Knowledge accumulation through automatic merging of ontologies

Adolfo Guzmán-Arenas *and* Alma-Delia Cuevas
Centro de Investigación en Computación (CIC), Instituto Politécnico Nacional (IPN)
and Instituto Tecnológico de Oaxaca (México)
a.guzman@acm.org; almadeliacuevas@gmail.com

Abstract. In order to compute intelligent answers to complex questions, using the vast amounts of information existing in the Web, computers have (1) to translate such knowledge, typically from text documents, into a data structure suitable for automatic exploitation; (2) to accumulate enough knowledge about a certain topic or area by integrating or fusing these data structures, taking into account new information, additional details, better precision, synonyms, homonyms, redundancies, apparent contradictions and inconsistencies found in the incoming data structures to be added; and (3) to perform deductions from that amassed body of knowledge, most likely through a general query processor.

This article seeks to solve point (2) by using a method (OM, Ontology Merging), with its algorithm and implementation, to fuse two ontologies (coming from Web documents) without human intervention, producing a third ontology, taking into account the inconsistencies, contradictions and redundancies between them, thus delivering an answer close to reality. Results of OM working on ontologies extracted from Web documents are shown.

Key words. Ontology fusion; Ontology alignment; Artificial Intelligence; Knowledge representation; Semantic Web.

1 Introduction

In this post-industrial age, most treasured resources are (a) knowledge and (b) efficient processing of such knowledge; akin to land, cattle and slaves in the middle age. By (a) we mean not data, but *information stored in an way amenable for computer deductions*. Plenty of information, to be sure. By (b) we mean that the computer (and not just people) should be able to make deductions and infer (non trivial) consequences, and provide answers, through suitable deductive mechanisms or algorithms. We find ontologies (§2.1) suitable for this knowledge storage. Internet and searchers such as Google (or on-line encyclopedias such as Wikipedia) are able to provide large quantities of knowledge, thus partially fulfilling (a). Nevertheless, this knowledge (contained in a long list of perhaps relevant documents[1]) must be manually processed (each of them has to be read by a person), missing goal (b). Current situation is similar to the librarian of a large library, who answers a petition from a user by supplying her a large list of books, some of which *may* contain the desired answer (or more often, from the careful reading of several books, the user obtains her answer). The desired situation would be a librarian (now a wise man) who *replies with the right answer* to a question about a topic that his library covers, since he already read and digested every book in his shelves.

Since the computer must acquire the information in point (a) by gradual accumulation of much new knowledge (concepts, relations, typical values…) into the information it previously digested, the machine has to identify by itself redundancies, repeated information, small and big contradictions, synonyms and homonyms, among others cases.

This paper presents OM (for *ontology merging*), an algorithm that (automatically) merges ontologies A and B yielding ontology C, thus addressing goal (2) in the abstract [Neither goal (1) nor (3) are addressed].[2] C is consistent with the knowledge of A and B. Also, the accumulation of knowledge requires that a fair amount of ontologies about the same topic should be merged, which calls for repeated use of OM.

The correct merge must consider not only the syntax of the word and phrases describing the concept (these are called the *definition* of a concept, see Figure 3), but the semantics of the concept thus represented, too [its "meaning" as described by its relations to other concepts in the source ontologies, its similarities to other concepts, superficial

[1] Most of the irrelevant documents result because the search is specified through words, not through concepts. It is a syntactic search.

[2] A tool addressing goal (1) may take the form of a good parser that uses its previous knowledge about the subject (most likely, another ontology) and translates the input text into an ontology representing the knowledge found in it. A tool for (3) looks like a general question-answerer or a deductive program that matches a question (written perhaps as a graph [or an ontology] with free variables, which will be bound to the desired values or answers) with the comprehensive ontology. See, for instance, [2]. Neither tools (1) or (3) are the subject of this paper.

likeness, homophones, etc.].[3] Therefore, the representation to be used (described in §2.2) should be able to characterize these relations. CyC [18] also sought to build a common-sense ontology, but by manual methods. Recent previous work is described in §1.4. Section 2 explains the parts of OM, Section 3 describes how OM achieves the knowledge accretion, and Section 4 shows some results.

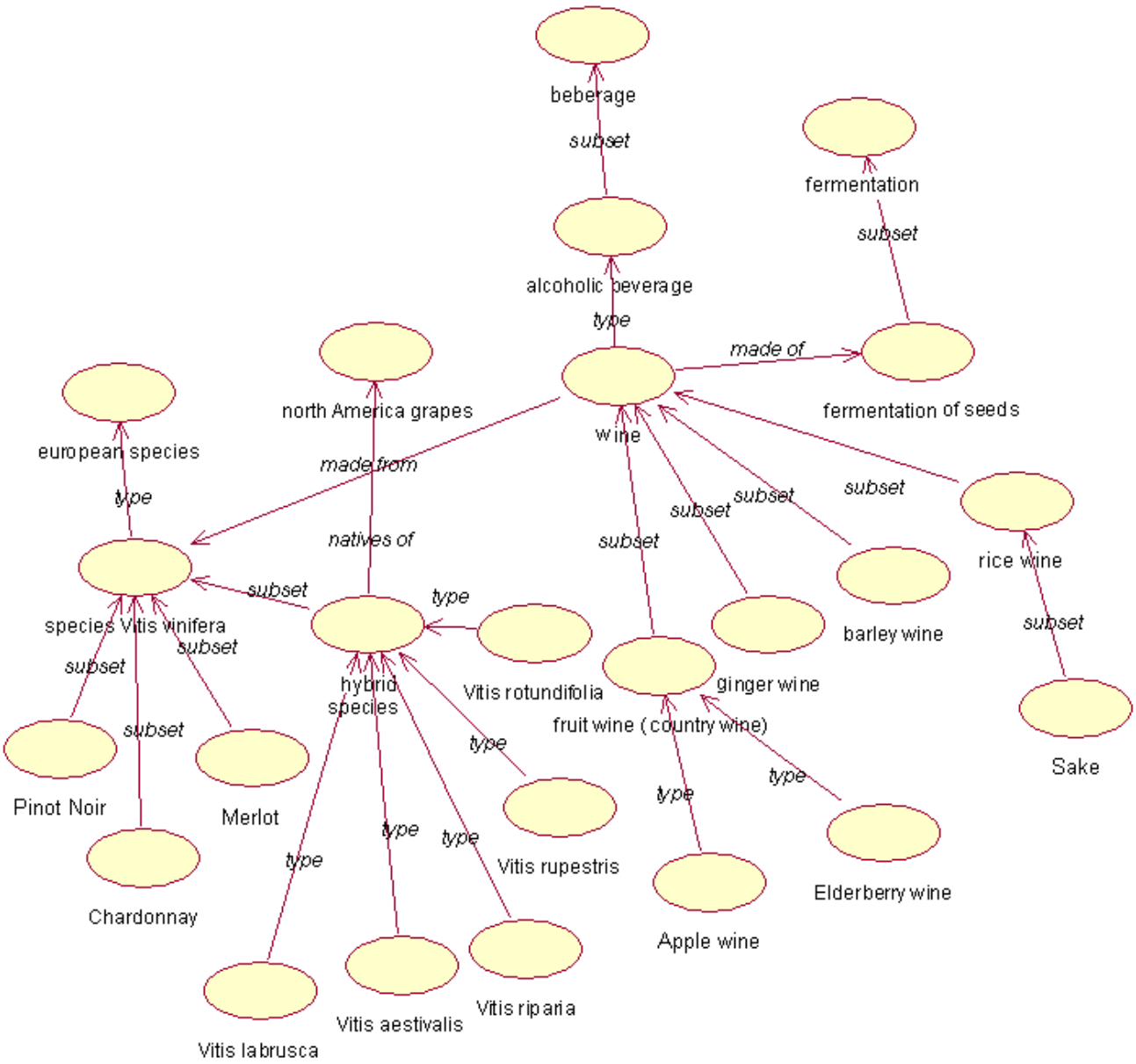


Figure 1. Part of an ontology A (refer to Figure 4) about wine, constructed from text in [25]. Only some nodes are shown

[3] When reading a phrase such as "Clyde is an elephant," human beings immediately assume that Clyde has a trunk, four gray feet, and so on (previous knowledge). "Elephant" has a lot of meaning to us. OM is incapable of doing that. Its only available knowledge is in ontologies A and B. Of course, if B (or A) stores the information that elephants have a trunk, then OM can reflect in C that Clyde has a trunk, too. If A and B both say that the earth is flat, so it will say C. OM is incapable of finding what holds in real life, what is true or right. Also, for *useful* knowledge accumulation (amassing true facts) in A, OM should be fed with trusted ontologies, and in the right order. For instance, in C = OM(OM(OM(OM(A0, A1), A2), A3), A4), the first ontology, A0, should be very basic and well formed, much like the first facts a child begins to learn in his(her) life.

1.1 Knowledge accumulation and extraction by people

A. Accumulation into comprehensive documents. Comprehensive documents that describe in detail knowledge in a given area are now produced by informed writers, who express this knowledge in text documents (or in mathematical notation) in a precise and clear way. What they do is to articulate as clearly as possible their knowledge, accumulated through years of experience, deductions and experiments. Unfortunately, being these documents in a natural language, the extraction of knowledge from them by other persons is done by *reading* and *understanding* them, not by automated means. That is, they are useful for human deduction, not for machine deduction.

B. Knowledge extraction from (A) by the reader. An user extracts from a comprehensive document (as described in (A) information to add to his/her knowledge, by reading and understanding it. In this way, his/her knowledge increases.

Another way for the reader to increase his/her knowledge in a given topic is to find documents that may contain part of the information he/she seeks, looking for them in the library, in an encyclopedia, or in the Web. Internet contains a lot of information in billions of documents, amount them there are Web sites, texts documents, doorway services, music, images, maps, etc. When accessed through searchers (Google, CiteSeer,…), they recover only a small part of the available information because the access is in syntactic form (through labels, words and phrases); that is, through lexicographic and syntactic comparisons. Also, the answer is a large list of documents that are not always appropriated. Thus, each of them has to be read and discarded or digested (understood), a manual process for the reader.

1.2 Knowledge accumulation and extraction by computer

C. Accumulation into comprehensive ontologies. If we believe that ontologies may be a useful way to represent knowledge for human and computer knowledge extraction, then a comprehensive ontology of a given subject may be produced by the tool of point (1) in the abstract, applied to the corresponding comprehensive text document (§1.1.A). Lacking such document, the tool of point (1) in the abstract could be used to build an ontology for each of the many documents found as explained in §1.1.B, and then merging such ontologies using OM or some further improvement to OM. The result would be a comprehensive ontology useful for computer extraction of information and knowledge.

