* will contain fly ⊂ `insect` ⊂ `animal` and fly ⊂ `navigate`, that is, OM recognizes (Case C of COM) two different concepts with the same name. Another example: if *A* contains the concept `printer` ⊂ `company` and *B* contains `printer` ⊂ `computer peripheral`, then *C* will contain both: `printer` ⊂ `company` and `printer` ⊂ `computer peripheral`, that is, OM recognizes both concepts as different, although they have the same name `printer` (Figure 11).
3. *Synonyms when considering their properties*. If *A* has `maize` ("maize") ⊂ `cereal`, color `yellow`, size `1cm`, contains `hydrocarbons` and *B* has `corn` ("corn") ⊂ `cereal`, color `yellow`, size `0.5inch`, contains `carbohydrates`, then case B of COM will correctly identify `maize` and `corn` as synonyms, and thus will contain `maize` ("maize", "corn") ⊂ `cereal`, color `yellow`, size `1cm` (`0.5inch`), contains `hydrocarbons` ("hydrocarbon", "carbohydrates"). That is, `corn` and `maize` have many properties equal or similar (by recursive use of COM). See Figure 3.
4. *New knowledge.* If one ontology knows nothing about dinosaurs, and the other has some concepts about them, then *C* will contain each ontology's unique knowledge, appropriately referring to knowledge common to both ontologies, such as "legs" or "fly."

Figure 2. Case A of COM

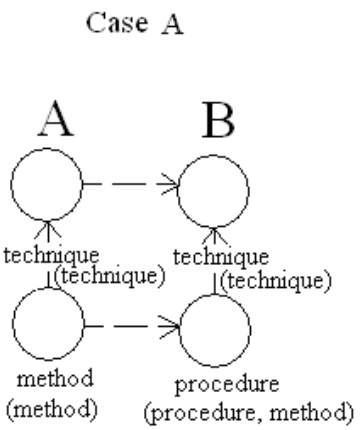
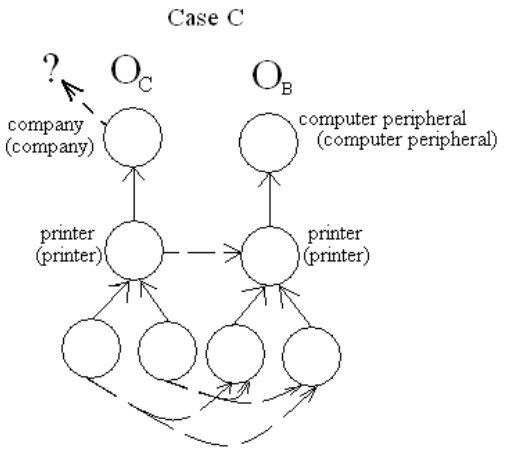


Figure 3. Case C of COM



5. *Other cases* where the knowledge in each ontology is properly taken into account are discussed in "Using OM. Examples," and by Cuevas-Rasgado (2006).

f. *OM avoids placement of redundant relations.* If *A* contains `lemon ⊂ fruit`, and *B* contains `lemon ⊂ citric ⊂ fruit`, then the resulting merged ontology *C* will contain `lemon ⊂ citric ⊂ fruit`, finding that *A*'s knowledge (`lemon ⊂ fruit`) is redundant.

g. *The OM algorithm detects inconsistencies* (contradictions) in the knowledge in *A* vs. the knowledge in *B*, using inconsistency measurements (Jimenez, n.d.) and confusion. An example where inconsistency is detected and solved is: Let *A* contain `AcmeCorp` incorporated_in `Maryland` and `incorporated _ in` arity `1`; let *B* contain `AcmeCorp` incorporated_in `USA`. OM detects an (apparent) inconsistency between Maryland and USA (two different concepts), which is solved by conf because Maryland is part of USA, conf(Maryland, USA)=0. Then, OM stores in *C* `AcmeCorp` incorporated_in `Maryland` (but it does not store in *C* `AcmeCorp` incorporated_in `USA`). Nevertheless, when trying to merge *A* with *D* which contains `AcmeCorp` incorporated_in `France`, OM will detect a contradiction, since the confusion between Maryland and France is large, and incorporated_in is single-valued. Unable to solve this contradiction, OM keeps in *C* the knowledge coming from *A*.[20]

h. *Expunging redundant values*. If *A* contains George_Washington visited (Paris, Africa, Madrid, Maryland) and *B* contains George_Washington visited (France, Morocco, Spain, USA, Argentina), then OM uses confusion to prune *C* to contain George_Washington visited (Paris, Morocco, Madrid, Maryland, Argentina). *Warning:* In the presence of symbolic values (places visited, in the example) at different hierarchy levels, selecting the most specific values may work, but there are other cases where the more general values are preferred. More knowledge is needed for OM to always solve correctly this case. See the section on "Suggestions for Further Work."

i. Cuevas-Rasgado (2006) provides *other heuristics* and rules that fortify OM.

Commercial Areas Ready to Exploit OM

OM enables the automatic development of larger and better ontologies. Also, with OM it is possible (but see footnote 7) to generate on-demand ontologies, tailored to the application needs.

