Measuring the understanding between two agents through concept similarity

Adolfo Guzman-Arenasm Jesus M. Olivares-Ceja
Centro de Investigacion en Computacion (CIC), Instituto Politecnico Nacional, Mexico
{a.guzman, jesuso}@acm.org

SUMMARY. Two agents previously unknown to each other cannot communicate by exchanging concepts (nodes of their own ontology): they need to use a common communication language. If they do not use a standard protocol, most likely they use a natural language. The ambiguities of it, and the different concepts the agents possess, give rise to *imperfect* understanding among them: How closely concepts in ontology O_A map[1] to which of O_B? Can we measure these mismatches?

Given a concept from ontology O_A, a method is provided to find the *most similar concept* in O_B, and to measure the similarity between both concepts. The paper also gives an algorithm to gauge **du**(A, B), the *degree of understanding* that agent A has about the ontology of B. The procedures use word comparison, since no agent (except the Very Wise Creature, VWC) can measure **du** directly. Examples are given.

KEY WORDS: Ontology matching, natural language, concept similarity, degree of understanding, imperfect knowledge.

1. Introduction and objectives

Agents A and B having to communicate with each other in order to achieve their goals, do so in one of two ways:

(a) (Easier) Through a private language or protocol and naming convention, which requires prior agreement among programmers' team or a Standards Committee;

(b) (More general) Through a common popular general-purpose language, most likely a natural language.

This paper addresses concept communication in the (b) setting. Two points stand: (1) Such communication can not be fulfilled through direct exchange of concepts belonging to an ontology, since A and B *do not share* the same ontology (they do not know *exactly* the same, from the same point of view), and O_A and O_B are in different *address spaces*. (2) Unlike case (a), the communication language is very often ambiguous. Together, (1) and (2) give rise to imperfect understanding (confusion).

Knowledge is stored in concepts, which are mapped by the talker into words of the communication language; perceived words are internalized as concepts by the listener. If the concepts exchanged are animals and plants, Latin is fine: *Felix Leo* represents the con-

[1] O_A and O_B are the ontologies of agents A (the talker or sender) and B (the listener), in the rest of the paper.

cept lion-león-loin[2] while *Canabis Indica* stands for the concept marijuana. Other examples of words or symbols with a unique (universal) meaning: 4, π, Abelian group, Mexico, (23° 22' 57"N, 100° 30'W), Abraham Lincoln, Berlin Symphony Orchestra. There are also semi-universal (popular) conventions, such as standard naming for chemical compounds; the Library of Congress Catalog for books, or the USA Social Security Number; they provide non-ambiguity for those who adhere. If two agents can select a non-ambiguous language (each of its words maps exactly to one concept) or convention to exchange concepts, great: they fall in case (a). If not, they have to settle for an ambiguous language, such as English [10], falling in case (b).

When A talks to B, can either discover *what* produces the confusion? Is it possible to measure it? How can I be sure you *understand* me? Can I *measure* how much you understand me? Can *you* measure it? These questions have intrigued sociologists; they are also relevant to agents "not previously known to each other"[3] trying to "interact with free-will",[4] for which they have to exchange knowledge. *The paper provides answers to: (i) What is the most similar concept* $c_B \in O_B$ *to concept* $c_A \in O_A$*? How similar is* c_B *to* c_A*? (ii) how much does B know about* O_A*?* If two agents do not share a concept, at least partially, they can not communicate it or about it. Thus, a measure of the amount of understanding is the number of concepts they share, and *how well* they share them.[5] We will sharpen these measures for both the ambiguous and the non ambiguous communication language.

1.1 Related work

Huhns [12] seeks to communicate several agents sharing a single ontology. The authors have constructed [9, 10] agents communicating with previously unknown agents, so that not much *a priori* agreement between them is possible.[3]

An ancestor of our *sim* (§3.1) matching mechanism is [4], based on the theory of analogy. Most work on ontologies involve the construction of a single ontology [for instance, 15], even in those that do collaborative design [11]. Often, ontologies are built for man-machine interaction [13] and not for machine-machine interaction. Everett [3] tries to identify conceptually similar documents, but use a single ontology. Gelbukh [5, 6] does the same using a topic hierarchy: a kind of ontology. Linguists [18] identify related *words* (semantic relatedness), not *concepts*, often by statistical comparisons.

With respect to measuring how close is a concept to another, Levachkine [16, 17] makes simpler measurements between qualitative values ("words," you can say) belonging to a hierarchy, to measure their *confusion.* More at §3.1.2.1 below.

With respect to the communication language, we prefer [2] in decreasing order:

[2] We represent concepts in Courier font or with a subscript c. A concept is language-independent: the concept cat is the same as the concepts gato-gata, gatto-gatta, chat-chatt, Katze, КОТ-КОШКА, ねこ (neko), meaning "a small domestic feline animal". Concepts appear in English in this paper, for readers' benefit.

[3] It is much easier to design the interaction (hence, the exchange of concepts) between two agents (or pieces of software), when the *same* team designs both agents. In this sense, "they previously know each other:" each agent knows what the other expects, when (the calling sequences), and the proper vocabulary to use.

[4] Not-free will or canned interactions are those that follow defined paths. For instance, the interaction between a program and a subroutine it calls, where the calling sequence (of arguments) is known to both. Free will usually requires goals, resources, and planning [19].

[5] Knowledge is also stored in the relations (or verbs, actions, processes) between objects (or nouns, subjects): §1.2.

1. A language whose words (tokens, atoms) are formed by concepts [19];
2. One with unambiguous words (a word represents only one concept). Examples: the Natural Numbers; the Proper Nouns;
3. One where each word has a small number of ambiguities, for instance, a natural language [18] (we use this option in this paper; see figure 1);
4. One where each word points to a foreign address space, thus representing “black boxes” that can only be compared with = and ≠. Example: a language consisting of tokens such as “toves,” “borogove,” “mome”, “outgrabe”, “fromelic,” “meratroping,” tokens that only admit equality as their comparator.

These are the four choices to name a concept. Often, an ontology skips details by just keeping the names (and nothing else) of a concept, as when it says `John [owns=wardrobe]` and it does not say anything else about `wardrobe`. It is a *shallow* comcept.

The book [1] gives other approaches and experimental results in semantic analysis.

```
thing (thing, something, object, entity) {
  food { vegetable [contain=chlorophyll, structure=living]
    { onion [color=white]
      lemon [color=green, shape=circle_shape, taste=sour, size=3cm] }
    fruit { orange [color=orange_color, shape=circle_shape] }
    meat {pork [texture=soft, color=pink] beef [texture=medium, color=red]
      turkey [texture=medium, color=white] veal [texture=soft, color=red] }
    seed (seed, cereal) { rice [color=white, shape=oval_shape]
      wheat [color=yellow, shape=oval_shape] } }
  tool (tool, hardware) { wrench (wrench)
    electrical_tool (household, electrical_tool) {
    dishwasher [use=cleaning, structure=manmade, size=small, covering=paint]
    blender [use=blend, structure=manmade, covering=paint]
    drill  mixer [use=mix, structure=manmade] } }
  furniture { bed table sofa (sofa, seat) bench }
  pet { cat [covering=hair, family=feline]
    dog [covering=hair, family=canine] canary [covering=feather] }
  relation { color { white yellow green  purple orange_color }
    shape (shape, contour) { circle_shape (circle, oval) square_shape (square ) }
    taste { sweet sour } size { 3cm (3cm, 1-2in) }
    covering { hair feather paint (paint, painted) }
    structure { living (living, alive) manmade (handcrafted, manmade) }
    contain (contain, content) { chlorophyll }
    family { feline canine } use { cleaning blend mix }
  matter_states (matter_state) {liquid suspension gel solid gas } } }
```

O_P

Figure 1. O_P, the ontology of person P. Concepts such as `sofa` appear in `Courier font`, while the words *w*(`sofa`) = (sofa, seat) (Cf. §1.2) denoting the concept appear in Times Roman. These words do not appear in the listing if identical to the concept denoted. Thus, we see `onion` and not `onion` (onion). Facts (Cf. §1.2) appear as `cat [covering=hair]`, meaning that, for `cat`, the value of relation `covering` is `hair`. O_P contains 68 concepts.