D. Knowledge extraction from (C). Whatever information the user decides to extract from a comprehensive ontology could be mined by posing his/her query in suitable form to such ontology, using the tool of point (3) in the abstract.

1.3 Where does OM fit?

OM may render possible the automatic accumulation of knowledge by computer, if enough specific ontologies were available.[4] A computer that knows a lot about a given topic, and that stores such knowledge in a way amenable to automatic processing (deduction, inferences), will be a formidable knowledge assistant. See also §3.3 "Future Work."

1.4 Current work

A. Building comprehensive ontologies. There have been projects [18] that seek to build a comprehensive ontology by hand, or a list of synonyms (collections of English words and phrases), notably WordNet [7].

B. Ontology representation languages. There are several languages used to represent ontologies. DAML + OIL (Darpa Agent Markup Language + Ontology Inference Layer) [4] is a tag language that provides semantic information to Web resources. As a proposition for ontology presentation, OIL uses formal semantics and logical reasoning to describe the meaning of terms and the implicit information deduction. OIL has a well defined syntax, based on DTD[5] and XML Schema[6]. The Resource Description Framework RDF/XML [15] is strongly related to XML; therefore, useful to use it in conjunction with XML. It is exploited mainly to represent metadata. OWL [1] has all the features of RDF and has been adopted by the W3C consortium as an standard language to represent ontologies in the semantic Web. The Knowledge

[4] Relatively few ontologies exist now, since they have to be manually constructed.

[5] DTD or Document Type Definition is as definition of an SGML document, an older version of XML, that specifies restrictions on its structure.

[6] It is an XML base alternative for DTDs, describing the structure of an XML document. See: w3shools.com/schema/default.asp.

Interchange Format KIF [9] is a language designed for knowledge exchange among different computer systems, created by different programmers at different times, languages, etc. OCML, Operational Concept Modeling Language [6], is a language useful to construct knowledge models, allowing the specification and operation of functions, relations, classes, instances and rules. The Open Knowledge Base Connectivity OKBC [20] is a protocol to access knowledge bases stored in knowledge representation systems, for instance an object oriented data base. Our OM notation (§2.2) was designed not to produce still another definition language, but as a fast way to provide ontologies the needed features that will enable OM's work. Thus, if a given knowledge composition present in the source document was needed to be characterized in the ontology, the OM notation could be modified to allow its easy depiction. The advantage of this approach was a moldable representation language where features could be quickly added. The disadvantage is that we will need perhaps, in the future, to write a translator from OM notation to an standard ontology language.

C. Accumulation of knowledge through ontologies. There are computer aided methods to identify and join the knowledge in two ontologies. These methods help the user to complete the work. PROMPT [8], Chimaera [16], OntoMerge [6], IF-Map and ISI [21] require that user solve problems presented during the fusion. Methods like FCA-Merge [19] use Formal Concept Analysis for ontology representation, forcing these to be necessarily consistent. But the majority of ontologies in Web present inconsistencies, either among different parts of the same ontology, or when one is compared with another. A recent advance in ontology union is HCONE-merge [13], which extracts from the semantic data base WordNet [7] intermediary information for the fusion, requiring less user support.

2 The parts of OM

OM fuses ontologies represented in a notation, called simply "OM notation" (§2.2 and Figure 3). The examples in this paper show ontologies that were manually created from real documents extracted from the Web, and written in such notation. Then OM fused them (automatically, that is). The same ontologies were also manually fused. The manual result was then manually compared against the automatic result. Table 1 shows some of these comparisons.

OM has some built-in (mostly syntactic) initial knowledge about words and word phrases (§2.3). OM rests heavily in two functions previously defined elsewhere: COM (§2.4), which compares two concepts in different ontologies, and conf (§2.5), which finds the confusion (a kind of dissimilarity) among two concepts belonging to the same hierarchy.

2.1 Ontology definition

In Philosophy, ontology (which is called "being theory") is a general analysis of the being, what is it, how is it and how is it possible.

In Computer Science, ontology is a data structure, a data representation tool to share and reuse knowledge between Artificial Intelligence systems which share a common vocabulary. Thus, an ontology is a set of definitions, classes, relations, functions and other objects of speech [5].

Mathematically, it does not exist a satisfactory theory which characterizes formally the ontology definition. However, some intents have appeared, such as [12]. Formally [5], ontology is a pair

$$O = (Ç, R)$$

Where:

Ç is a set of nodes (representing concepts), some of which are relations.

R is a set of restrictions, of the form $(r; c_1; c_2;\ldots; c_k)$ between the relation r and concepts $c_1,\ldots, c_k$. Lower c is used to refer to each concepts of set Ç, while a semicolon separates members in the restrictions. For example: (cut; `scissors`; `sheet`), (print; `printer`; `document`; `ink`). In these examples, the concepts that are relations too, are: cut and print. The first element of a restriction is a relation. The restrictions are not limited to have two members besides the relation. Therefore, an ontology is a hypergraph with Ç the set of nodes and R the set of hyper-relations (which we call restrictions). For clarity, in this paper ontologies are represented as graphs, where nodes are concepts and edges are restrictions.

Figure 1 shows part of an ontology about wine. Figure 2 is another ontology about wine. Notice how they describe wine from different view points.

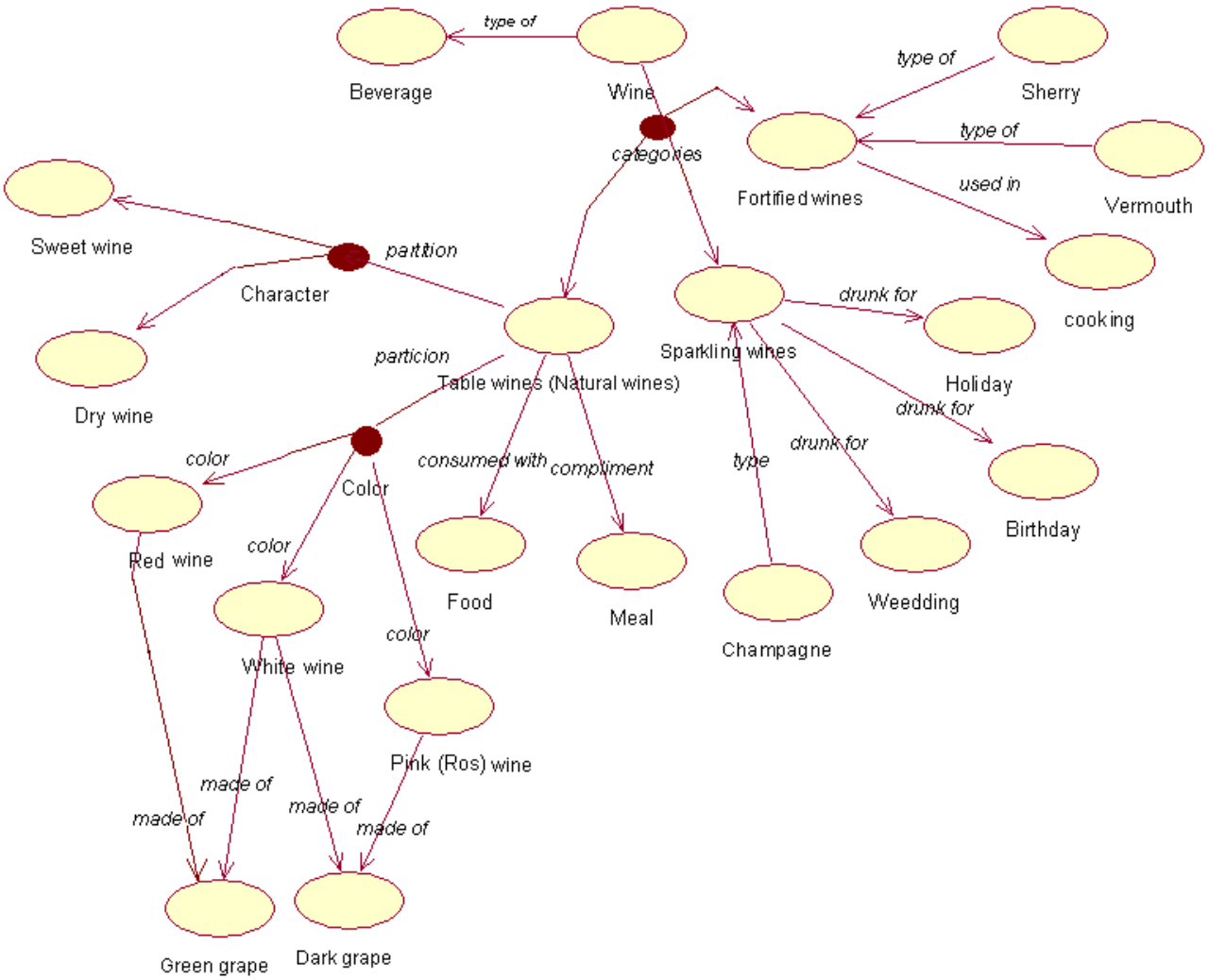


Figure 2. Ontology B about wine (refer to Figure 4 and Table 1), partially shown. A solid dot represents a partition. Thus, `Table Wines` are partitioned by color into `Red wine`, `White wine` and `Pink wine`. The relation character partitions `Table wines` into `Sweet wine` and `Dry wine`. Constructed from [24]

2.2 OM notation

Until now, ontologies have been proposed to solve Semantic Web problems, facilitating computers to better understand Web sites, for instance. The ontologies should express the required semantics, and the language in which these ontologies are written should be powerful enough for this purpose.

The purpose of OM notation is to facilitate automatic merging. It represents ontologies through a structural design with XML-like labels, identifying the concepts and their restrictions. Examples appear in Figure 3 and Figure 4.

```
<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>
```

Figure 3. Representation of an ontology in OM Notation. 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 (a word or a world phrase such as concrete_object, physical_object) goes between <word> and </word>; the restrictions of the concept (such as eats of concept `human being`) go between <relation> and </relation>. The description of a concept ends in </concept>. Nested concepts (such as `thing` and `physical_object`) indicate that `physical_object` is subordinate (or hyponym) of `thing`, the precise meaning of this subordination is indicated by <subset> `thing` </subset>. The restrictions expressed by nesting are called implicit restrictions (currently, they are member of, part of, subset, part*). The other restrictions, such as eats, are called explicit. Explicit restrictions of a concept are known in other works as relations, properties or attributes of the concept. Notice how `human being` is partitioned by age. A complete description of the OM notation appears in [5]

In OM Notation, a restriction can be n-ary; a restriction relates concepts (the first concept in a restriction is a relation). For example, (Color; `Zebra`; `White`; `Black`) is a restriction on concepts Color, `Zebra`, `White` and `Black`. The restrictions can be loosely considered as properties or characteristics of the node or concept where they are defined. Example: the restriction eats in Figure 3. Other restrictions exist, such as hyponym (subordinate) restrictions, that are expressed through nested concepts. Example: `plant` is a subset of `physical object`.