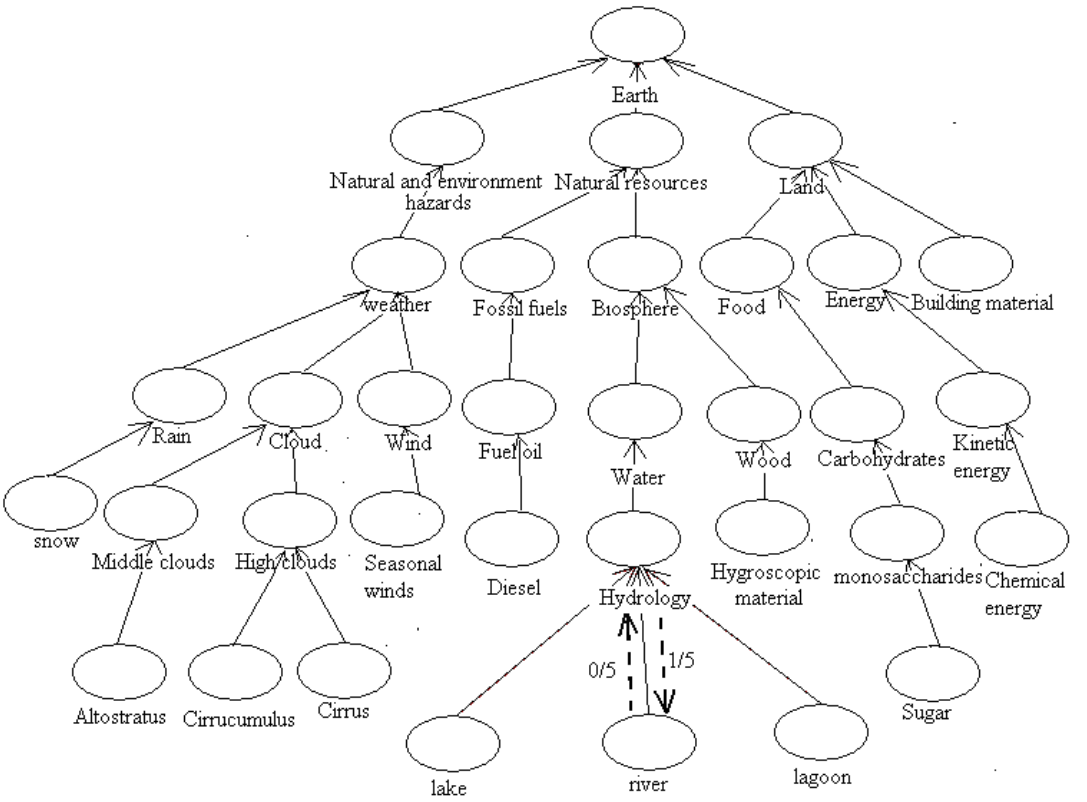
Ontology merging is at its infancy (see the section on "Suggestions for Further Work"). Its promise is the automatic acquisition of relevant knowledge. How can this help a business?

- Discovery of new markets. A glass factory in Indonesia may discover that their small glasses could be used in Mexico to drink tequila.
- Market trends. How many newspaper job ads demand a manufacturing engineer? How many require persons speaking Cantonese? (now done through text mining).
- Business intelligence. Mexico has large oil deposits in semifractured strata. How are other nations exploiting similar beds? (now done through word search of documents).
- Product improvement. Japanese consumers pay dearly for a fruit similar to a prickly pear, but without seeds. Can Jalisco adapt its prickly pears to this market?
- Electronic commerce.
- Public relations monitoring. What is New Yorkers' perception about the occupation of Irak? And Australian citizens' perception? (now done through polls).

Additional areas where OM can be productively used are: [18]

Semantic Web: Crawlers need to understand[19] large amounts of Web-available information. Central to this understanding is the assimilation of new

Figure 4. Solving contradictions. conf(river, Hidrology)= 0 whereas conf(Hidrology, river)= 0.2



information in ways consistent with already acquired knowledge. Use: to answer non-trivial questions (see "Suggestions for Further Work") needing multiple Web sources.

Electronic Commerce: Agent *A* can enrich its ontology (in order to acquire synonyms for its products, to facilitate finding new customers, suppliers or uses of its products) by joining its ontology with suitable ontologies *B*, *C*, ... Such "enriched agent" will understand better the queries and needs from other agents (or human beings) that may acquire products from *A*.

Virtual learning: Virtual book *A* in International Finance can merge its ontology with the ontology of another virtual book dealing with the same or near-by topic. The enriched ontology will be better suited for students learning from *A*. Also, *A* can join its ontology with a "predecessor" ontology from other book dealing with Economics. This helps students to refresh previous concepts.

USING OM: EXAMPLES

Figures 5 and 6 show only relevant parts of large ontologies *A*, *B* and *C*.

Figure 5. Ontology A has deeper knowledge about `hammer` *than ontology B*

```
<concept>tool
        <Language>english<word>tool</word> </Language>
        <subset>thing </subset>
        <concept>hammer
                <Language>english<word>hammer</word> </Language>
                <subset>tool </subset>
                <concept>carpenter hammer
                        <Language>english<word>carpenter hammer</word> </Language>
                        <subset>hammer </subset>
                </concept>
                <concept>blacksmith hammer
                        <Language>english<word>blacksmith hammer</word></Language>
                        <member>hammer </member>
                </concept>
        </concept>
        <concept>nail
                <Language>english<word>nail </word></Language>
                <subset> tool</subset>
        </concept>
</concept>
```

Example 1: Ontology Merging in spite of the Generality or Specificity of Contents

Here we merge two ontologies of businesses that sell tools for handcrafts.

Ontology *A* describes `hammer` with two sons: `carpenter hammer` and `blacksmith hammer` (Figure 5). Ontology *B* (Figure 6) contains a more general description of `hammer`. During the merging of *A* and *B*, OM detects that COM matches `hammer` in *A*, and its two sons, with the (unique) `hammer` in *B* (Figure 6).

Figure 7 presents concepts of *A* that have matched with those of *B* and vice versa. *A*'s `hammer` has matched with *B*'s `hammer`. When OM complements the words and properties of `hammer`, it copies the brothers of *B*'s `hammer` to *C*, but before that copying, it searches each of these brothers in *A*.

Figure 6. Ontology B has concepts screw and saw, inexistent in A

```
<concept>tool
        <Language>english<word>tool</word> </Language>
        <subset>thing </subset>
        <concept>hammer
                <Language>english<word>hammer</word> </Language>
                <subset>tool </subset>
        </concept>
        <concept>screw
                <Language>english<word>screw</word></Language>
                <subset> tool</subset>
        </concept>
        <concept>saw
                <Language>english<word>saw </word></Language>
                <subset> tool</subset>
        </concept>
</concept>
```

Figure 7. Mapping between A and B. Hammer, carpenter hammer, blacksmith hammer and nail from A match with hammer in B (dotted lines). Heavy lines identify matches from B into A