`thing` (thing, something) {

pharmacy (pharmacy, drugs, medicine, medicament) {
self_service {
personal_care {
mouth_care feminine_hygiene burns (burns, solar_protector) repellent }
diet (diet_substance) { diet_powders (diet_powders, diet_creams)
diet_solutions diet_tablets }
stomach { laxatives_suspension [is_a=suspension]
laxatives_suppository [is_a=gel]
laxatives_tablets [is_a=solid, shape=circle] antiparasites } }
patent_pharmacy }
general_merchandise {
hardware { bathroom_kitchen electrical { socket multicontact }
lightbulb manual screw } }
cloth {ladies_stocking hosiery shoe }
perishable { frozen_food meat { pork beef turkey veal }
fruit_vegetable {
garlic_onion { garlic white_onion (onion) [color=white]
yellow_onion (onion) [color=yellow] purple_onion (onion) [color=purple] }
bulk_fruit { bulk_citrics (citric) exotics annual_fruits season_fruits }
packed_fruits {avocados packed_citrics (citric) } }
cooked (cooked_food) seeds {rice cereal bean }
bulk_vegetable {gourd potato lemon [color=green,
shape=spheric_shape, size=3cm, taste=sour] }
seafood (fish_seafood) { fish (fish) preparations (fish_preparations) } }
groceries { oils (eatable_ oil) { aerosol_oil sunflower_oil corn_oil
mixed_oil olive_oil }
rice (seed_rice) {extra_rice integral_rice shushi_rice
precooked_rice super_extra_rice (super_extra_rice) }
sugar (sugar, sweetener) { aspartame (aspartame, diet_sugar) glass_sugar (sugar)
black_sugar (sugar) refined saccharin standard }
coffee {decaffeinated_grain grain sweetened_coffee (sweetened)
normal_soluble sweetened_soluble decaffeinated_soluble
special_soluble }
hot_cereal { oats mixed granola }
cold_cereals { cereal_bars(bar) fiber_bars granola_bars energetic_bars
filled_bars cold_rice_cereal (rice) cold_oats_cereal (oat)
cold_corn_cereal (corn) cold_wheat_cereal (wheat)
cold_mixed_cereal (mixed_cereal) diet_cereal }
chili{ancho_chili tree_chili guajillo_chili morita_chili pasilla_chili }
dry_fruits(dry, fruits) {prune coconut jamaica tamarind raisin (raisin, dry_grape)}
flour (flour) { rice_flour (rice) wheat_flour (wheat) corn_flour (corn)
hotcakes_flour (hotcakes, flour) pastries_flour (pastries, flour)
powder (powder) other_flour (barley, rye, flour) } }
other { relation { color { white yellow green purple }
shape (shape, contour) {circle (circle_shape) spheric_shape (sphere)
square_shape (square) }
taste { sweet sour } size {3cm (3cm, 2in) } }
matter_states (matter_state) { liquid suspension gel solid gas } } }

Fig. 2. Superama ontology (150 concepts). To save space, words of a concept do not appear when identical to it

1.2 Definitions

Named entity. An object, relation, property, action, process, idea or thing that has a name: a word (or word phrase) in a natural language. ♦[6] A named entity is shared.

Concept. The representation of a named entity. ♦[7] Examples: `peak-uttermost`, `to_-fly_in_air`, `angry-mad`. So, *concepts have names*: those words (or word phrases, such as New York City) used to denote the named entity that the concept represents.[2] Unfortunately, the names given by different people to concepts differ (*synonymy*) and, more unluckily, the same word is given to two concepts (examples: words peak; fly; mad). Thus, *words are ambiguous*, *while concepts are not.* A person or agent, when receiving words from a speaker, has to solve their ambiguity in order to understand the speaker, by mapping the words to the "right" concept in his/her/its own ontology. This mapping is called *disambiguation*.

There are also composite or complex concepts, such as "to ski in a gently slope under a fair breeze while holding in the left hand a can of beer." These can be shared with other agents, too, but they do not possess a name: they have not been reified (Cf. §2.2).

Concepts can enter also in relations (called restrictions in some Logics) with other concepts, as defined below.

A concept represents three kinds of real-world entities: (a) *sets*, like `animal` (represents all the animals); (b) *individuals*, like `Abraham Lincoln` (represents a particular person), and (c) *relations*, such as `buys`.

Relation. A relation of order k is a sequence of concepts (rel c_1 c_2 … c_k) representing that relation `rel` holds between c_1, c_2, .., c_k. ♦ Notice that `rel` is also a concept (which may be shallow).

Notation. For binary relations (rel c_1 c_2) we write in the ontology `rel [c1=c2]`, for instance `lemon [color=green]` in figure 2. Notice the use of `c {p q r …}` to indicate that the subsets of `c` are `p`, `q`, `r`… More at [19].

Fact. A relation is a *fact* if it holds in reality. ♦ It agrees with the real world. It faithfully represents an aspect of reality. To discover facts is not easy: people make tedious experiments and observations to ascertain what "really happens in the real world" and what does not. Thinking or philosophizing is not enough. We are not interested in how to discover facts. Thus, we postulate that an agent "knows" or "believes" that its knowledge consists of *only facts:* truthful representations of reality. It is possible for some $rel \in O_A$ not to be a fact, but A does not know this. An agent does not lie to itself. Relations not agreeing with reality are "lies."

Knowledge is the concrete internalization[8] of *facts* among real-world entities ♦[6] It is stored as facts and as concepts; it is *measured* (*grosso-modo*) in "number of concepts."

Ontology. It is a formal explicit specification of a shared conceptualization. [7] ♦ It is a taxonomy of the concepts I want to represent.[9] See figure 1.

---

[6] Symbol ♦ means: end of definition. Having a name in a shared language means that is known to many people.

[7] We differ from Formal Concept Analysis (FCA) where a *concept* is *any* subset of attributes (or the corresponding set of objects) while we narrow *concept* to represent only an objects or thing that has a name in a natural language.

[8] By "concrete internalization" we mean writing down (storing) the fact as a relation, for later use.

[9] Each concept that I know and has a name is shared, since it was named by somebody else. More at §2.2.