The purpose of OM notation was not to introduce yet another language; but to provide the ways and means to represent the nuances of knowledge that OM required to perform an effective automatic accumulation.

2.2.1 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 languages (§1.4.B) may not. For instance, not only `male person` and `female person` are subsets of `person`, they are a partition of

person. The gender of a person will tell us to which of the partitions male person and female person that person belongs. See in Figure 2 how categories partitions wine into fortified wines, sparkling wines and table wines. Another example: if John is a person and person has a partition called age with subsets child, young person and adult, from the age of John (a number) the system can classify John as a child, a young person or an adult.

(b) A concept can be a relation, too. Often, ontologies are represented as a graph O = (C, R) consisting of two disjoint sets: C (nodes, or concepts) and R (edges, or relations).[7] Two disadvantages of this visually oriented approach are: • all the relations are binary; • a concept can not be a relation. In OM notation relations are also concepts (Cf. §2.1). With this improvement it is possible[8] to know more about the relation that restricts the concepts. For example, one can say (mother of; Mary Ball Washington; George Washington) to indicate that Mary is George Washington's mother, but mother of can be a concept that contains more information elsewhere in the ontology, for instance, related to child of by the relation inverse,[9] and it is a subset of relation relative.

2.3 Initial knowledge used by OM

OM has built-in some (mostly syntactic) initial knowledge about words and word phrases. These are:

A.- Articles and linking words like *in, the, to, this, and, or*, etc. are ignored in the definition of the concept.

B.- Words that change or reject concepts in the relation's definition, such as: *except*, *without*. For example: Poppy without Petiole. Which means that the concept Petiole does not form part of the concept Poppy.

C.- Hierarchies of concepts.[10] The hierarchy represents a taxonomy of related terms. It is used to compute the *confusion* (§2.5). We exploit it to detect synonyms and "false inconsistencies" due to degree of detail.

Additional sources of initial knowledge (not yet incorporated into current OM) could be:

D.- OM could use as part of its initial knowledge the previous knowledge already amassed. See formula in footnote 3. OM does not do this, yet. Other potential sources of initial knowledge are mentioned in §4.2.

2.4 COM finds the closest concept to C_A in another ontology B

Let A, B be ontologies. The algorithm COM takes a concept $C_A \in A$ and finds its most similar concept $C_B \in B$. It has been described elsewhere [10]; we explain it here briefly for completeness. $P_A \in C_A$ stands for the parent of A,[11] and $P_B \in B$ stands for the parent[11] of C_B. To find the similarity between nodes of an ontology, OM uses COM, which employs the definitions (words) and restrictions on those concepts according to four cases:

Case A: C_A finds its match $C_B \in B$ and its parent P_A also finds its corresponding $P_B \in B$ (§2.4.1).

Case B: P_A finds its match $P_B \in B$ but C_A can not find its corresponding $C_B \in B$ (§2.4.2).

Case C: C_A finds its corresponding C_B but its parent P_A can not find a suitable $P_B \in B$ (§2.4.3).

Case D: C_A can not find its corresponding C_B, and neither P_A can find its corresponding $P_B \in B$ (§2.4.4).

2.4.1. Case A: C_A finds its match C_B and P_A also finds its match P_B

Concepts C_B and P_B are looked in B, seeing if the definition of some concept in B matches the majority of the words and word phrases of the definition of C_A, and the definition of a parent of this C_B matches the majority of the words and word phrases of the definition of P_A, a parent of C_A. If needed, grandparents of C_A and C_B are considered, too. Finding a match, COM returns two values: (1) The C_B found in B; that is, the most similar concept to C_A; (2) The similarity value *vs*, a number between 0 (no similarity) and 1 (completely similar).

---

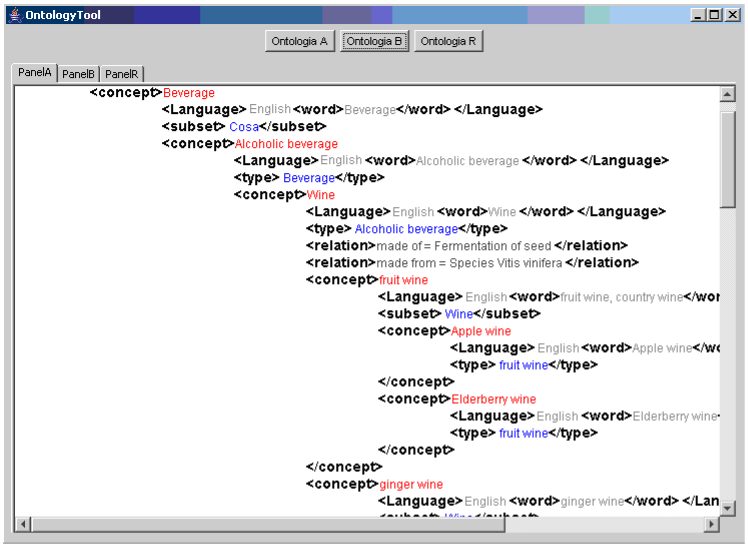
[7] Earlier ontology representations were even more restrictive: an ontology had to be a *tree,* a graph without cycles. In these, a concept could not have two ancestors (two parents): Mexico could not be both a nation and an emerging market.

[8] But not required. Thus, a relation represented by just its name is a "shallow" relation of which little is known (except its name).

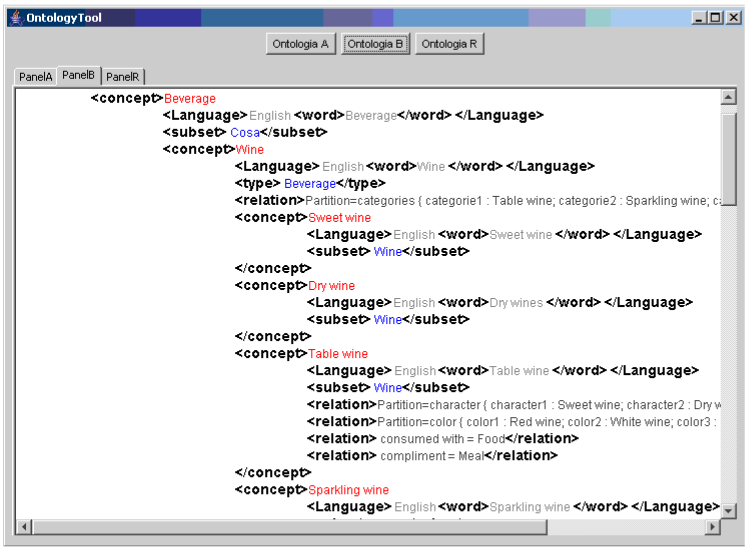
[9] Another example: the relation to light (that is, to illuminate) may also be a concept, if one wishes to add more properties to this action (for instance, that it spreads in 3 dimensions). A different concept that can also be represented is light (an electromagnetic radiation). OM does not consider them equivalent, even if somebody gave them the same name. OM has machinery to identify homonyms (§3.6).

[10] A *hierarchy* H of an element set E (a set whose elements are non-numeric constants) is a tree with root E and where each node it is either an element of E or it is a set whose sons form a *partition* of such node.

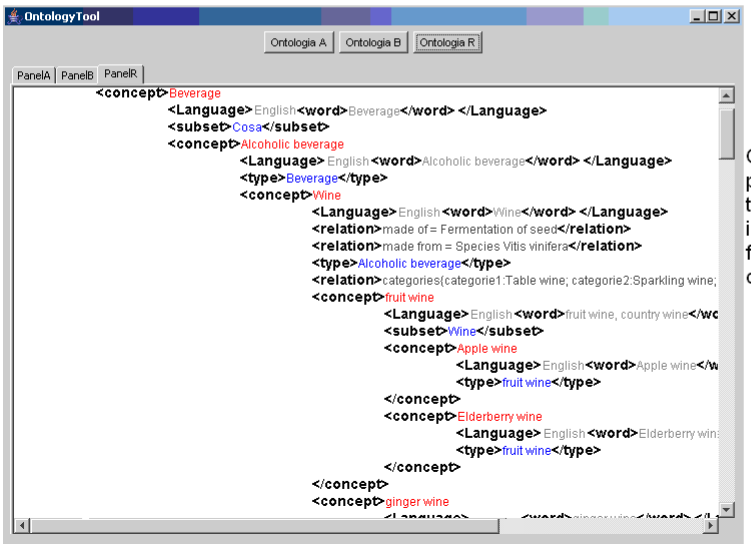
[11] Or for *a* parent of it, since a concept can have several parents. Each possible parent is considered in turn.



Ontology A about wines. It is one of the inputs to OM. It can be displayed by pressing the button "Ontologia A".



Ontology B about wines. The other input to OM.



Ontology C, the result produced by OM. Pressing the button "Ontologia R" invokes OM to compute the fusion of A and B, and to display the result C.

Figure 4. Ontologies A, B, C about wines, partially shown. The result of the fusion is ontology C. How OM produced the result is described in §3. Accuracy of C is found in table 1.

Example: Suppose hammer ∈ A has the definition (hammer; mallet) and its parent tool ∈ A has definition (tool; utensils; appliance). mallet ∈ B has the definition (mallet; hammer), while its parent implement has definition (implement; utensils; device; tool). In this case hammer ∈ A matches mallet ∈ B, since its parent tool matches implement. COM returns mallet as the most similar concept to hammer ∈ A, and *vs*=1. Example: suppose we want to find the most similar concept to lemon ∈ A. Let us have (parent of; lemon; fruit)∈A, (parent of; lemon; citric)∈B, (parent of; orange; citric)∈B, (parent of; tangerine; citric)∈B, (parent of; citric; fruit)∈B, (parent of; apple; fruit)∈B, (parent of; pineapple; fruit)∈B. When looking for the most similar concept to lemon ∈ A, its parent fruit ∈ A does not match with citric ∈ B, but with the grandparent of lemon ∈ B, namely fruit ∈ B. Thus, COM returns lemon ∈ B.