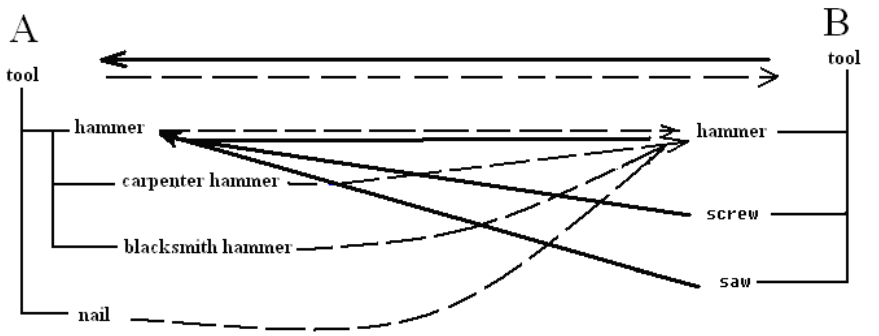


Figure 8. Result C for example 1. Here, C is symmetric: $A \cup B = B \cup A$. In the presence of contradictions, such symmetry may not hold (Cuevas, 2006; footnote 20)

```
<concept>tool
        <Language>english<word>tool</word> </Language>
        <subset>thing </subset>
        <concept>hammer
                <Language>english<word>hammer</word> </Language>
                <subset>tool </subset>
                <concept>carpenter hammer
                        <Language>english<word>carpenter hammer</word> </Language>
                        <subset>hammer </subset>
                </concept>
                <concept>blacksmith hammer
                        <Language>english<word>blacksmith hammer</word> </Language>
                        <subset>hammer </subset>
                </concept>
        </concept>
        <concept>nail
                <Language>english<word>nail </word></Language>
                <subset> tool</subset>
        </concept>
        <concept>screw
                <Language>english<word>screw </word></Language>
                <subset> tool</subset>
        </concept>
        <concept>saw
                <Language>english<word>saw </word></Language>
                <subset> tool</subset>
        </concept>
</concept>
```

For screw in *B*, even when COM answers hammer (from *A*) as the most similar concept in *A* (because the parents of screw and hammer coincide), OM compares their names: "hammer" and "screw". Being different, it considers screw as a new son of tool in *A*, and it copies screw into the merged result as a new node. The same happens to concept saw in *B*, and to carpenter hammer, blacksmith hammer, and nail in *A* (which are found by COM to be similar to hammer in *B*): they all go to *C*, Figure 8.

Example 2: Merging Ontologies with Mutually Inconsistent Knowledge

Differences in *A* and *B*'s knowledge arise from repetitions, reference to the same facts through diverse words, different level of details, type of description, and contradictions. For instance, *B* contains: veteran John Nash Sr. was born in Bluefield, while *A* contains: mathematician John Forbes Nash was born in West Virginia. Both ontologies duplicate some information (Nash's birthplace), different expressions (veteran / mathematician), different level of details (Bluefield / West Virginia), and contradictions (John Nash Sr. / John Forbes Nash). A person will have in her mind a consistent combination of information: John Nash Sr. and John Forbes Nash are not the same person, or perhaps they *are* the same. If she knows them she may deduce that one is the son of the other. We solve these problems everyday via *common sense knowledge* and previously acquired information. This is not so easy for computers, since they lack everyday's knowledge and usually they don't use, as OM, a previous knowledge base (See "Knowledge Support for OM"). Also, OM deals with inconsistency by measuring (step 2.a.iii of its algorithm) conf(Bluefield, West Virginia). [20]

Example 3: Joining Partitions, Synonym Identification

Numbers in Figure 9 match those below, for easy identification.

1. Copying new partitions. *B* has one partition: printing technology. *A* has two partitions: types and methods of image creation. Thus, printing technology is added to *C* (thin lines in Figure 9).
2. Copying concepts. procedure in *B* is copied to *C*, because it is not found in *A*. Its ascendant (not shown in Figure 9) is also copied to *C*.
3. Change into a (full) concept. Synonym identification. Adding more semantics. Relation method in *A* is copied to *C*; then, procedure in *B* is identified as a synonym of method, so method in *C* changes to procedure. In addition, procedure is a concept in *B* (it was just a phrase in *A*)[9], so it becomes a full concept in *C*. Finally, new semantics is added to procedure in *C* by adding to its definition print striking to the paper with small pieces from *B*.

Example 4: Numbers in Figure 10 Match Those Below

4. Removing redundant relations. In *A*, liquid inkjet printer ⊂ printer, whereas in *B* liquid inkjet printer ⊂ non-impact printer ⊂ printer. Adding both to *C* would make liquid inkjet printer to have two ascendants: printer and non-impact printer. OM detects the redundancy liquid inkjet printer ⊂ printer and expunges it from *C*, to keep only liquid inkjet printer ⊂ non-impact printer ⊂ printer.
5. Comparing relations (Figure 10). The relations method in *B* and its method is in *A* are considered to be the same, because connectors and, or, its, and so forth, are ignored (see the section on "Knowledge Support for OM").

Example 5: Homonyms

Concepts printer in *A* (Figure 11) and in *B* have the same syntax, but different semantics. OM finds them different, as explained in Example 1.

Figure 9. Relations method in A and procedure in B are synonyms, thus both of their definitions are added to node procedure in C