Our definition. O = (C, R, `root`), an ontology is a structure formed by *concepts* and *relations* among concepts. Each concept that represents a set (such as the concept `animal`) necessarily holds the relation `subset` with some other set, except the distinguished concept `root`, which is subset of nobody, and which is superset of all other sets. Each concept that represents an individual must be in relation `member of` with some set-representing concept. ♦ Each concept and relation has as name (or has associated) words from a natural language, since the ontology tries to represent a part of the real world.

The required presence of relations `subset` and `member of` give an ontology a tree-like or taxonomy-like appearance; nevertheless, a set (`apple`) may be subset of more than one sets, such as `fruit` and `food`, and a person may be member of several sets.

Word(s) associated with a concept: $w(\mathtt{c}_A, O_A)$ = the words associated in O_A to concept $\mathtt{c}_A$.

Example: (figure 1): $w(\mathtt{tool}, O_P)$ = (tool, hardware).

2. Degree of knowledge about a concept; amount of knowledge of an agent

How much does an agent know? In this section, we measure the amount of knowledge of an agent against the "total ontology" (an impractical abstraction). Later (§3.2), we measure its knowledge *relative to the knowledge of another agent.* The idea is to know first how much an agent knows about a given concept, and then sum these amounts over all its known concepts.

Total ontology. It is the ontology O_K of an agent K that knows *everything known to every agent.* ♦ It is the union of all the ontologies of all the intelligent creatures, people and agents. Observations: 1. The total ontology is finite, and grows every day. 2. It is unique. Idea: think of a huge encyclopedia. 3. It contains only facts.

To comply with 2 and 3 above, when trying to merge contradicting facts, K must discover *what "fact" is a lie.* K could keep prevailing knowledge, but this can produce widespread knowledge that is nevertheless wrong, such as `(Earth-World shape-form flat-plane)`. Truth is not discovered by majority voting.

Since it is not the purpose of this article to unveil a method to find facts, we simply postulate that the total ontology of K is formed with the help of a god or Very Wise Creature (VWC) that makes sure lies coming from some person or agent do not enter it. Thus, O_K contains only *facts*,[10] while an agent or person can still have parts of its/his/her ontology formed by lies. O_K holds all known and "right" knowledge. The degree of knowledge (dk, §2.1) of an agent is measured against O_K.

[10] This *total ontology* grows every day, as new discoveries are performed. Also, notice that the ontology of a VWC must be larger or at least equal than the *total ontology.* Nevertheless, a VWC **can not** easily communicate its "excess knowledge" to any person or agent A, because A will not understand some trios coming from VWC. Nevertheless, VWC can (sequentially) *teach* this new knowledge to A, cf. §3.3.1. In most cultures' cosmogony, there is a god that did just that. Such god is a VWC. In fact, we can define a VWC as an agent who has the *total ontology.* K=VWC ♦

2.1 Degree of knowledge of an agent about a concept

This degree is a measure of the (imperfect) grasp of a concept by an agent A. The idea is that the more relations that concept has in O_A, the larger such degree of knowledge is.

The *degree of knowledge* $dk_A(\mathtt{c})$ of A about a concept `c` is a number between 0 and 1, obtained by counting the number of arcs connecting to/from `c` in O_A, adding the number of arcs labeled `c` in O_A,[11] and dividing into the similar calculation for `c` in the total ontology O_K. ♦ The closer $dk_A(\mathtt{c})$ is to 1, the less imperfect is A's knowledge of `c`.

2.1.1 Amount of knowledge of an agent

The idea is that every concept in O_A contributes to the amount of knowledge of A.

The amount of knowledge an agent has = Σ $dk_A(\mathtt{c_i})$ over all $\mathtt{c_i} \in O_A$. ♦ It is measured against the total ontology. It is approximated by the area under the histogram of concepts (see Clasitex in [8]). Similarly, the *amount of knowledge* of an agent *in a discipline* or area $D \subset O_K$ is Σ $dk_A(\mathtt{c_i})$ over all $\mathtt{c_i} \in D$.

Definitions 2.1 "Degree of knowledge of an agent about a concept" and 2.1.1 "Amount of knowledge of an agent" are impractical since the total ontology is out of our reach. In their place, we shall define and compute,

i.) (instead of 2.1) The degree of knowledge of an agent about a concept *with respect to other agent's knowledge* of such concept, to be called in §3.1 the *similarity value* (sv).

ii.)(instead of 2.1.1) The amount of knowledge of an agent with respect to another agent's knowledge, to be called in §3.2 the degree of understanding (du) of A about B: how much A understands about what B knows.

2.2 Reification slightly increases knowledge

Reification. The action of exposing the internal representation of a system in terms of programming entities that can be manipulated at runtime. The opposite process, absorption, consists of effecting the changes made to reified entries into the system, thus realizing the causal connection link [14]. ♦ To reify a (usually complex) concept or relation is to represent it by an atomic symbol ("to give it a name"), probably because we want to say more complex things about it; for instance, to affect it by other relations.

Example: The complex concept "When Bill Clinton tries to convince a person about x, he explains and praises very much and with abundant examples the good things about x, but (without skipping any) explains lightly the bad things about x, thus giving the false appearance of an unbiased explainer"[12] can be reified by giving it a name: clintonize (x). Then we can speak of clintonization, quasi-clintonization, and express clearly relations such as "clintonizing is immoral."

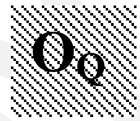
An ontology $O_A' = O_A \cup \mathtt{clintonize}$ has one more concept than O_A: the concept `clintonize`. A private concept, shared by few; not very useful for communications.[13] In

[11] This makes sense when `c` is a relation.

[12] I believe this definition was introduced by President George Bush (Sr.) during a debate with then-candidate Bill Clinton.

[13] An agent that reifies a concept could find this very useful for *its own work* (not for communication with others), much as an experienced locksmith invents a new tool by modifying other, in order to work faster: a private tool.

our ontologies we will not cache (store) derived concepts [9], unless they are shared by many people (by having a name and a definition in a certain dictionary, say); that is, unless they are *reified* and given the same name by many people.




```
thing (thing, something, entity) {
   eatable_thing (food, foodstuffs, groceries) {
      plant {
         vegetable [structure=living] {lettuce tomato onion
            potato (potatoes, potato, pome_de_terre) }
         fruit { strawberry  banana }
         tree { lemon [color=green, shape=circle_shape]
            orange [color=orange_color] }
         cereal {rice[color=white]  wheat[color=light_brown] }}}
   animal (animal, creature) [structure=living] { bird [covering=feather] {
         canary[size=small, covering=feather, color=yellow] }
      reptile[covering=scale] {
         lizard [size=medium, covering=scale, color=green] }
      mammal [covering=hair] {cat [covering=hair, size=medium, family=feline]
            dog [covering=hair, size=medium, family=canine] } }
   inanimate_thing (inanimate, object) {
         tool (tool,hardware) { manual (manual_tool) { wrench  hammer }
            appliance (appliance, electrical) { drill [use=boring]
               electrical_saw (saw) [use=cutting] dishwasher [use=cleaning]
               blender [use=blend] } }
         furniture { bed  table (table, surface) sofa (sofa, place) chair (chair, seat) }
         energy
         water }
   relation {
         color { white yellow purple green light_brown orange_color }
         shape (shape, contour) { circle_shape (circle, oval, round) square_shape (square ) }
         size { small medium }
         structure { living (living, alive) manmade (handcrafted, manmade) }
         covering { hair feather }
         family { feline  canine }
         use { cleaning blend mix } |
   matter_state (matter_state) { liquid  suspension gel  solid gas plasma } }
```

Figure 3. Ontology of person Q that buys in the supermarket of figure 2. It has 75 concepts.