2.4.2. Case B: The parent P_B similar to P_A is found, but none of P_B'sons matches C_A

If no C_B is found, then COM is called recursively with P_A as parameter to see who is P_A's parent. If the parent of P_A is the root of A (O_{ARoot}), COM finishes without success. Otherwise, each son of P_B is sequentially examined. If some son C_B' of P_B has the majority of its restrictions and values similar (through recursive use of COM) to those of C_A, such C_B' is the one looked for, and COM returns C_B'∈B as the concept most similar to C_A. If no son C_B' of P_B has such property, then we search for such C_B' among the sons of the parent of P_B, that is, among the brothers of P_B. If needed, such C_B' is sought among the sons of the sons of P_B, that is, among the grandsons of P_B. The successful C_B' is the desired answer. If no C_B' can be found after all this search, then COM returns "a new son of P_B," meaning that the most similar node in B is a son of P_B, which does not yet exist in B. In this case OM will just add C_A as an additional son to P_B in the resulting ontology C. *This is how synonyms are handled* by OM. For instance, we want to find in B the most similar concept to maize ∈ A. Suppose (parent of; maize; cereal)∈A, (parent of; corn; cereal)∈B, (parent of; wheat; cereal)∈B, (parent of; sorghum; cereal)∈B. We see that cereal ∈ A matches cereal ∈ B, but no maize ∈ B is found. Then COM looks successively at corn, wheat, sorghum, observing if most of the properties of one of these sons (size of seed, shape, color…) of cereal match the corresponding properties of maize. The answer is corn, which is returned by COM. If (parent of; corn; cereal)∈B were absent from B, then COM will return "a new son of cereal ∈ B" as the most similar value to maize ∈ A.

2.4.3. Case C: C_A finds its corresponding C_B but its parent P_A can not find a suitable P_B∈B

If C_B is found but no P_B is found, then the grandfather of C_B∈B and great-grandfather of C_B∈B are tested to see if they are similar to P_A. If some of them is, that is enough and C_B is returned as the value of COM (this is Case A). If this does not happen, COM verifies if the majority of the restrictions (relations and its values) of C_A are compatible with those of C_B; also, do most of the sons of C_A coincide (through recursive use of COM) with respective sons of C_B? Such being the case, C_B is the most similar concept to C_A, and COM finishes. If just part of the properties of C_B are similar to those of C_A, the answer is "probably C_B" and the algorithm finishes. If no one or few properties nor sons are similar, then the answer "doesn't exist" is returned. This case considers concepts that have the same definition but different meaning, for instance, if (parent of; bill; invoice)∈A, while (parent of; bill; currency)∈B, then COM returns as the most similar concept to bill∈A the value "it does not exist" in B. But it will also find that Mexico ∈ A is most similar to Mexico ∈ B, where (parent of; Mexico; nation)∈A and (parent of; Mexico; emerging market)∈B, since most properties of both Mexico's coincide. In C, Mexico (represented by a single concept) will be both son of nation and of emerging market. Another example: if A contains (parent of; tangerine; citric), and B contains restrictions (parent of; orange; citric), (parent of; lemon; citric), (parent of; grapefruit; citric), then COM may return "probably orange ∈ B" as the most similar concept to tangerine ∈ A, depending on how many restrictions in both fruits were similar.

2.4.4. Case D: C_A can not find its corresponding C_B, and neither P_A can

If neither C_B nor P_B can be found in B, then the answer is "doesn't exist" and COM ends. This situation is common when we are talking about two different ontologies, for example: A is an ontology about Information Systems while B is an ontology about Natural Resources. This is a very easy case for OM: if A knows nothing about dinosaurs, and B knows a lot, then all of B's dinosaur knowledge is added to C, since it finds no contradictions or equivalences in A. But beware that A *may* have some knowledge that is also shared with B's dinosaur knowledge, for instance the concepts size, leg, skin, feather, wing, fang, herbivore, etc. These shared concepts are fused, not just added to C.

2.5. conf finds the confusion among two concepts placed in a hierarchy

The function conf has been explained elsewhere [14]; we give here a short introduction, for completeness. conf(*r, s*) measures the confusion (a kind of error) when object *r* is used instead of the intended or real object *s.*

What is the capital of Germany? *Berlin* is the correct answer; *Frankfurt* is a close miss, *Madrid* a fair error, and *sausage* a gross error. What is closer to a *cat*, a *dog* or an *orange*? Can we measure these errors and similarities? Can we retrieve objects in a database that are close to a desired item? Yes, by arranging these symbolic (that is, non numeric) values in a hierarchy. [10]

The confusion appears when a value of a restriction in C_A is incompatible, contradicts or is inconsistent with respect to the corresponding value of the equivalent restriction in C_B, Example: (form; `Earth`; `round`)∈A and (form; `Earth`; `flat`)∈B. conf gives a value between 0 (total contradiction) and 1 (total compatibility), not just 1 or 0.

Let *r* and *s* be two values (nodes) in a hierarchy with height[12] *h*.

The function CONF(*r, s*) called *absolute confusion* that result by using *r* instead of *s* (the intended value) is:

CONF(*r*, *r*) = CONF(*r, s*) = 0 when *s* is one of the *r* antecessor;

CONF(*r, s*) = 1 + CONF (*r*, *father_of*(*s*)) otherwise.

CONF(*r, s*) is the number of *descending* links as the hierarchy is traveled from *r* to *s*, the intended value.

CONF is not a symmetric function. Its value can be altered by simply adding additional nodes between or levels in the hierarchy. To avoid this, we define the relative confusion, or simply confusion.

Definition. The function conf(*r, s*) that results by using *r* instead of *s* is:

$$conf\,(r,s) = \frac{CONF(r,s)}{h}$$

The function conf(*r, s*) is the absolute confusion CONF(*r, s*) divided by *h*, the height of the hierarchy.

Example: If a hierarchy has (part of; `San Pablo Guelatao`; `Ixtlan Mountains`), (part of, `Ixtlan Mountains`; `Oaxaca`), (part of; `Oaxaca`; `Mexico`), then CONF(`San Pablo Guelatao`, `Mexico`) = 0; CONF (`Mexico`, `San Pablo Guelatao`) = 2.

3. Automatic merging of ontologies by OM

OM merges two ontologies A and B into a result C. At least the roots of A and B coincide. OM was developed at CIC-IPN in a doctoral dissertation [5]. It is a totally automatic algorithm because it does not need user intervention and it is strong because it can join ontologies with inconsistencies (see Table 1). The general form for obtaining C is:

$$C = A \cup \{ c_B', r_B \mid c_B', r_B = ext(r_A, r_B) \ \forall \ c_A \in A \}$$

The resulting ontology C is the original A increased by some concepts and restrictions of B that the function *ext* gets.

Where:

C_A is a concept of ontology A, r_A are the restrictions of c_A that exist in A, r_B are restrictions of c_B that exist in B and c_B is the most similar concept in B to c_A (as found by COM, §2.4);

∪ means ontology joining. It is a carefully joining, different to set union.

ext(r_A, r_B) is the algorithm that completes the restrictions r_B which are not in A with those c_B' (which are in B) that do not contradict knowledge from A. That is, for each node $c_A \in$ A, all its restrictions in A are retained in C, and only *some* restrictions in B of c_B (the concept most similar in B to c_A) are added to C, as well as their "target" concept c_B'. For instance, let c_A = `Gottfried Wilhelm Leibniz` with most similar concept c_B = `Leibniz` ∈ B. If the restriction (r_B c_B c_B') = (was born in; `Leibniz`; `Leipzig`) is in B, and if it does not contradict knowledge from A, then restriction (r_A c_A c_B') = (was born in; `Gottfried Wilhelm Leibniz`; `Leipzig`) is added to c_A in C. More examples below.

This extraction must be carefully, so as not to introduce inconsistencies, contradictions or redundant information in C. *ext* is large, and it is explained in §3.2 to 3.7. When applying *ext* to each concept c_A∈A, OM adds to C the additional

[12] The height of a tree is the number of ridge that exist in the way of its root to the most distant leaf.

knowledge (about c_B) in B not present in A. Notice that *as new nodes from B are added to C,* these are also searched for additional restrictions in B. So that the quantifier $\forall\, c_A \in A$ in the above formula should really say $\forall\, C \in C_{OLD}$.

Past ontologies generated by OM could be available to OM as built-in knowledge to be used in subsequent knowledge accumulations (Cf. §2.3.D). This will greatly enhance its merging ability.

3.1 General description of OM to fuse ontologies

An overview of OM is:

1. Copy ontology A into C.
2. Starting from root concept c_{Root} in C, and through each concept in C (including those recently added as partial result of the fusion):
   - I. Look in B for its most similar concept c_B (using COM of §2.4).
   - II. If such c_B exists in B, then
     - a) its new restrictions are added to C;
     - b) redundant restrictions are eliminated(§3.2);
     - c) some collections of subsets are promoted to partitions (§3.3);
     - d) new partitions are added to the result C (§3.5);
     - e) identical partitions are detected (§3.3);
     - f) synonymous are verified (§3.4);
     - g) some inconsistencies are detected and solved (using Confusion theory, §2.5);
     - h) homonyms are detected and handled as two different concepts (§3.6);
     - i) some contradictions are detected and solved (§3.7), so that inconsistency does not creep into the result.
   - III. If no such c_B is found in B, then take the following concept c_C (traveling C in depth-first priority) and go to I.

   If in step II the contradiction between some node in A and another node in B could not be solved, then the value of A prevails, and it is conserved in the result C. Thus, if A says that the Earth is flat, and B that it is round, and no further knowledge is available to detect the contradiction, A prevails. More in §3.7.
3. Stop when all the concepts of C are analyzed.

3.2 Elimination of redundant restrictions

OM purges redundant restrictions from the resulting fusion, as these are created. Example: In Figure 5, ontology A has the knowledge (part; `Apparatus`; `Organ`), while B knows (subset; `Apparatus(System)`; `Thing`). Thus, the result C should have both of these restrictions. Nevertheless, OM compares the predecessors of `Apparatus` in A against the predecessors of `Apparatus (System)` in B, and discovers that restriction (subset; `Apparatus`; `Thing`) can be derived now in C through the recently introduced restriction (part; `Apparatus`; `Organ`). Therefore, (subset; `Apparatus (System)`; `Thing`) is deleted from C.