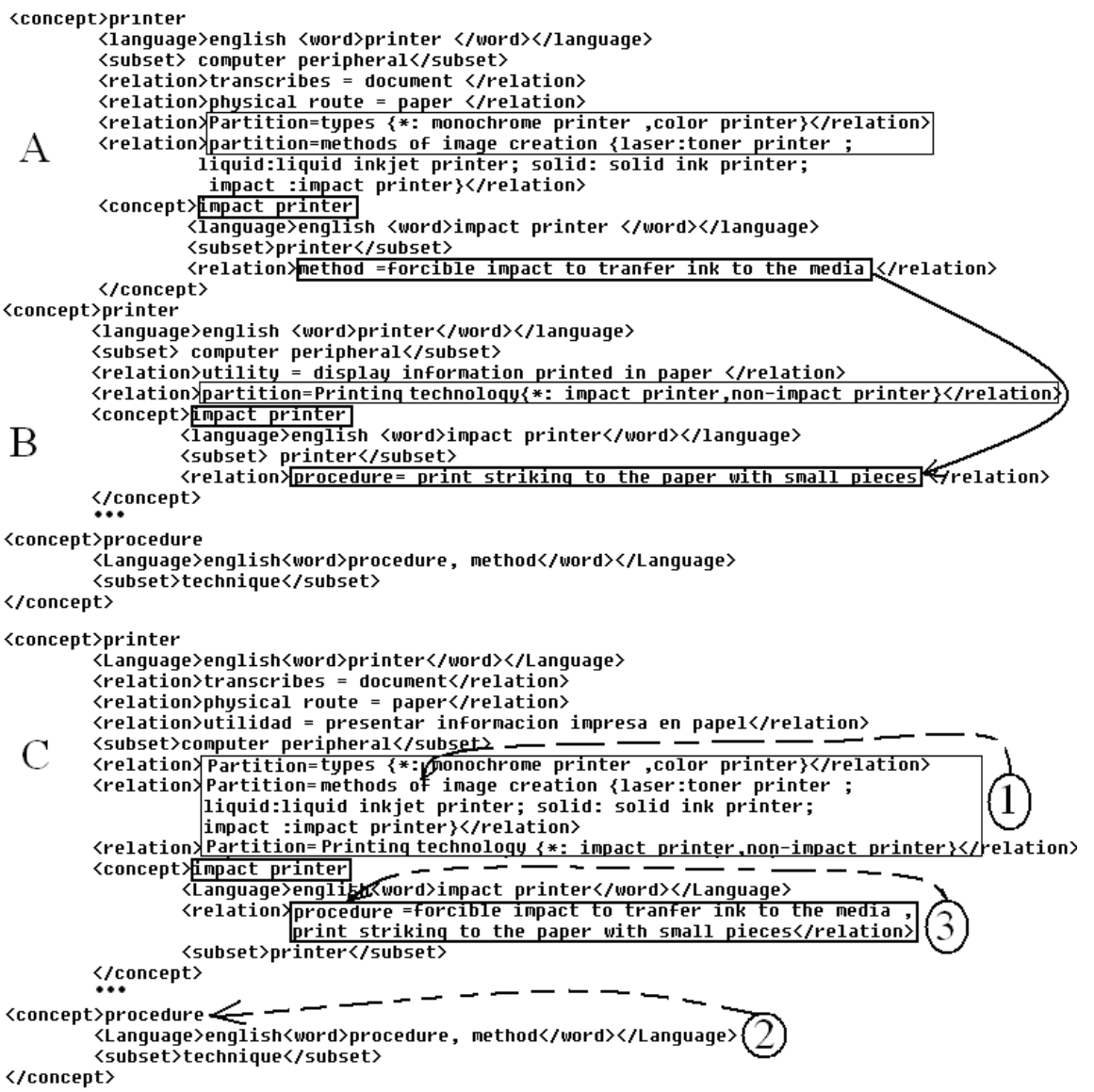
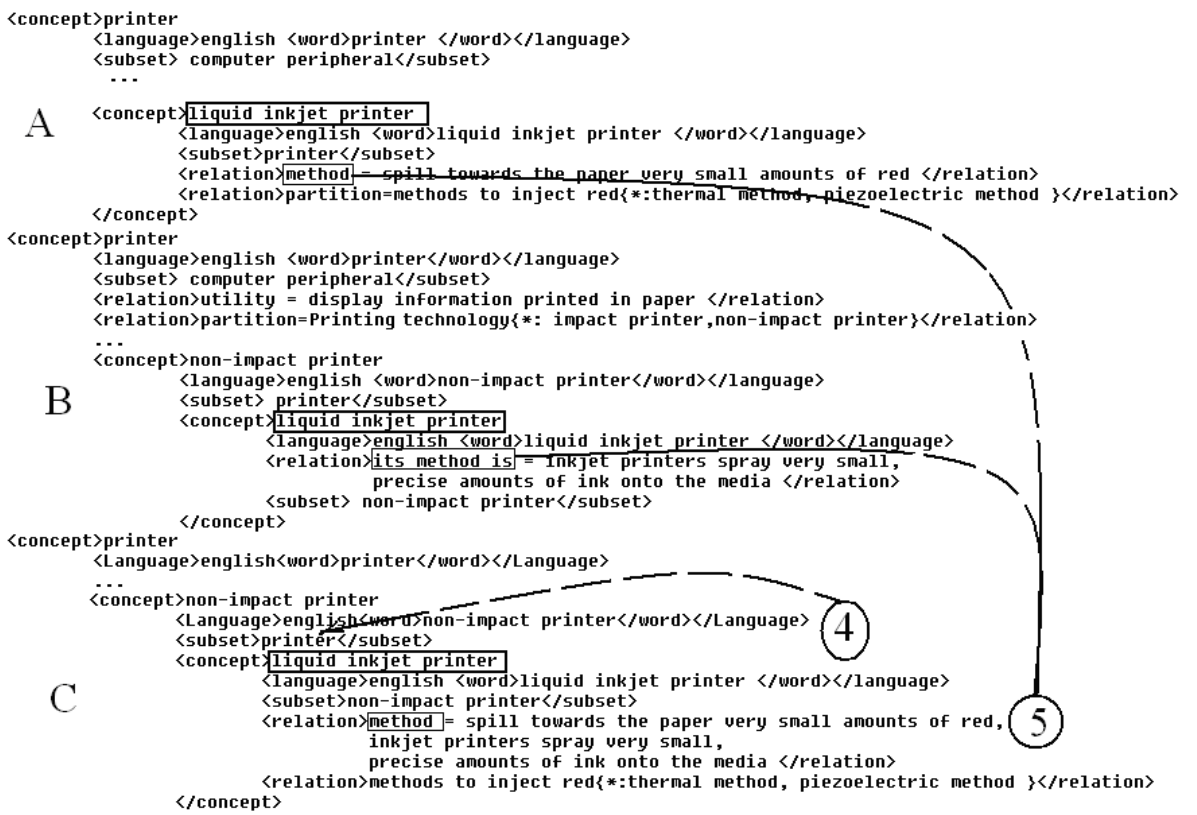


Figure 10. Relations method in A and its method is in B are the same, so they are merged in a single relation method in C (label 5)




```
<concept>printer
        <language>english <word>printer </word></language>
        <subset> computer peripheral</subset>
          ...
        <concept>liquid inkjet printer
                <language>english <word>liquid inkjet printer </word></language>
                <subset>printer</subset>
                <relation>method = spill towards the paper very small amounts of red </relation>
                <relation>partition=methods to inject red{*:thermal method, piezoelectric method }</relation>
        </concept>
<concept>printer
        <language>english <word>printer</word></language>
        <subset> computer peripheral</subset>
        <relation>utility = display information printed in paper </relation>
        <relation>partition=Printing technology{*: impact printer,non-impact printer}</relation>
        ...
        <concept>non-impact printer
                <language>english <word>non-impact printer</word></language>
                <subset> printer</subset>
                <concept>liquid inkjet printer
                        <language>english <word>liquid inkjet printer </word></language>
                        <relation>its method is = inkjet printers spray very small,
                                precise amounts of ink onto the media </relation>
                        <subset> non-impact printer</subset>
                </concept>
<concept>printer
        <Language>english<word>printer</word></Language>
        ...
        <concept>non-impact printer
                <Language>english<word>non-impact printer</word></Language>
                <subset>printer</subset>
                <concept>liquid inkjet printer
                        <language>english <word>liquid inkjet printer </word></language>
                        <subset>non-impact printer</subset>
                        <relation>method = spill towards the paper very small amounts of red,
                                inkjet printers spray very small,
                                precise amounts of ink onto the media </relation>
                        <relation>methods to inject red{*:thermal method, piezoelectric method }</relation>
                </concept>
```