3. Measuring the degree of understanding (comprehension)

This section finds the concept $c_B \in O_B$ most similar to $c_A \in O_A$, and how similar they are. From this, adding the similarities over all $c_B \in O_B$, we measure the degree of understanding of A with respect to O_B. How well A knows the concepts in O_B.

3.1 Finding the concept in your ontology most similar to one I have

Consider two persons P and Q buying something specific at a real supermarket (www.superama.com.mx; its real ontology contains more than 500 concepts, we have chosen 150 of them for Ontology S, figure 2). Their ontologies are shown in figures 1 and 3. If a person does not find exactly what he wants, he would like to know what is the most similar item available.

The *sim* algorithm (called "*hallar* (c_A)" or COM in [19]) finds *the most similar concept* $c_B \in O_B$ to concept $c_A \in O_A$. It also computes a similarity value $sv \in [0, 1]$ expressing how similar was c_B to c_A. Agent A makes known concept c_A to B by sending to B $w(c_A, O_A)$ (words[14] denoting c_A) , and also sending words $w(p_A, O_A)$ denoting the father p_A of c_A. Four cases exist for $c_B = sim(c_A)$.[15]

Case a) *Node and father match.* (Figures 4, 5) We look in O_B for nodes p_B and c_B such that:

(1) $w(c_B, O_B) \cap w(c_A, O_A) \neq \emptyset$ (the intersection between the words associated to c_B and those associated with c_A is not empty); and

(2) $w(p_B, O_B) \cap w(p_A, O_A) \neq \emptyset$ (the intersection between words associated to p_B and those associated to p_A is not empty); and

(3) p_B is the father, grandfather or great-grandfather[16] of c_B.

If such p_B and c_B are found, then c_B is the answer and the algorithm returns $sv = 1$, too.

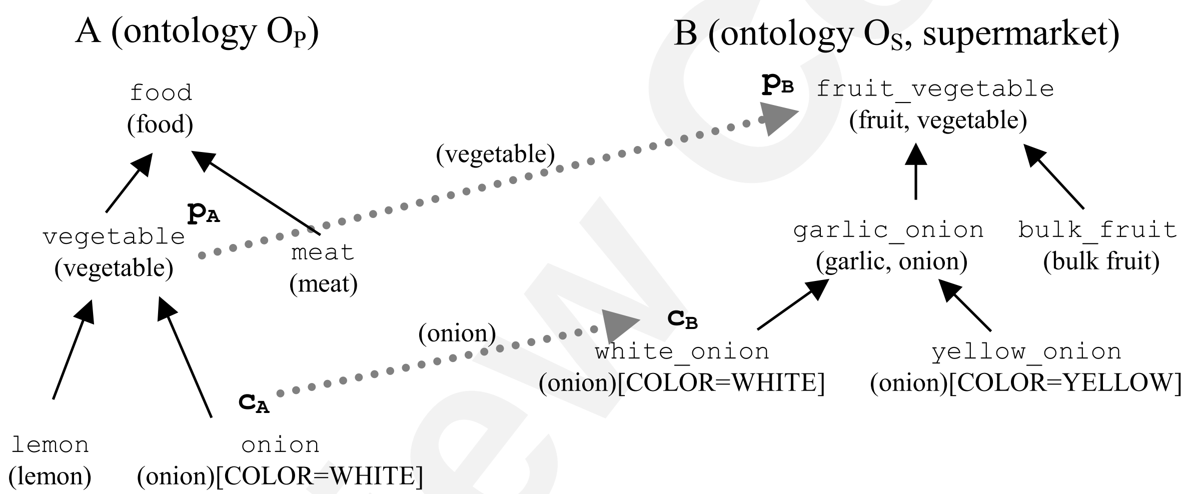


Figure 4. Case a) Agent P (with ontology O_P, figure 1) wants to find the most similar concept to c_A = `onion` in O_S (figure 2). Words (shown inside round parenthesis) from c_A map to different concepts in B, but the most similar concept found (c_B) is `white_onion`. We see that p_A maps into p_B, the grandfather of c_B. *sim* does not compare concepts directly; it compares their words.

[14] By §1, A can not send any *node* of O_A to B.

[15] Rigorously, *sim is* a function of two variables that returns two values, so it should be written $(sv, c_B) = sim(c_A, O_B)$.

[16] If p_B is found more than three levels up, the "semantic distance" is too high and *sim* says "no match."

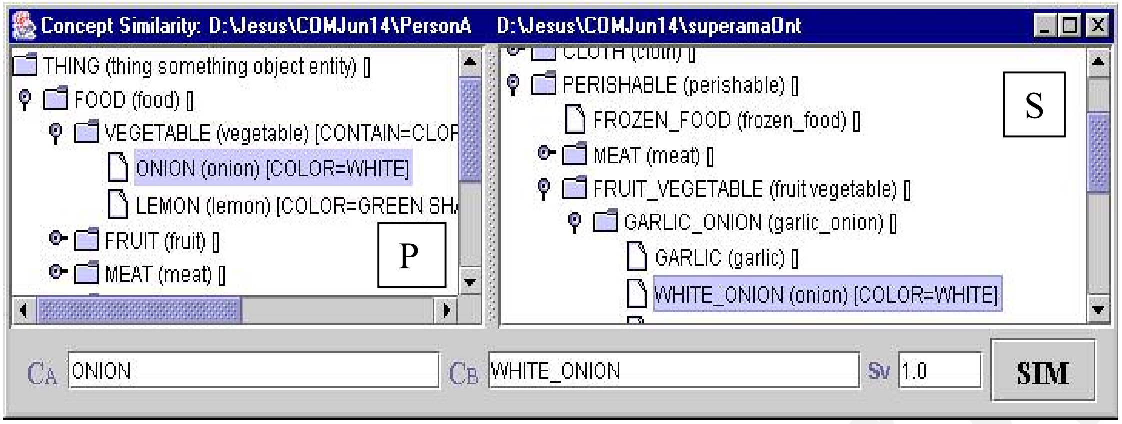


Figure 5. Case a) Father and son in O_A find matches in O_B. There are in O_B three candidate nodes, the first that matches is the most similar concept, in this case `white_onion`.

Case b) *Father matches but node does not.* Figures **Error! Reference source not found.** and **Error! Reference source not found.**. This case occurs when (2) of case (a) holds, but (1) and (3) do not. p_B is found in O_B but c_B is not. In this case, *sim* is called recursively, and we try to compute $p_B' = sim(p_A)$ to confirm that p_B is the ancestor of c_A, the concept of interest.

(1) If the p_B' found is `thing`, the root of O_B, the algorithm returns `not_found` and concludes; $sv = 0$;

(2) Otherwise, a certain child of p_B, to be called c_B', is searched in O_B, such that:

A. Most[17] of the facts of c_B' coincide with the corresponding facts of c_A. Children of p_B with just a few matching facts[17] are rejected. If the candidate c_B' analyzed has children, they are checked (using *sim* recursively) for a reasonable match[17] with the children of c_A. If a c_B' is found with the desired properties, the algorithm reports success returning c_B' as the concept in O_B most similar to c_A. Then sv = the fraction of facts of c_B' coinciding with corresponding facts of c_A.