3.3 Handling subsets and partitions

If an ontology has a partition of a given node into subsets, whereas the other ontology has some of these subsets present, then the partition prevails in the result C. Subsets outside those forming the partition still go to C. Example: height ∈ A partitions `person` ∈ A into `short person`, `medium height person`, `tall person`. In B, `person` has subsets `physician`, `tall person`, `engineer`, `medium height person`, `plumber`, `short person`. Then `person` ∈ C will have the partition `short person`, `medium height person`, `tall person`, and the subsets `physician`, `engineer`, `plumber`.

If one ontology partitions a node in a way (example: age partitions `person` ∈ A into `young`, `mature`, `old`), and the other ontology partitions the same node in a different way (gender partitions `person` ∈ B into `male`, `female`), then both partitions appear in C.

A partition appearing in just one of the ontologies goes to the result (§3.5).

Two partitions on the same relation (say, age partitions `person ∈ A` into `young`, `mature`, `old`, while age partitions `person ∈ B` into `juvenile`, `adult`, `senile`) may be merged if the corresponding subsets are indeed equivalent; otherwise, both partitions go to C. More about partition handling in [5].

Example: in Figure 6 the concept `Human body system` in A finds its most similar concept in B to be `Human body system`, too, which has the partition System. OM observes that the parts of this partition correspond to known parts of `Human body system` in A. Thus, System is kept as partition in C.

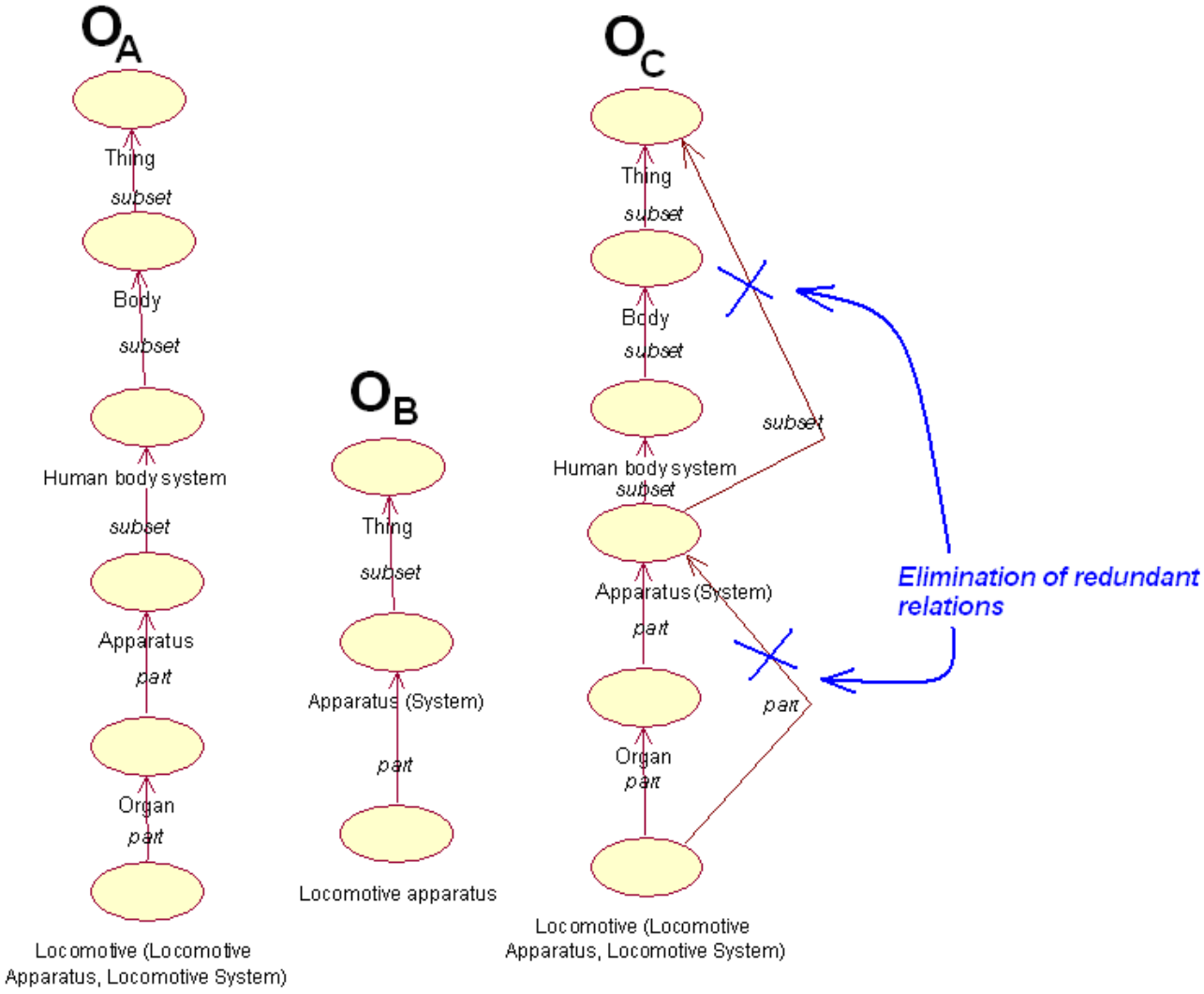


Figure 5. In ontology C the concept `Apparatus (System)` has the restrictions (part; `Apparatus (System)`; `Organ`) and (subset; `Apparatus (System)`; `Thing`). This last restriction is redundant and OM eliminates it. Also, in C the concept `Locomotive` has the definition (Locomotive Apparatus, Locomotive System) and the restrictions (part; `Locomotive`; `Organ`) and (part; `Locomotive`; `Apparatus (System)`). This last restriction is expunged from C by OM, which deems it redundant

.

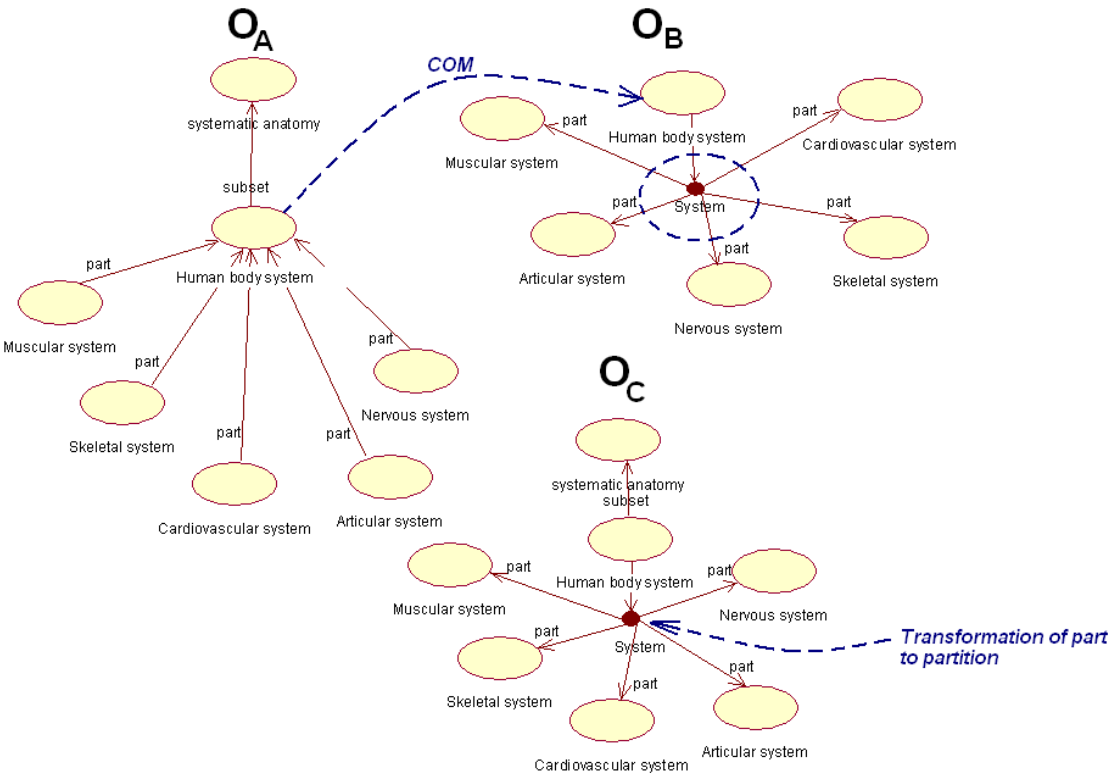


Figure 6. The partition System of the concept `Human body system` in B is added to resulting ontology C, replacing the less precise knowledge that `Muscular System`, `Skeletal System`, `Cardiovascular System`, `Articular System` and `Nervous System` as just part of `Human body system`. That is, for A some other system besides those five could still be part of `Human body system`, while for B (or C) this is not possible

3.4 Identification of synonyms and their treatment

Figure 7 shows an example with concepts `Locomotive system` in A and `Locomotive Apparatus` in B. Through COM, OM finds that for concept `Locomotive system` in A its most similar concept in B is `Locomotive Apparatus`, by syntactic comparison of their respective definitions "Locomotive System/Locomotive Apparatus" and "Locomotive Apparatus." Therefore, both concepts are merged into a single concept in C, which inherits all the restrictions of `Locomotive system` ∈ A and `Locomotive Apparatus` ∈ B.