Figure 11. Concepts printer in A and in B are found not to be the same, they both go to C (not shown) as two different concepts with the same name

```
<concept> company
        <Language>english<word>company</word></Language>
        <concept> printer
A               <Language>english<word>company</word></Language>
                <relation>operation = company that provides commercial printing services </relation>

<concept> computer peripheral
        <Language>english<word>computer peripheral</word></Language>
        <concept> printer
B               <Language>english<word>printer</word></Language>
                <relation>transcribes = document </relation>
```

Example 6: Promotion of Subsets to Partitions

Figures 12 and 13. In *A*, Etnolinguistic group of Oaxaca has subsets zoque set, ixcateco set, huave set and mixteco set, whereas in *B*, the same concept Etnolinguistic group of Oaxaca has a partition with the same elements that *A* has a subsets. Therefore, OM adds to *C* the partition from *B*. (A small error: OM fails to remove those elements as subsets from Etnolinguistic group of Oaxaca in *C*).

Example 7: Unsuccessful Promotion of Subset to Partition

It is not always possible to organize subsets into partitions. Figure 14 shows concept `stem` in *A* matching with `stem` in *B*. Thus, the partition `Color` belonging to `stem` in *B* is considered for copying to *C*. This partition has two elements: `Gray` and `Green`, which are searched in *A*. OM finds that `Gray` and `Green` are not descendants of `stem` in *A*.[21] OM finds them in `Color` in *A* (not shown in Figure 14), but they have

Figure 12. The partition Etnolinguistic in B is not in A, but before adding it to C, OM verifies that each of its elements (zoque set, ixcateco set…) are brothers in A and that no additional brother appears in A. These elements are all descendants of Etnolinguistic group of Oaxaca (thus, they are brothers) and no additional brother appears in A. Therefore, the partition Etnolinguistic from B is copied to concept Etnolinguistic group of Oaxaca in the resulting ontology C. Sizes of complete ontologies: A = 234 nodes; B = 117

A

```
<concept> Etnolinguistic  group of Mexico
        <Language>English<word> Etnolinguistic  group of Mexico</word></Language>
        <subset> Etnolinguistic  group </subset>
        <concept> Etnolinguistic  group of Oaxaca
                <Language>Ingles<word> Etnolinguistic  group of Oaxaca </word></Language>
                <subset> Etnolinguistic  group of Mexico</subset>
                <concept> zoque set
                        <Language>English<word> zoque set</word></Language>
                        <subset> Etnolinguistic  group of Oaxaca</subset>
                </concept>
                <concept> ixcateco set
                        <Language>English<word> ixcateco set</word></Language>
                        <subset> Etnolinguistic  group of Oaxaca</subset>
                </concept>
                <concept> huave set
                        <Language>English<word> huave set</word></Language>
                        <subset> Etnolinguistic  group of Oaxaca</subset>
                </concept>
                <concept> mixteco set
                        <Language>English<word> mixteco set</word></Language>
                        <subset> Etnolinguistic  group of Oaxaca</subset>
                </concept>
```

B

```
<concept> Etnolinguistic  group of Oaxaca
        <Language>Ingles<word> Etnolinguistic group  de Oaxaca</word></Language>
        <subset> Etnolinguistic  group of Mexico</subset>
        <relation>Partition = Etnolinguistic  {
                *:zoque set, ixcateco set, huave set, mixteco set }
        </relation>
```

Figure 13. The result C shows the partition Etnolinguistic added to the concept Etnolinguistic group of Oaxaca

C

```
<concept> Etnolinguistic  group of Mexicn
        <Language>English<word> Etnolinguistic  group of Mexico</word></Language>
        <subset> Etnolinguistic  group </subset>
        <concept> Etnolinguistic  group of Oaxaca
                <Language>Ingles<word> Etnolinguistic  group of Oaxaca </word></Language>
                <subset> Etnolinguistic  group of Mexico</subset>
                <relation>Partition = Etnolinguistic  {
                        *:zoque set, ixcateco set, huave set, mixteco set }
                </relation>
                <concept> zoque set
                        <Language>English<word> zoque set</word></Language>
                        <subset> Etnolinguistic  group of Oaxaca</subset>
                </concept>
                <concept> ixcateco set
                        <Language>English<word> ixcateco set</word></Language>
                        <subset> Etnolinguistic  group of Oaxaca</subset>
                </concept>
                <concept> huave set
                        <Language>English<word> huave set</word></Language>
                        <subset> Etnolinguistic  group of Oaxaca</subset>
                </concept>
                <concept> mixteco set
                        <Language>English<word> mixteco set</word></Language>
                        <subset> Etnolinguistic  group of Oaxaca</subset>
                </concept>
```

Figure 14. B has a partition Color, while A does not have it

A

```
<concept>stem
        <Language>English<word>stem </word></Language>
        <part>poppy </part>
        <relation>ramification = little graft </relation>
        <relation>forms = erect </relation>
        <relation>consists of = central ribbing </relation>
        <relation>forms = fine </relation>
</concept>
```

B

```
<concept>stem
        <Language>English<word>stem </word></Language>
        <part>poppy </part>
        <relation>Partition = Color {*:Gray, Green} </relation>
        <relation>it has = vein </relation>
        <relation>forms = smooth </relation>
</concept>
```

two additional brothers: `white` and `red`. Thus, they are not added to *C* (Figure 15) as a partition of `Color`, but as a partition of `stem`.