B. Otherwise c_B' is sought among the sons of the father (in B) of p_B; that is, among the brothers of p_B; if necessary, among the sons of the sons of p_B; that is, among the grandsons of p_B. If found, the answer is c_B'. sv = the sv returned by c_B' multiplied by 0.8 if c_B' was found among the sons of p_B,[18] or by $0.8^2 = 0.64$ if found among the grandsons of p_B.

C. If such c_B' is not found, then the node nearest to c_A is some son of p_B, therefore *sim* returns the remark (`son_of` p_B) and the algorithm concludes. $sv = 0.5$, an arbitrary but reasonable value. For example, in Figure 6, A sends words that correspond to the pair (c_A = `drill`, p_A = `electrical`), whereas B has the concept `electrical` but doesn't have `drill` nor any similar. In this case, the concept $\texttt{drill}_A$ is translated by B into $\texttt{(son\_of electrical)}_B$, which means "some $\texttt{electrical}_B$ I don't know" or "some $\texttt{electrical}_B$ I do not have in my ontology.

[17] We have found useful the threshold 0.5: more than half of the compared entities must coincide.
[18] We have found that 0.8 allows for a fast decay as one moves up from father to grandfather and up.

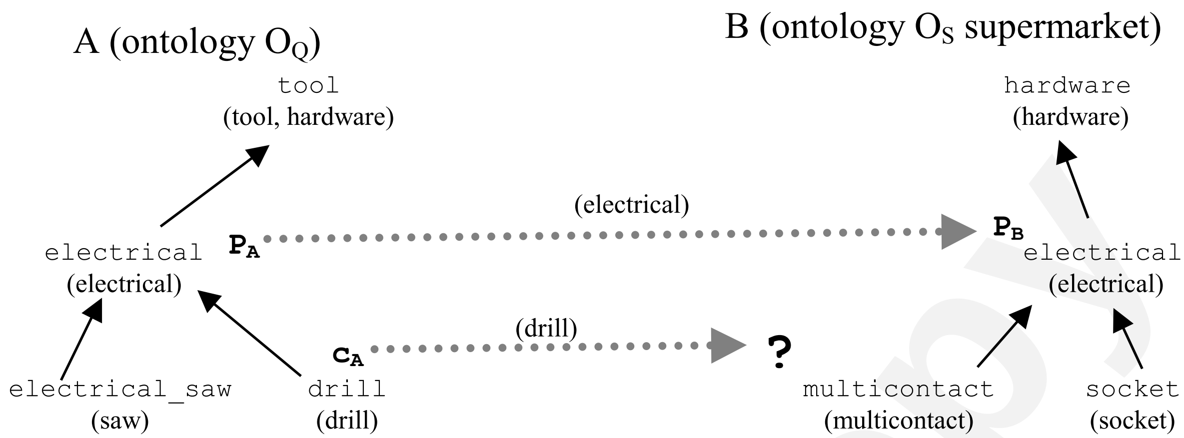


Figure 6. Case b). Agent A (with ontology of person Q, figure 3) wants to find in ontology O_S the concept most similar to c_A = `drill`. Words from p_A map to words from p_B but c_A has no equivalence in the ontology O_S of B.

Figure 7 shows the execution of *sim* for case (b)2(C). In this case concept $\texttt{drill}_A$ has no equivalent in B. Here $\texttt{son\_of\_electrical}_B$ is chosen from B as the most similar concept because parents $\texttt{electrical}_A$ and $\texttt{electrical}_B$ coincide and were confirmed. We assign *sv* = 0.5

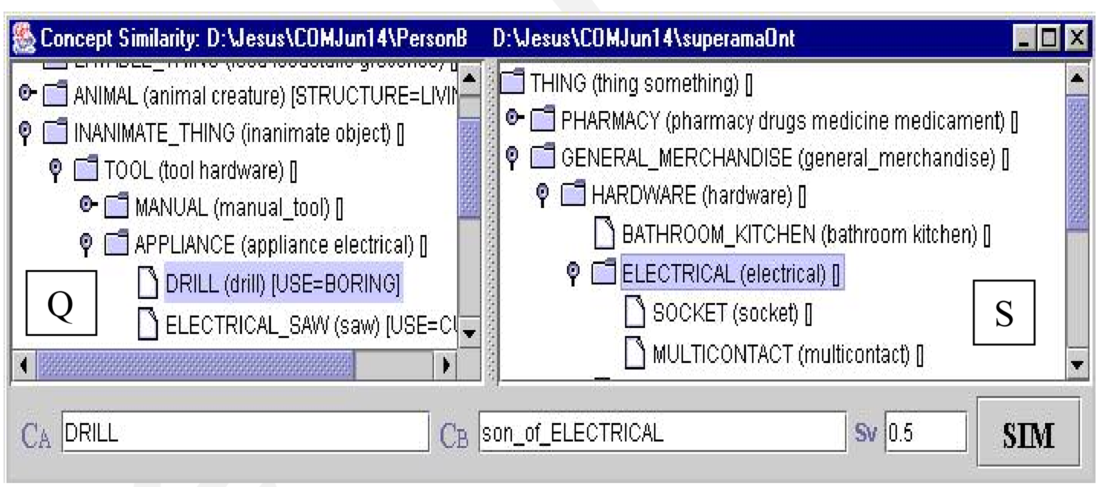


Figure 7. Case b) The father of c_A finds its corresponding p_B, but c_A does not find a matching c_B

Case c) *The node matches but the father does not.* This case occurs when (1) of case (a) holds but (2) and (3) do not. See figures 8 and 9. Concept c_B is found but p_B is not. We verify two conditions (A) and (B):

(A) Most[17] of the facts of c_B should coincide (using *sim*) with those of c_A ; and

(B) Most of the children of c_A should coincide (by *sim*) with most[17] of the children of c_B.

(1) If the facts in (A) and the children in (B) coincide, the algorithm concludes with response c_B, although it did not find the $p_B \in O_B$ that corresponds to $p_A \in O_A$. Here *sv* = the fraction of facts and children of c_B matching with corresponding entities of c_A.
(2) If even fewer properties and children are *sim*ilar then response is `(probably` c_B`)` and the algorithm finishes. Here *sv* is computed like in (1)B.
(3) If neither properties nor children are *sim*ilar, response is `not_found` and the algorithm finishes. *sv* = 0.