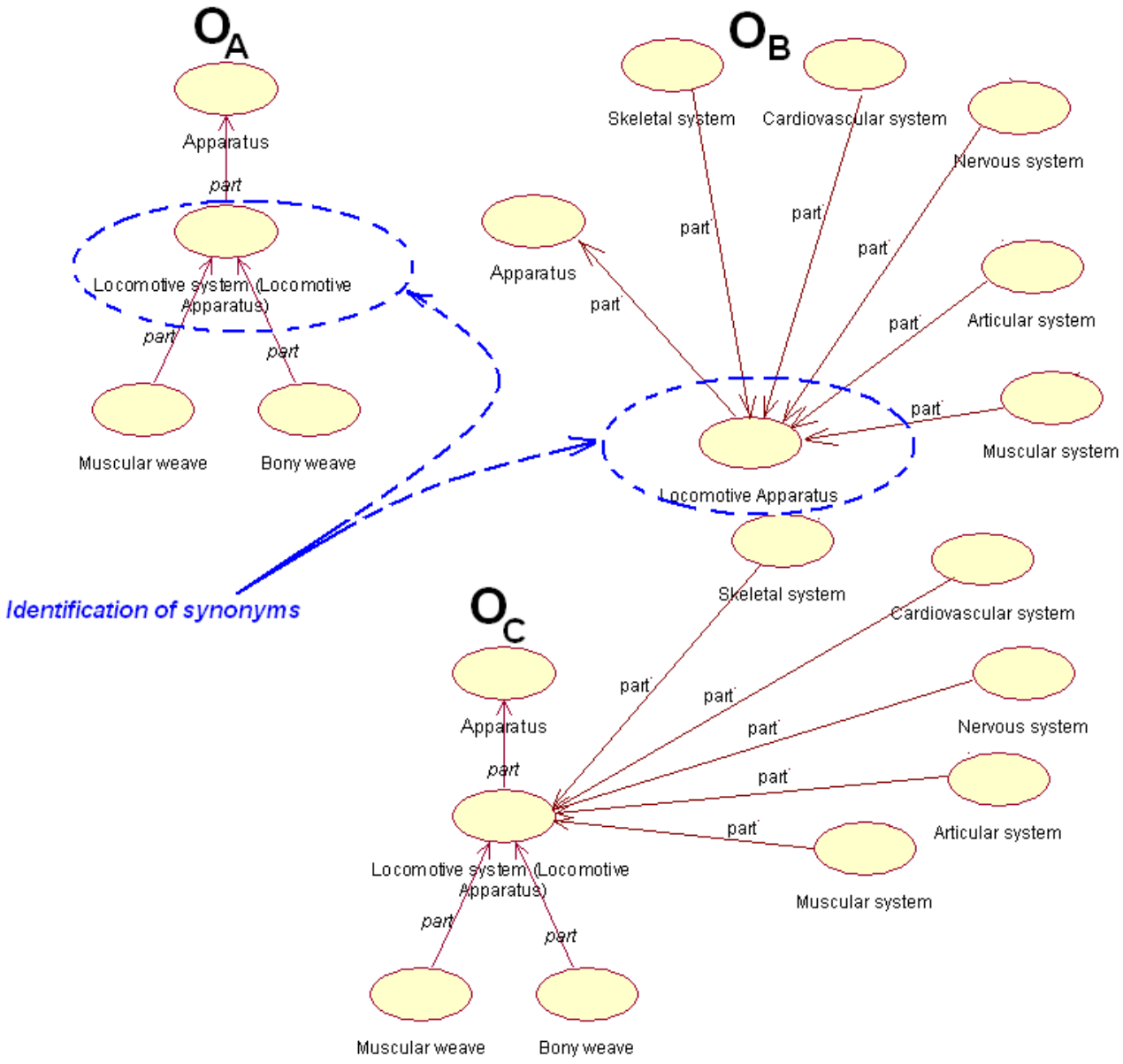


Figure 7. Ontologies A and B where `Locomotive system` and `Locomotive Apparatus` concepts are detected as synonyms

This same process of identification of synonymy is applied when the concepts are relations.

3.5 Copy of a new partition

The example in

Figure 8 shows ontology A with the concept `Esplacnology` having a partition that `Esplacnology` in B lacks. Therefore, the partition is added to the resulting ontology.

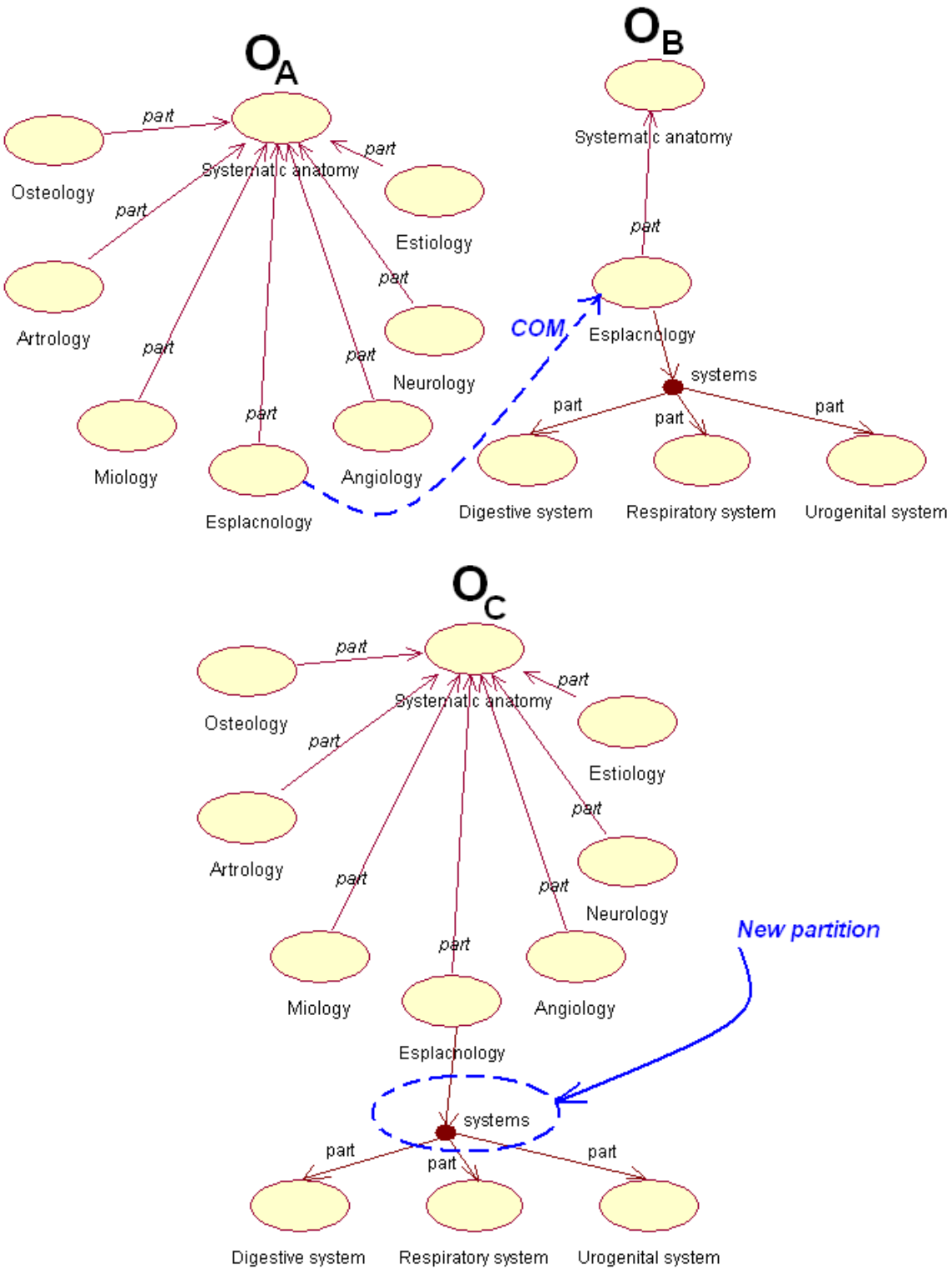


Figure 8. The partition systems is copied into the resulting ontology C

3.6 Homonyms –detection and proper handling

`Fluke` in A and `fluke` in B are detected as different concepts, since they have different parents, different descendants, and different properties. Both are copied to C (Figure 9), as two different concepts with the same name.

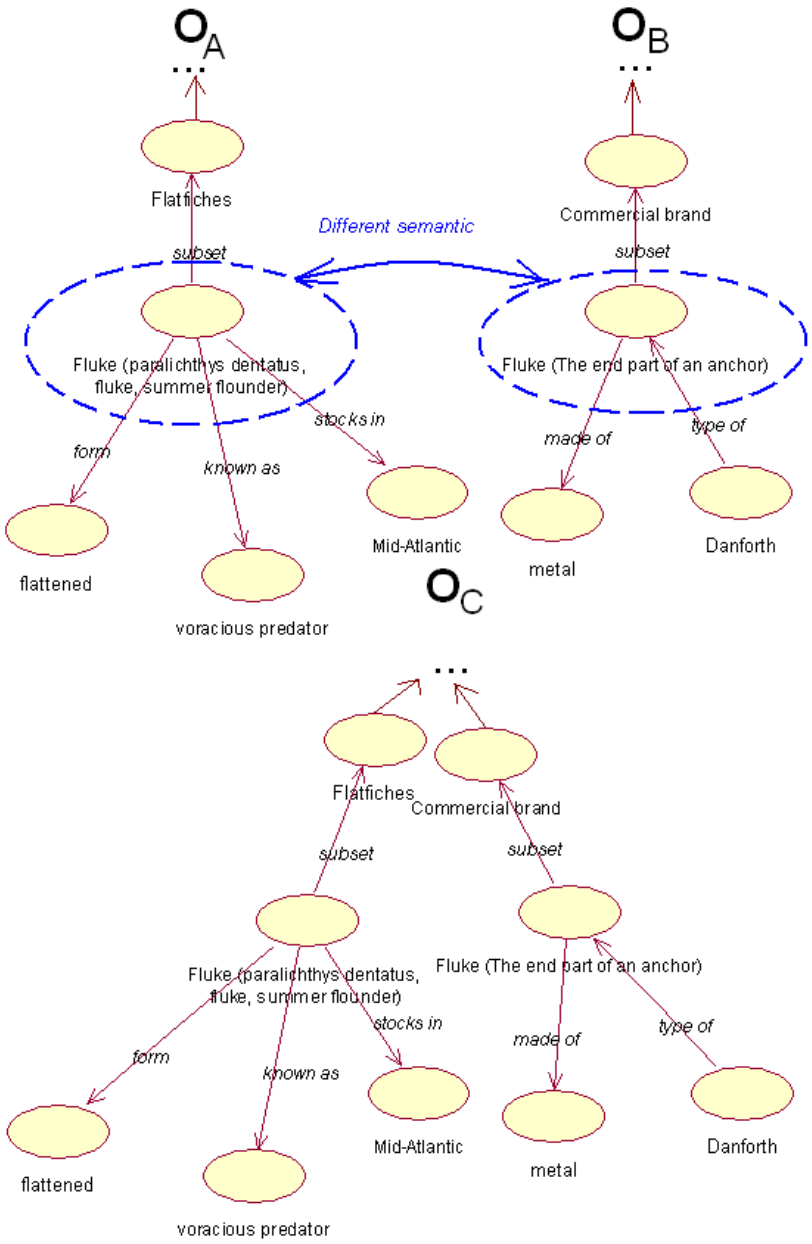


Figure 9. `Fluke` in ontologies A and B are homonyms, thus C has two different concepts with the same name

3.7 Detecting contradictions and resolving some of them

In addition to the detection of some contradictions shown already in §§3.3, 3.4, 3.6, OM has another powerful way to detect and solve some types of contradictions. Many relations can have several values (say, my_friend_is), but some relations (say, my_father_is) can only have one value. Thus, `Lincoln` can not have two fathers. A person can not be born

in two different places. He or she can not be buried in two different sites. How does OM handle the case where A says (was born in; Lincoln; Kentucky) while B says (was born in; Lincoln; USA)? First, it checks that the relation was_born_in is indeed single-valued. If it is, it checks that the relation was_born_in ∈ A is the same as the relation was_born_in ∈ B (it could be just homonyms, with different meaning, as in §3.6). Then, it looks at USA and Kentucky. Are they perhaps synonyms, just as maize and corn (§3.4)? Failing all of this, OM resorts to conf (§2.5) to find the confusion between USA and Kentucky, and the confusion between Kentucky and USA, and it finds that one of them, conf(Kentucky, USA), is 0. Kentucky can be used instead of USA in this case; so, OM keeps the most specific value [the value *r* that forced conf(*r*, *s*) to be 0], Kentucky in this case, in the result C.