ADDITIONAL EXAMPLES FOR REAL-WORLD CASES

OM has been applied by Cuevas-Rasgado (2006) to ontologies derived from Web documents (see their URLs in the references), including:

Figure 15. The resulting ontology C for example 7. Stem is partitioned into Green (that is, green stem) and Gray (that is, gray stem) while color still has as subsets gray, green, white and red

C

```
<concept>stem
        <Language>English<word>stem </word></Language>
        <part>poppy </part>
        <relation>ramification = little graft </relation>
        <relation>forms = erect,fine,smooth </relation>
        <relation>consists of = central ribbing </relation>
        <relation>Partition = Color {*:Gray, Green} </relation>
        <relation>it has = vein </relation>
</concept>
...
<concept>Color
        <Language>English<word>Color </word></Language>
        <subset>Cosa </subset>
        <concept>gray
                <Language>Spanish<word>gray </word></Language>
                <subset>Color </subset>
        </concept>
        <concept>green
                <Language>English<word>green </word></Language>
                <subset>Color </subset>
        </concept>
        <concept>white
                <Language>English<word>white </word></Language>
                <subset>Color </subset>
        </concept>
        <concept>red
                <Language>English<word>red </word></Language>
                <subset>Color </subset>
        </concept>
</concept>
```

- Geographic zones: two different documents about Oaxaca
- Animals and flowers: two description of turtles, two of poppies
- Biographies: two about Benito Juárez, two about Newton
- Description of tools and products
- Novels: portions of *100 Years of Loneliness* (two different texts)

From these documents, ontologies were manually written in OM notation, obtaining two ontologies for each animal, flower, and so forth. Each pair of ontologies was merged (automatically) by OM. Validation of results (more at the section entitled "Discussion") has been made by comparing against a person's results, yielding Table 1.

CONCLUSION

As the world becomes a global village, businesses that do not adopt tools for automatic harvesting of knowledge disseminated through the Web will be at

Table 1. Performance of OM in some real-world examples

Table 1. Performance of OM in some real-world examples. How to read the table: Row *Neurotransmitter-Schizophrenia* says that OM merged the two ontologies in 2 seconds. 79 relations in B and 51 in A produced 127 relations in C, but the correct result (obtained by hand) contains 129 relations. OM missed 2 of 129 relations. *For concepts*, OM merged 56 concepts from B and 26 from A, producing 77 concepts in C, while the correct result contains 79 concepts. OM missed 2 of 79 concepts. The error is computed as (relations + concepts wrongly copied in the C produced by OM) / (relations + concepts in the correct C) = (2 + 2)/(129 + 79) = 4/208 = 0.019. Similarly, efficiency = 100 * (relations + concepts correctly copied to the C produced by OM) / (relations + concepts in the correct C) = 100*(127 +77)/208 = 98%

ONTOLOGIES	TIME	RELATIONS	CONCEPTS	ERR	% Effic
Turtles	4 sec.	6(B) ∪ 8(A) = 10(C). All were correctly copied..	35(B) ∪ 29(A) = 35(C). All nodes were correctly copied.	0	100
Martillo (Hammer)	6 sec.	30(B) ∪ 8(A) = 36(C). All were correctly copied.	33(B) ∪ 24(A) = 51(C). All nodes were correctly copied.	0	100
Amapola (Poppy)	14 sec.	20(B) ∪ 21(A) =37(C). All were correctly copied.	35(B) ∪ 34(A) = 58(C). All nodes were correctly copied.	0	100
100 Años de Soledad	10min.	283(B) ∪ 231(A) =420. * 432 (-12 from 432). 12 out of 432 relations were incorrectly omitted from C.	126 (B) ∪90 (A)=141 (C). * 149 (-8, 149). 8 out of 149 concepts were incorrectly omitted.	0.034	96
Oaxaca	5 min.	43(B) ∪ 61(A) =96(C). All relations were correctly copied	117(B) ∪ 234(A) =309(C). * 310 (- 1, 310). 1 out of 310 concepts were incorrectly omitted.	0.002	99.7
Neurotransmitter-Schizophrenia	2 sec.	79(B) ∪ 51(A) =127(C). * 129 (-2 from 129). 2 out of 129 relations were incorrectly omitted	56 (B) ∪26 (A)=77 (C). * 79 (- 2, 79). 2 out of 79 concepts were incorrectly omitted.	0.019	98
Inconsistent Ontologies	1 sec.	3(B) ∪ 4(A) = 7(C). * 2 (C 1). 1 of 2 inconsistencies was solved	5(B) ∪ 6(A) = 9(C). * 5 (C 5). 5 inconsistencies were not solved.	0	100
Tourism of Acapulco	20 sec.	60(B) ∪ 64(A) =131(C). All were correctly copied	61(B) ∪ 65(A) = 124(C). All nodes were correctly copied.	0	100
Multimedia	10 sec.	7(B) ∪ 6(A) = 10(C). All were correctly copied	8(B) ∪ 7(A) = 11(C). All concepts were correctly copied.	0	100

a disadvantage. Unfortunately, until now there were only tools that partially met this need. The emergence of OM provides new support.

OM is an automatic, robust algorithm that fuses the knowledge from two ontologies into a third one, solving some inconsistencies and avoiding redundancies.

The examples shown, as well as others (Cuevas-Rasgado, 2006), illustrate the power of OM: in spite of joining very general or very specific ontologies, it generally does a good job. This is because OM not only compares words, but it also takes into account the semantics or context of each node in the source

ontologies for copying or modifying new properties and concepts into the resulting ontology. It also uses its base knowledge (see the section on "Knowledge Support for OM").

DISCUSSION

Syntactic vs. semantic analysis. OM builds data structures (in OM notation) from data structures, and thus uses limited knowledge (its in-built knowledge, plus the knowledge in *A*, plus the knowledge in *B*) and it exploits "only" syntactic facts. OM does not pretend to find "the truth"[22] among two inconsistent relations, but as more knowledge (more syntactic facts) come into its built-in knowledge, it will do a better job. In fact, when compared with the fusion done by a person imbedded with semantic knowledge, OM already does a reasonable job (Table 1), despite its "limited methodology" and its use of "only syntactic analysis."

OM is automatic. Human intervention takes place outside OM (see the section on "Additional Examples for Real-World Cases"). OM will produce consistent (no redundancies, no contradictions) and complete (no concepts missing in the fusion) ontologies if it were to achieve 100% accuracy (that is, the accuracy of a person that does the merging by hand). This is the goal of OM. How well does OM achieve its goal? About 96% (see Table 1). The section on "Suggestions for Further Work" explain how to improve further its performance.