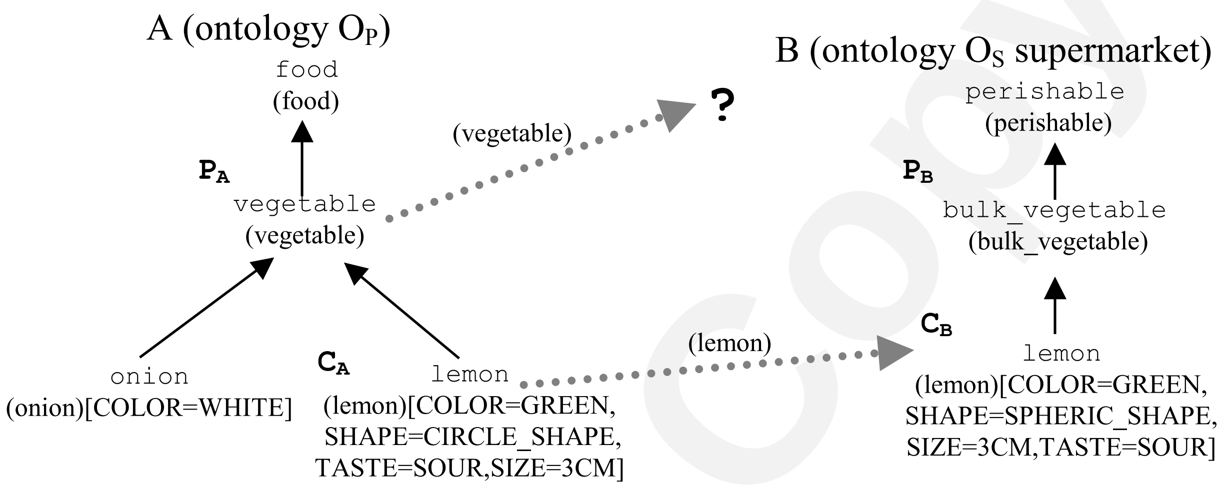


Figure 8. Case c). *Node matches but father does not*. Agent A (with ontology O_P of figure 1) wants to find a concept in B's ontology O_S corresponding to c_A = `lemon`. Words from c_A match with words from c_B, but there is no equivalence for words from p_A.

Figure 9 shows an example of case (c)(2). In this case we use *sim* to seek in B the most similar concept to $\texttt{lemon}_A$. Here concepts match but parents ($\texttt{vegetable}_A$, $\texttt{bulk\_vegetable}_B$) do not (words are different for each parent), therefore similarity of the properties are used (calling recursively to *sim*). *sv* = 0.75 because parents do not coincide, and the answer is "`probably lemon`."

Case d) If neither c_B nor p_B are found, the algorithm concludes returning the response `not_found`. Figures 1 and 1. *sv* = 0. c_A could not find a similar node in O_B. The agents may have different ontologies (they know about different subjects) or they may not share a common cmmunication language. See figures 10 and 11.

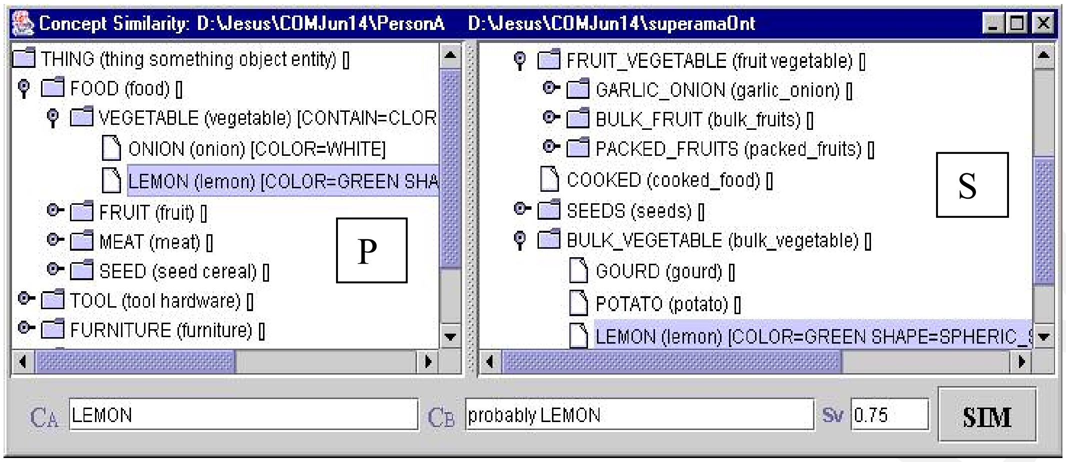


Figure 9. Case c) Unable to find in B the father of c_A , although c_B corresponding to c_A was found.

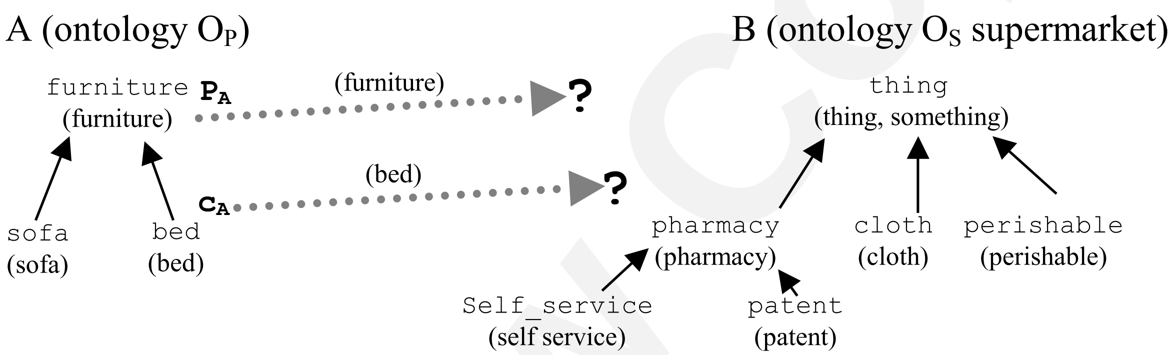


Figure 10. Case d) *Neither node nor father match.* Agent A wants to find the most similar concept in B to c_A = `sofa`. There are no words from c_A that map into words of c_B nor for p_A and p_B. Ontologies are those of figures 2 (O_S) and 3 (O_P).

Figure 11 shows the execution for case (d). Observe that A's ontology O_P fragment of interest is mainly about furniture while B's ontology O_S is mainly about groceries. There are some concepts in common, but not the involved concepts. sv = 0.

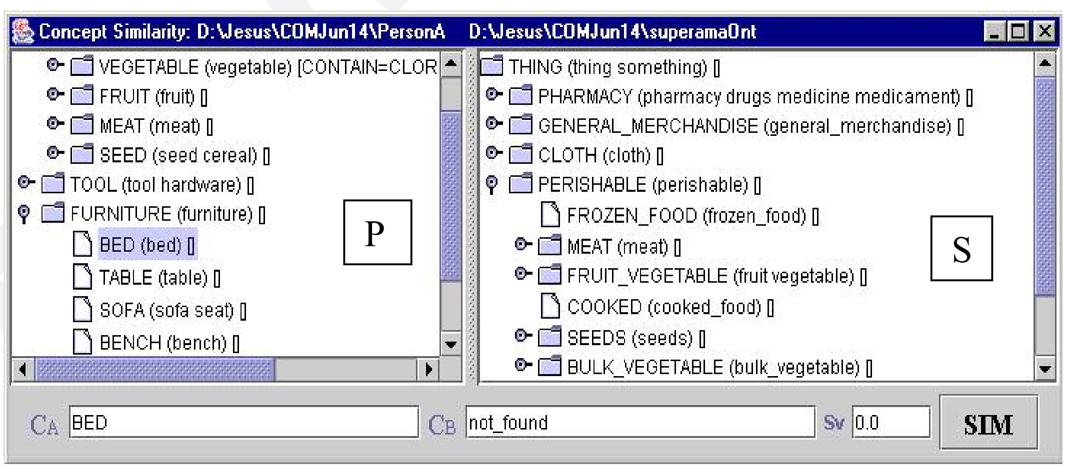


Figure 11. Case d) Neither c_A = `bed` nor p_A = `furniture` find suitable matches in ontology S.