When trying to merge (place of birth; Gottfried Wilhelm Leibniz, Germany)∈A with (place of birth; Gottfried Wilhelm Leibniz, Leipzig)∈B (Figure 10), the ontology in Figure 11 is of no use for conf (it does not mention Leipzig, a city of Saxony). Then, OM does not add (place of birth; Gottfried Wilhelm Leibniz, Leipzig) to C, but it keeps the current knowledge form A, that is (place of birth; Gottfried Wilhelm Leibniz, Germany). Thus, if OM can not solve the contradiction between knowledge in A and knowledge in B, *A prevails.* OM refuses to add to C the unresolved contradicting knowledge coming from B. The unfortunate outcome is that once C gets something wrong, it will not change it regardless of enough posterior evidence to the contrary. §4.2.E suggests how to defeat this.

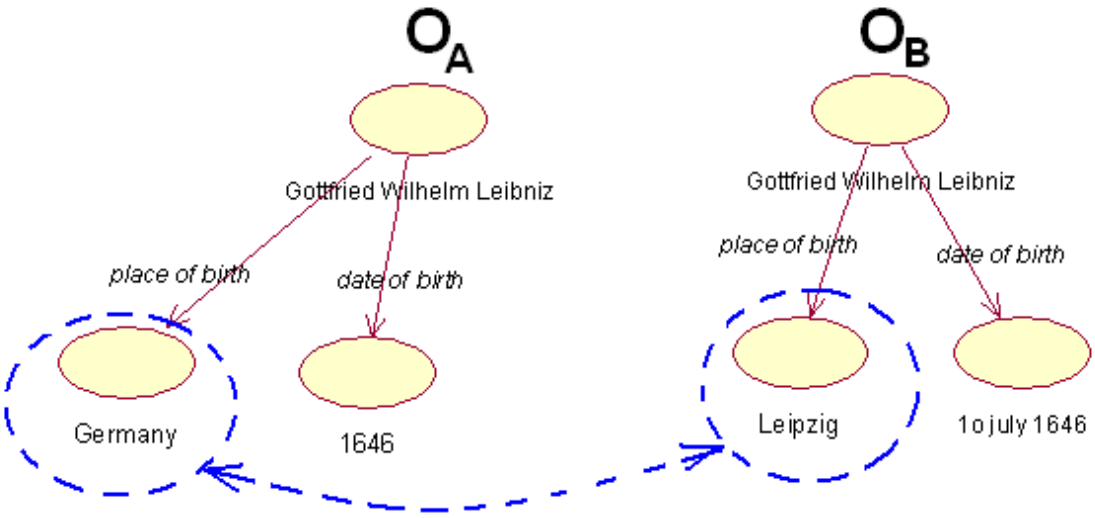


Figure 10. Ontology A says that (place of birth; Gottfried Wilhelm Leibniz; Germany) while B says (place of birth; Gottfried Wilhelm Leibniz; Leipzig). Contrary to our intuition, OM selected (place of birth; Gottfried Wilhelm Leibniz; Germany) as the result to keep in C. The text in §3.7 explains why

4 Results and discussion

We now present some results of the automatic accumulation or merging of ontologies A and B by OM. These two ontologies were merged to produce the result C (another ontology). Each source pair A and B describe the same topic; both must have the same root concept. A and B were manually obtained from different documents in the Web. Then, C was automatically produced by OM. To check this resulting ontology, the same source ontologies were manually fused, this fusion being by definition "correct." The error and efficiency of the result C was computed by the following formulas, where T is the total number of restrictions and concepts in the correct answer:

Error = (number of wrong restrictions and concepts in C) / T

Efficiency = 100 * (number of correct restrictions and concepts in C) / T.

The first column of Table 1 presents the source ontologies. The time that OM took to fuse them appears in column 2. The slowest fusion is for *One hundred years of solitude* because it has more restrictions and OM verifies carefully elements of each restriction. The second column shows details for concept accumulation; the third column, for accumulation of restrictions. These columns also compare the result of OM against the correct result.

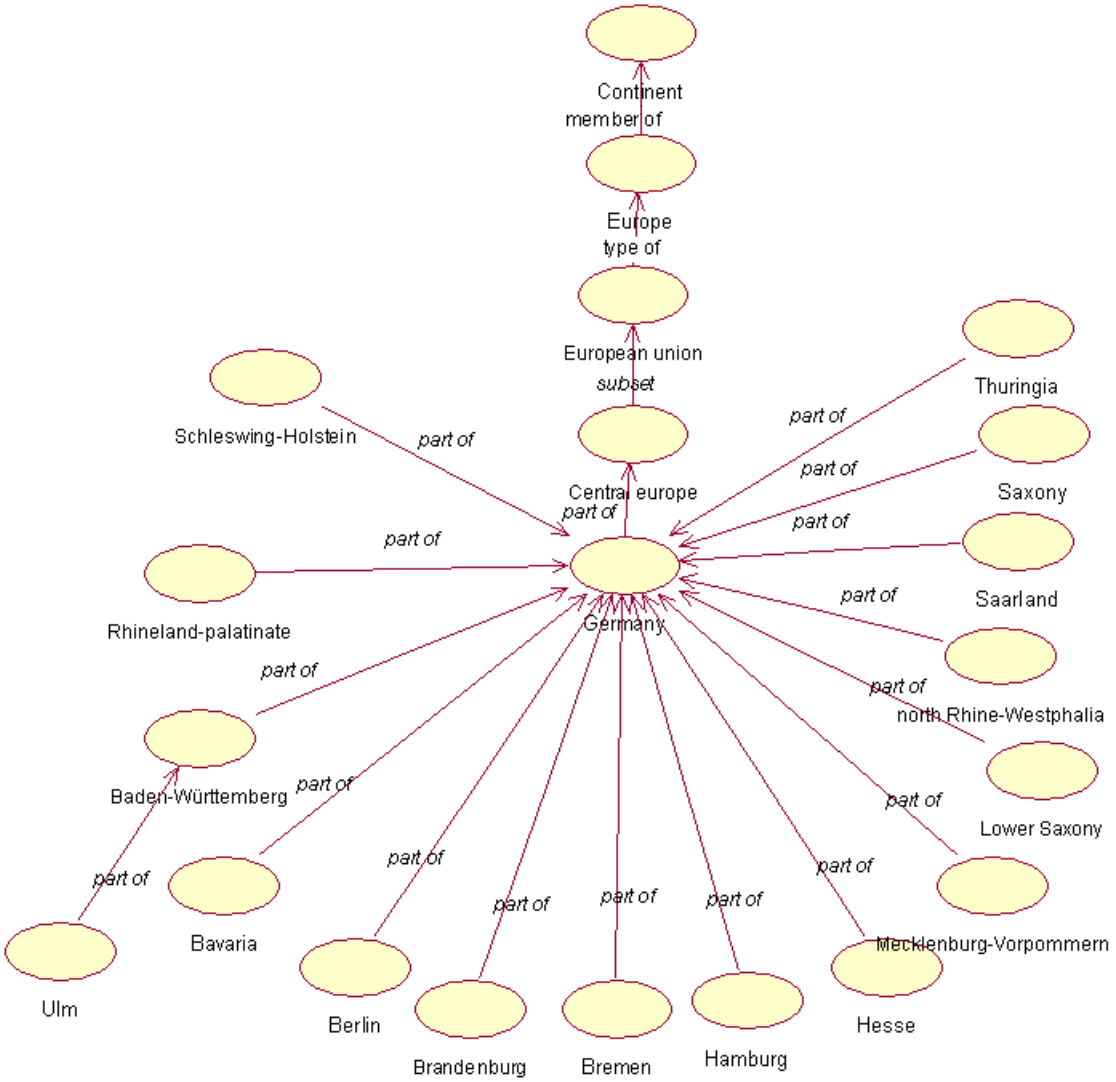


Figure 11. Parts of Germany

4.1 Discussion

Recent ontology fusion methods perform the alignment (suggesting which concepts in A should match with which of B) by comparing names and labels of such concepts and of their neighbors. Fusion (to decide which of the suggested concepts is indeed the right one to be fused) is done with the help of a person. HCONE-Merge [13] uses the data base WordNet [7] to find the meaning of the concepts in the alignment, alleviating the user chores. OM uses the definition of the concepts, their neighborhood and their characteristics. These are verified through a recursive process (each characteristic [a relation] can be also a concept, and it is thus analyzed as such). This recursive process can be interpreted like a semantic search or semantic analysis of the concepts. Instead of relying on a person to do the fusion of the aligned concepts, OM relies on COM (§2.4) and the confusion conf (§2.5), thus achieving automatic fusion.

As an alternative to the subjective way of manually checking the results of OM to gauge its efficiency and error rate, perhaps a better way to verify them would be to use tool (3) of the abstract. With that question-answerer at hand, the user could pose "interesting questions" to the result C. The answers could be verified (manually) to see if indeed they are right. Again, we can not escape from subjectivity. A Ph. D. thesis in progress [2] seeks to generate this question-answerer, but for heterogeneous data bases, not for ontologies.