VERIFICATION OF RESULTS

How can we be sure that OM (or some other merger tool) did a good job? We check (as in Table 1) for wrong, missing, misplaced, or additional (but wrong) relations and nodes in OM's result, against the result manually obtained by a person. At times, the person uses previous knowledge to build nodes or relations in the result that are impossible to be added by OM with the information available to it. The person may add `Dog` eats `meat`, among nodes `Dog` and `meat`, but neither *A* nor *B* say this. We do not count these as mistakes, but we mark them as "areas where more knowledge should produce *this* result." Another verification method could use the editor-reasoner when built, in order to pose questions to the resulting ontology, and check its answers against the answers from a person. These are subjective methods, but probably there could never be (for this purpose) an objective method, where a person's opinion is absent. [23]

REAL WORLD EXAMPLES AND CHALLENGES

Until now, ontology merging was a machine-aided activity, so few real world problems were tackled. OM is the first automated tool for ontology merging, but it has not yet tackled commercially interesting problems. The examples in the chapter come from ontologies obtained from documents found in the Web. But the largest ontology produced by OM (that for a portion of One Hundred Years of Loneliness, Table 1) has only 561 concepts (420 relations + 141 nodes). Larger experiments need to be carried out.

Issues, problems and trends are touched upon in that section.

SUGGESTIONS FOR FURTHER WORK

- Commercial applications appearing in the section on "Commercial Areas Ready to Exploit OM."
- As more knowledge is fused by OM, it could be kept and used as its built-in knowledge (explained in "Knowledge Support for OM"), to improve its accuracy.
- OM could resort to external knowledge sources—some are mentioned in "Knowledge Support for OM." These will help to eradicate some of the arbitrary decisions that current OM makes:

  - Uncertain handling of Case C, footnote 16
  - Preferring its own knowledge, footnote 20
  - Expunging redundant values, contribution h
  - Mistake in Example 6
- Another improvement may come from adding more similarity measures, see "Confusion."

Needed extensions to OM are:

- Handling of time. When young, Juárez was a law student; later he became Governor of Oaxaca, then President of the Supreme Court, then President of Mexico; at that time he fought against Emperor Maximiliam of Habsburg…
- Representing and merging disjunctions. `Ann bought` (a `candy` or an `ice cream`).
- How to represent (and merge) beliefs.
- How to represent conditionals and in general logic restrictions *in a way that OM can analyze*, check (for mutual inconsistency, say), improve, and change them. They should not be *opaque* to OM machinery.

Tools external to OM that will extend OM's applicability:

- A parser that transforms a document into a data structure using OM notation. A difficult task (footnote 7).
- A deductive machinery (a *reasoner*) that answers complex questions posed to the ontology perhaps as graphs, to avoid using a natural language interface.
- Those in footnote 7.

With these, the goal of attaining a large knowledge ontology (see the section on "Increased Yield Through Better Processing the Web Resources"), ready to answer difficult questions, could be achieved—building it incrementally by a tool, not by hand. Let us call this extended tool OM*. And what could be its use? Well, we could add it to our system software, perhaps as a part of the operating system. In the same manner as current word processors check for spelling and grammar, OM* would check documents or data bases for factual or semantic mistakes (assertions not agreeing with OM*'s knowledge), tagging for instance sentences or rows in a data base such as "Abraham Lincoln was born in Japan," "The applicant's age is 27, and he has been working in his previous job for 25 years," or "the shoe has hepatitis." In addition to *common sense knowledge,* OM* will provide *real world knowledge* to the computer. This sounds like exaggerations and wild thoughts, so we shall stop here and concentrate instead in the construction of missing parts of OM*.

ACKNOWLEDGMENT

We acknowledge support from CONACYT Grant 43377.

REFERENCES

AKT project. Retrieved June 26, 2007, from http://plainmoor.open.ac.uk/ocml/domains/aktive-portal-ontology/techs.html

Asunción, P., & Suárez, M. (2004). Evaluation of RDF[S] and DAML+OIL import/export services within ontology platforms. *Lecture Notes in Artificial Intelligence, 2972*, 109-118.

Bechnofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P. et al. (2004, February 10). *OWL Web ontology language* (reference). W3C Recommendation. Retrieved June 26, 2007, from http://www.w3.org/TR/2004/REC-owl-ref-20040210/

Connolly, D., van Harmelen, F., Horrocks, I., McGuinnes, D., Patel-Schneider, P., & Stein, L. (2001, March). *DAML+OIL reference description*. W3C Note, December, 18, 2001. Retrieved June 26, 2007, from from http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218

Cuevas-Rasgado, A.A. (2006). *Merging of ontologies using semantic properties*. Unpublished doctoral dissertation (in Spanish), CIC-IPN, Mexico. Retrieved June 26, 2007, from http://148.204.20.100:8080/bibliodigital/ShowObject.jsp?idobject=34274&idrepositorio=2&type=recipiente

Domingue, J., Motta, E., & Corcho, O. *Knowledge modeling in WeboOnto and OCML, A user guide* (Version 2.4). The Open University and Knowledge Media Institute.

Dou, D., McDermott, D., & Qi, P. (2002). Ontology translation by ontology merging and automated reasoning. In *Proc. EKAW Workshop on Ontologies for Multi-Agent Systems.*

Ganter, B., Stumme, G., & Wille, R. (2005). *Formal concept analysis: Foundations and applications* (1st ed.). New York, NY: Springer

Guzman, A., & Levachkine, S. (2004). Hierarchies measuring qualitative variables. *Lecture Notes in Computer Science (LNCS), 2945*, 262-274.

Guzman, A., & Olivares, J. (2004). Finding the most similar concepts in two different ontologies. *Lecture Notes in Artificial Intelligence (LNAI), 2972*, 129-138.

Jimenez, A. (n.d.). *Quantifying inconsistencies in sentences (facts) with symbolic values*. Ph. D. thesis. CIC-IPN, Mexico.

Kalfoglou, Y., & Schorlemmer, M. (2002). Information-flow-based ontology mapping. In *Proceedings of the 1st International Conference on Ontologies, Databases, and Applicatio of Semantics*.

Knowledge Interchange Format. Draft proposed. American National Standard [dp ANS] NCITS. T2/98-004. Retrieved June 26, 2007, from http://logic.stanford.edu/kif/dpans.html

Kotis, K., Vouros, G., & Stergiou. K. (2006). Towards automatic of domain ontologies: The HCONE-merge approach. *Elsevier's Journal of Web Semantic, 4*(1), 60-79. Retrieved June 26, 2007, from http://authors.elsevier.com/sd/article/S1570826805000259

Large Resources. *Ontologies (SENSUS) and Lexicons.* Retrieved June 26, 2007, from http://www.isi.edu/natural-language/projects/ONTOLOGIES.html

Lenat, D., & Guha. R. (1989). *Building large knowledge-based systems*. Addison-Wesley.