If c_B is the concept most similar to c_A, it is not necessarily true that c_A is the concept most similar to c_B. *sim is not symmetric.* Example: Refer to Figure 12. Levachkine [16] digs deeper into this.

The function *sim* is only defined between a concept c_A in O_A and *the most similar concept* c_B in O_B; extensions sim' amd sim'' are in §3.1.2 below.

3.1.1 Who runs *sim*?

Who compares these two concepts, since they belong to different ontologies? That is, who runs *sim*? Either agent A or B can execute it, since *sim* compares *words,* not concepts. Nevertheless, when A runs *sim*, it needs the collaboration of B (and vice versa), which has to provide words to be used by *sim* (thus, by A). Also, even when A executes *sim* producing c_B as result, A can not "have" or "see" c_B: it is a pointer to the memory O_B, a meaningless pointer for A, such as the tokens of point 4 of §1.1. What A can see of c_B is: *(1)* the words which denote c_B, as well as (the words for) the nodes related to c_B; *(2)* corresponding words for the father, grandfather, sons... of c_B (and words for *their* relations); *(3) sv,* indicating how similar that elusive c_B is to its (very solid) c_A. In fact, A still has c_A as "the concept I have been thinking all along." When A runs *sim*, B can see, of course, c_B, but it can not "see" or "grasp" c_A. The most of what B can see of c_A is that "A wants to talk about something of which the closest I have is c_B".[19] B can sense from the words sent to it by A some differences between its solid c_B and the elusive c_A of A. More in §3.3.

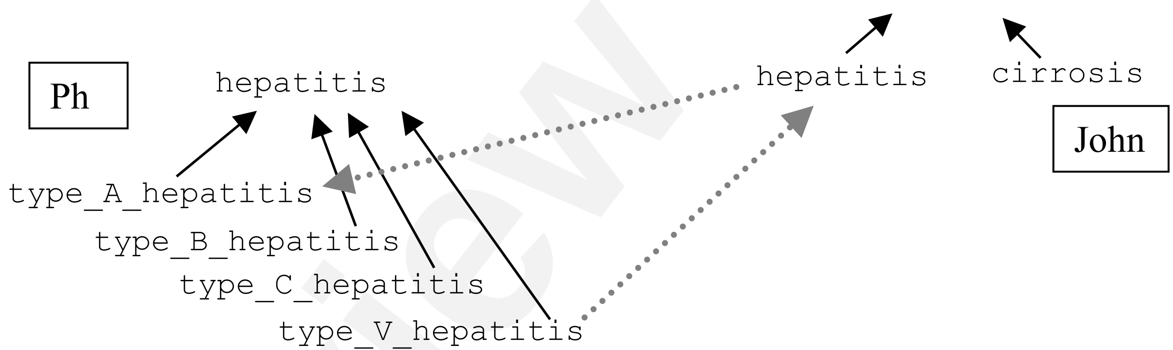


Figure 12. *sim is not symmetric.* Physician Ph knows four kinds of `hepatitis`, including the popular `type_A_hepatitis` and the rare `type_V_hepatitis` (viral), while John only knows `hepatitis`. The `type_V_hepatitis` of Ph (and all others) finds John's `hepatitis` as "the most similar concept John has," while John's `hepatitis` best maps into Ph's `type_A_hepatitis`. Ph knows more than John, so Ph can select a better target in his rich ontology for John's vague concept. *John can not make such selection.*

[19] It will not help if B is more cooperative. For instance, dumping all its ontology O_B into A's memory will not help A, who will still see a tangled tree of meaningless pointers. Well, not totally meaningless—some understandable words are attached to each node. Yes: A can (patiently) understand (untangle) O_B by comparing each of O_B nodes with its own O_A –that is, by using *sim*!

3.1.2 Generalizing sim

sim' (c_A, d_A) for two concepts belonging to the *same ontology,* is defined as the 1/(1+length of the path going from c_A to d_A in the O_A tree). ♦ sim' (c_A, d_A) ∈ [0, 1]. *sim' is symmetric*. Example: see figure 14.

3.1.2.1 Relation to confusion

In [16], the *confusion* conf(c_A, d_A) occurred by using c_A instead of d_A, is defined as the length of the descending[20] path from c_A to d_A. ♦ This definition holds for *hierarchies;* it is here extended to ontologies. If we had defined sim' (c_A, d_A) = (1/(1 + length of the descending path going from c_A to d_A in the O_A tree), we would have had sim' (c_A, d_A) = 1/1+conf(c_A, d_A). We prefer, for ontologies, the definition of sim' in §3.1.2, since it is symmetric, while conf is not. Example: for ontology D of figures 1 and 2, conf (`liquid_food, food`) = 0; the confusion when using `liquid_food` instead of `food` is 0, since liquid food *is* food. conf (`food, liquid_food`) = 1; when I want liquid food but I am given food, there is an error of 1 (a small error, you could think). For other concepts, we obtain the values in figure 13.

Confusion and similarity for concepts x and y belonging to the same ontology O_P of figure 1	conf (x, y); confusion in using x instead of y	conf (y, x); confusion in using y instead of x	sim' (x,y) = sim' (y,x); similarity between x and y
`x =plant, y =animal`	1	1	1/3
`x =plant, y =inanimate_thing`	1	2	1/4
`x =plant, y =cereal`	1	0	1/2
`x = fruit, y = tool`	2	3	1/6
`x = strawbery, y = furniture`	2	4	1/7
`x=thing, y=furniture`	2	0	1/3

Figure 13. Examples of confusion and similarity (sim') for two concepts of the same ontology O_P, figure 1. conf is not symmetric; sim' it is.

For similarity between any two objects of different ontologies, we have:

sim'' (c_A, d_B) when d_B is *not* the most similar concept in O_B to c_A, is found by making first s_1 = *sv* returned by *sim* (c_A) [this also finds c_B, the object in O_B *most* similar to c_A]; then, find s_2 = sim' (d_B, c_B). Now, sim'' (c_A, d_B) = $s_1 s_2$.

3.2 Degree of understanding

The value *sv* found in c_B=*sim*(c_A) in §3.1 can be thought of as the degree of understanding that agent B has about concept c_A. Each c_A that forces B to answer *sv*=0 indicates that B has no idea (no concept) about this c_A. We can add all these *sv*'s for every concept

[20] Going towards more specialized concepts. "Using a person from Houston when I want to use a Texan person, confusion is 0; using a Texan person when a Houston person is needed causes confusion=1; using a US person causes confusion=2."

$c_A \in O_A$ and find the degree of understanding that agent B has about O_A.[21] It is as if A asks B, for each concept $c_A \in O_A$, «do you understand what is c_A?» How much do you understand my c_A? At the end, A has a good idea of the understanding of B (with respect to O_A).