Table 1. Behavior of OM in some real examples

Source Ontologies	Results for concept accumulation	Results for accumulation of restrictions	error	% Effic
Wine, Figure 4 [24, 25].	The 25 concepts of B were added and fused correctly to the 25 of A, getting a total of 46 concepts in C. 25A + 25B = 46C. (2 sec).	31 restrictions of B were added and fused correctly into 27 of A, getting a total of 55 restrictions in C. 31B + 27A = 55C.	0	100
Neurotransmission (A) and Schizophrenia. (B) [26].	56A + 26B = 77C. The manual method gave 79 (2 of 79 concepts were missing) (2 sec).	79A + 51B = 127C. The manual method gave 129 (2 of 129 were not copied).	0.019	98
Human anatomy.	48B + 28A = 59C (2 sec) [22, 23].	66B + 30A = 104C (New relations were deduced after fusion and added).	0	100
Continent.	54B + 50A = 66C (3 sec) [27, 28].	40B + 34A = 46C.	0	100
Self-inconsistent ontologies.	5B + 6A = 9C. There were 5 original inconsistencies (3 concepts in A and 2 in B were wrongly classified). C had the same 5 original inconsistencies (1 sec).	3B + 4A = 7C. There were 2 inconsistencies (2 restrictions that were not defined, one on each ontology). The result C had 1 inconsistency, OM solved the other.[13]	0	100
One hundred years of solitude.	126B + 90A = 141C. The manual method gave 149 (8 of 149 concepts were missed). (10 minutes). [29, 30]	283B + 231A = 420C. The manual method gave 432 (12 of 432 were not copied).	0.034	96.5
Solar System [31, 32].	60B + 79A = 125C. (Everything was correct) (4 sec.)	45B + 56A = 59C.	0	100
Oaxaca (5 min).	117B + 234A = 309C. Manual method gave 310 (1 of 310 was not copied).	43B + 61A = 96C [33, 34].	0.002	99.7

OM needs to be tried on large, extensive ontologies. Unfortunately, few large ontologies are publicly available now. Of them, most are "shallow" in that they do not have enough restrictions to provide meaning to the nodes; they still rely heavily on the name associated to the node to characterize it (what we call its definition, see caption to Figure 3).

With respect to the repetitive accumulation of knowledge in order to achieve a large resulting ontology (§1.2 and footnote 4), we have not done any practical test, yet. Lack of enough ontologies on the same topic to accumulate is the main reason. Moreover, when carried out, this accumulation must be done *under supervision* (the user should double check that no inconsistency creeps into the result C, specially in the early stages of knowledge accumulation[14]) and *in the right order* (Cf. footnote 3). The first ontologies to be assimilated should be basic, simple, accurate and solid, since they will be the foundation of further knowledge buildup.

Is this automated assimilation or accumulation of knowledge the entrance to "intelligence by computers" or "machine learning," or is it just another way to process knowledge? Is the computer capable to learn by reading documents? These are some of the questions that could be answered (by somebody else, we prefer to keep working ☺).

[13] In this case it is difficult to know what is the correct (manual) answer, since the source ontologies are self-inconsistent to begin with.

[14] As OM accumulates more knowledge, it will be more resistant to inconsistency creeping into its knowledge, since it will know more.

4.2 Recommendations for future work

Of course, the parser and the question answerer, points (1) and (3) in the abstract, should be tackled. Also, some additions to enrich OM and therefore improve the fusion process could be:

A. Use Wordnet, Wordmenu, etc. to improve the disambiguation of terms by increasing its knowledge of synonyms and near by concepts;
B. Annotate the text to be parsed by assigning to each word its correct sense. This is word disambiguation, as in [3].
C. Add built-in knowledge (or better yet, knowledge explicitly represented in OM notation) about the relation part of, to understand that a `boat` is composed of `prow`, `stern`, `deck`, `rudder`, `propeller`... (using perhaps WordMenu). As a generalization, if additional knowledge about relations could be conveniently added *in OM notation*, then OM could reason "more deeply" and better about these relations and their arguments in the corresponding restrictions. For instance, it could construct (deduct) new restrictions. This is opposite to defining this additional knowledge about relations by formal means, by logic equations, by a mathematical notation, or by a programming language, all of them opaque to OM.
D. Currently, OM does not know that things can change. It is not aware of the passage of time. It gets confused when A says that George Bush was governor of Texas, while B says that he was president of USA, since a person can not have both jobs simultaneously (OM ignores that the jobs were held sequentially). We notice that some documents do not mention when something happened, for instance "he left Santa Cruz seminary." Other phrases give a partial knowledge of time, such as: "he visited most of the Mexican republic carrying national documents after that he had been moved from his position office" or similar form. Other writings mention things that happen all the time; permanently, such as "Rouse is Benito's sister," or "Juarez was born in Oaxaca." A relevant work is [17].
E. The same for events that occur in space (different places at the same time). How is it possible that it is both raining in Mexico and not raining in Mexico? (Because Mexico is large).
F. For arguments that are single valued (such as the place of birth), OM keeps just one value, as expected. If it could keep more than one (the "right" or "current" answer and other "less likely" answers), then OM could change its mind if enough evidence arrives, something that actually it can not do (Cf. §3.7). A way to do this is to use the theory of inconsistency, now under development [11].

Acknowledgments. Work herein reported was partially supported by Project 43377 (CONACYT). The authors recognize the partial support of CONACYT and SNI. Prof. Sergei Levachkine has been providing useful suggestions.

References

URLs were last time consulted on November 17, 2008.

1. **Bechnofer, S., van Harmelen, F.,** Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P. F., Andrea, L. (2004) *OWL Web Ontology Language Reference*. W3C Recommendation. http://www.w3.org/TR/2004/REC-owl-ref-20040210/
2. **Botello, A**. Inferring relations between autonomous data bases. CIC-IPN. Ph. D. thesis in progress. In Spanish.
3. **Colorado, F.** Mapping words to concepts: disambiguation. M. Sc. thesis in progress. CIC-IPN. In Spanish.
4. **Connolly, C., Harmelen, F.,** Horrocks, I., McGuinnes, D. L., Patel-Schneider, P. F. Andrea Stein L. (2001) *DAML+OIL Reference description*. W3C Note 18. http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218
5. **Cuevas, A.** Ontology union using semantic properties. (2006) Ph. D. thesis, CIC-IPN. In Spanish.
6. **Dou, D., McDermott, D**., and **Qi, P**. (2002) Ontology Translation by Ontology Merging and Automated Reasoning. In Proc. *EKAW Workshop on Ontologies for Multi-Agent Systems*.
7. **Fellbaum, C**. (1999) WordNet An Electronic Lexical Database. Library of Congress Cataloging in Publication Data.
8. **Fridman, N.,** and **Musen, M**. (2000) PROMPT: Algoritm and Tool for Automated Ontology Merging and Alignment. In Proc. *Seventeenth National Conference on Artificial Intelligence*. pp 450-455, Austin, TX, USA.
9. **Genesereth, M.** *Knowledge Interchange Format. Draft proposed American National Standard* NCITS.T2/98-004.
10. **Guzman, A., and Olivares-Ceja, J.** (2006) Measuring the understanding between two agents through concept similarity. *Expert Systems with Applications* **30**, 4 577-591, Elsevier.
11. **Jimenez, A.** Characterizing and pondering logical properties on qualitative values organized into hierarchies. Ph. D. thesis in progress. CIC-IPN. In Spanish.

12. **Kalfoglou, Y**., and **Schorlemmer, M**. (2002) Information-Flow-based Ontology Mapping. Proceedings of the 1st *International Conference on Ontologies, Databases and Application of Semantics* (ODBASE'02), Irvine, CA.
13. **Kotis, K**., and **Vouros, G.**, **Stergiou, K**. Towards Automatic of Domain Ontologies: The HCONE-merge approach. *Elsevier's Journal of Web Semantic (JWS)* **4**:1, pp 60-79.
14. **Levachkine, S., and Guzman, A.** (2007) Hierarchy as a new data type for qualitative variables. *Expert Systems with Applications* **32**, 3, 899-910.
15. **Manola, F., Miller, E.** *RDF Primer. W3C Recommendation*. 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
16. **McGuinness, D., Fikes, R., Rice, J.**, and **Wilder, S**. (2000) The Chimaera Ontology Environment Knowledge. In Proceedings of the *Eighth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues (ICCS 2000)*. Darmstadt, Germany.
17. **Puscasu, G., Ramirez Barco P**, *et al*. (2006) On the identification of temporal clauses. *LNAI* **4293**, 911-921.
18. **Reed, S. L.,** and **Lenat, D.** (2002) Mapping Ontologies into CyC. In proceeding of *AAAI Workshop on Ontologies and the Semantic Web*, Edmonton, Canada.
19. **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 (FMII'01)*, Viterbo, Italy, 2001. lNAI, Springer (In press).
20. **WEB Onto.** http://137.108.64.26:3000/webonto?ontology=AKTIVE-PORTAL-ONTOLOGY&name=TECHNOLOGY&type=CLASS

References in Internet

Internet references [21, 24, 25] are in English; the rest are in Spanish.

21. Loom. http://www.isi.edu/isd/LOOM/LOOM-HOME.html
22. Human anatomy (A). From text in: http://es.wikipedia.org/wiki/Anatom%C3%ADa_humana
23. Human body (B) was built from text in: http://www.salonhogar.net/CuerpoHumano/Introd_Cuerpo_humano.htm
24. Wine (B). From text in: http://www.studyworld.com/newsite/ReportEssay/CreativeWriting/PersonalEssays%5CClassification_of_Wine-40444.htm.
25. Wine (A). From text in http://en.wikipedia.org/wiki/Wine
26. We thank Paola Nery www.geocities.com/paolaneriortiz/ because she manually converted two documents into ontologies: neurotransmisor (A): es.wikipedia.org/wiki/Neurotransmisor and esquizofrenia (B): www.nimh.nih.gov/publicat/spSchizoph3517.cfm.
27. Continent (B) was produced from text in: http://es.wikipedia.org/wiki/Redefinici%C3%B3n_de_planeta_de_2006.
28. Continent (A) was obtained from text in http://es.wikipedia.org/wiki/Continente
29. 100 years (A) was from text in http://html.rincondelvago.com/cien-anos-de-soledad_gabriel-garcia-marquez22.html (document no longer available)
30. 100 years (B) was obtained from text in http://www.monografías.com/trabajos10/ciso/ciso.shtml
31. Solar System (B) comes from text in: http://www.solarviews.com/span/solarsys.htm.
32. Solar System (A) was formed from text in http://es.wikipedia.org/wiki/Sistema_Solar
33. Oaxaca (A) was obtained from text in www.oaxaca-mio.com/atrac_turisticos/infooaxaca.htm
34. Oaxaca (B) was extracted from text found in www.elbalero.gob.mx/explora/html/oaxaca/geografia.html