Loom. Retrieved June 26, 2007, from http://www.isi.edu/isd/LOOM/LOOM-HOME.html

Madhavan, J., Bernstein, P., & Rahm, E. (2001). Generic schema matching using Cupid. In *27th International Conference on Very Large Data Bases,* Rome, Italy.

Manola, F., & Miller, E. (2004). *RDF primer.* W3C Recommendation. Retrieved June 26, 2007, from http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

McGuinness, D., Fikes, R., Rice, J., & Wilder, S. (2000). The chimaera ontology environment knowledge. In *Proceedings of the Eighth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues,* Darmstadt, Germany.

Noy, N., & A. Musen, M. (2000). PROMPT: Algoritm and tool for automated ontology merging and alignment. In *Proceedings of the National Conference on Artificial Intelligence,* Stanford Medical Informatics, Stanford University, CA.

Ontologies about Benito Juarez. Retrieved June 26, 2007, from http://es.wikipedia.org/wiki/Benito_Ju%C3%A1rez and http://www.artehistoria.com/historia/personajes/6496.htm

Ontologies about cien años de soledad. Retrieved June 26, 2007, from http://html.rincondelvago.com/cien-anos-de-soledad_gabriel-garcia-marquez_22.html and http://www.monografias.com/trabajos10/ciso/ciso.shtml

Ontologies about Newton. Retrieved June 26, 2007, from http://es.wikipedia.org/wiki/Isaac_Newton and http://thales.cica.es/rd/Recursos/rd97/Biografias/03-1-b-newton.html

Ontologies about Oaxaca. Retrieved June 26, 2007, from http: www.oaxaca-mio.com/atrac_turisticos/infooaxaca.htm and http: www.elbalero.gob.mx/explora/html/oaxaca/geografia.html

Ontologies about poppy. Retrieved June 26, 2007, from http://es.wikipedia.org/wiki/Amapola and http://www.buscajalisco.com/bj/salud/herbolaria.php?id=1

Ontologies about tools and products. Retrieved June 26, 2007, from http://sumesa.com/

Ontologies about turtles. Retrieved June 26, 2007, from www.damisela.com/zoo/rep/tortugas/index.htm and http://www.foyel.com/cartillas/37/tortugas_-_accesorios_para_acuarios_i.html

Serafini, L., Bouquet, B., Magnini, P., & Zanobini, S. (2003). An algorithm for matching contextualized schemas vio SAT. In *Proceedings of CONTEXT 03*.

Stumme, G., & Maedche, A. (2002). Ontology merging for federated ontologies on the semantic Web. In E. Franconi, K. Barker & D. Calvanese (Eds.), *Proc. Intl. Workshop on Foundations of Models for Information Integration*. Viterbo, Italy: Springer.

WEB Onto. Retrieved June 26, 2007, from http://137.108.64.26:3000/webonto?ontology=AKTIVE-PORTAL-ONTOLOGY&name=TECHNOLOGY&type=CLASS

ENDNOTES

1 An easy task for a person who uses *context* and previous knowledge.

2 Other important problems (question-answering, reasoning, the handling of time, how to represent beliefs, and so forth; see "Suggestions for Further Work") are outside the scope of this chapter.

3 OM forges ahead and does not fall into loops.

4 Without contradictions.

5 The result contains *all* available knowledge from the sources, avoiding redundancies.

6 Without user intervention.

7 A Ph.D. Thesis in progress by Paola Neri seeks to make such translation.

8 Some representations are even more restrictive: an ontology has to be a *tree*. In these, a concept could not have two parents: Mexico could not be both a *nation* and an *emerging market*.

9 A relation may be a (full) concept or just a name, a label, a "shallow" relation. Same applies to concepts.

10 Another example: the relation to light (that is, to illuminate) may also be a concept, if one wishes to add more properties to this action. A different concept that can also be represented is light (that is, an electromagnetic radiation). OM does not consider them equivalent, even if somebody gave them the same name. OM has machinery to identify homonyms (contribution e.2).

11 Ontology *C* is incrementally constructed, starting from ontology *A* (step 1). In step 2, OM first adds to *C* nodes in a "superficial" manner (only the label, description, and implicit relations are copied), later in that step 2 these nodes will be completed and "deeply" copied. See footnote 12.

12 *Deep copying.* OM finishes copying a node already superficially copied, adding to it its explicit relations. These new relations link to other concepts, which could already be in *C*. If they are not, they are now superficially copied, and later they, in turn, will be deeply copied.

13 Ontology *C* is searched *depth-first*. A branch of the tree is traveled until the deepest descendant is reached, before OM considers another branch. Since the OM notation uses trees, OM finds easy to do these travels.

14 For instance, in *A* we find `impact printer` method uses `force...` (Figure 9) and in *B* we find `impact printer` procedure uses `print striking to the paper` and C_C = `impact printer` in *A* has a most similar concept *cms* = `impact printer` in *B*, and method and procedure are found by OM to be synonyms. Then all the words in the definition

of procedure in *B* are copied into the definition of method in *C* (Figure 9).

15 Or grandparent, or great-grand parent.

16 This may be a mistake. More information is needed if OM were always to make the right decision. More at "Suggestions for Further Work."

17 In $C = A \cup B$, OM assumes *A*, *B, C* to be self-consistent. But mutual inconsistency can arise when joining *A* and *B*.

18 Progress will be slow until better tools appear (like OM*) and businesses become aware of them.

19 To *understand x* means: to process, exploit, make intelligent inferences about *x*. Pragmatically, a program *understands* some text, information, or situation when such program can productively use it to advance in its goals and objectives.

20 As last resort, if *B*'s new knowledge is inconsistent with *A*'s knowledge, *A* refuses to acquire this new knowledge.

21 If `Gray` and `Green` were the only descendants of its ascendant (`Color`), the partition would be added to `Color`.

22 How can a program that diagnoses EKG anomalies be checked: (1) by comparing its results against experts' diagnosis (subjective method), and (2) by checking reality, for example, an autopsy of the person will show his heart's ailments (objective method). But the purpose of OM is not to find out what is true in the real world; its purpose is to build a consistent *C* from *A*, *B* and its previous knowledge.