The *degree of understanding* of B about O_A, **du**(B, A) = {sum over all $c_A \in O_A$ of *sv* returned by *sim*(c_A)} / number of concepts in O_A. ♦ [20] Similarly, we can measure the degree of understanding of B about *some region* of O_A. **du** *is not symmetric.* In general, an agent understands some parts better than others. Notice that if O_A is a large (approaching in size the *total ontology* of §1.2), then the degree of understanding of B with respect to O_A approaches its *amount of knowledge* as given in §2.1.1.

du(B, A)≤ 1; in the regions where B knows more than A, **du**= 1. Example: for person P of figure 1 and supermarket S (Fig. 2), the degree of understanding of P about PS is **du**(P,S)= 29.75 / 150 =0.2. See figures 14, 1 and 15.

c_S	c_P= sim (c_S)	sv	c_S	c_P= sim (c_S)	sv
thing	thing	1.0	purple	purple	1.0
electrical	electrical_tool	1.0	shape	shape	1.0
socket	son_of_electrical_tool	0.5	circle	son_of_shape	0.5
multicon-tact	son_of_electrical_tool	0.5	spheric_shape	son_of_shape	0.5
meat	meat	1.0	square_shape	square_shape	1.0
pork	pork	1.0	taste	taste	1.0
beef	beef	1.0	sweet	sweet	1.0
turkey	turkey	1.0	sour	sour	1.0
veal	veal	1.0	size	size	1.0
white_onion	onion	1.0	3cm	3cm	1.0
lemon	probably lemon	0.75	matter_states	matter_states	1.0
relation	relation	1.0	liquid	liquid	1.0
color	color	1.0	suspension	suspension	1.0
white	white	1.0	gel	gel	1.0
yellow	yellow	1.0	solid	solid	1.0
green	green	1.0	gas	gas	1.0

Figure 14. We are trying to assess **du**(P, S), the degree of understanding that person P (figure 1) has about S's ontology (figure 2). Only rows where sv≠0 are shown. Columns 2 and 5 show the concept c_A most similar to each $c_S \in O_S$; columns 3 and 6 show the corresponding similarity value *sv*. The degree of understanding **du**(P,S) is computed by adding over every $c_S \in O_S$ the corresponding *sv* returned by c_P=sim(c_S) and dividing by 150 = number of concepts in O_S. Thus, **du**(P, S)= ($\Sigma_S sv$)/150= 29.75/150= 0.2. That is, P understands 20% of S. Results appear in Figure 15

[21] B does not know how many concepts there are in O_A, so it needs cooperation of A, for instance, when B asks A "give me the next concept in your ontology, please."

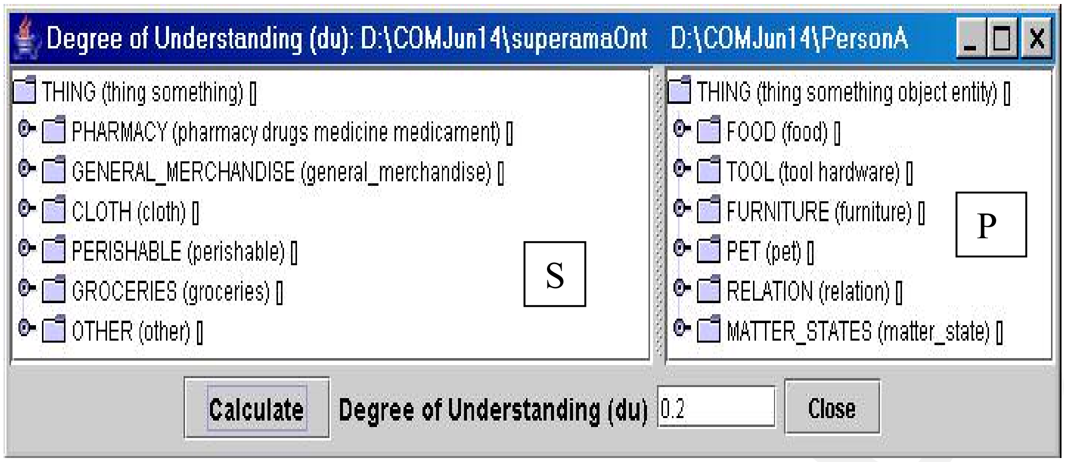


Figure 15. Computing **du**(P,S) as detailed in figure 14 shows that P understands 20% of S.

Since the size of S is larger than the size of P, probably du(S, P), the degree of understanding of S about P, will be larger than du(P, S). "The more I know, the more I understand." This is shown in figures 16 and 17.

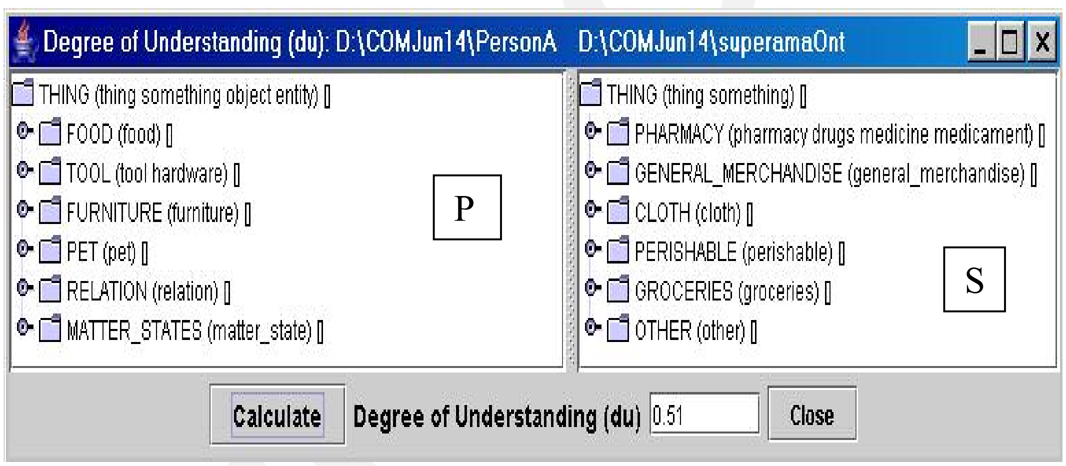


Figure 16. To compute du(S, P), the program adds all sv's produced (see figure 18) by each answer c_S=sim(c_P) when every concept of P is visited. Thus, du(S, P) = 0.51. S understands 51% of P.

17 of 24

c_P	c_S = sim (c_P)	sv	c_P	c_S = sim (c_P)	sv
thing	thing	1.0	orange_color	son_of_relation	0.5
onion	white_onion	1.0	shape	shape	1.0
lemon	probably lemon	0.75	circle_shape	son_of_shape	0.5
meat	meat	1.0	square_shape	square_shape	1.0
pork	pork	1.0	taste	taste	1.0
beef	beef	1.0	sweet	sweet	1.0
turkey	turkey	1.0	sour	sour	1.0
veal	veal	1.0	size	size	1.0
tool	hardware	1.0	3cm	3cm	1.0
wrench	son_of_hardware	0.5	covering	son_of_relation	0.5
electrical_tool	electrical	1.0	structure	son_of_relation	0.5
dish-washer	son_of_electrical	0.5	contain	son_of_relation	0.5
blender	son_of_electrical	0.5	family	son_of_relation	0.5
drill	son_of_electrical	0.5	use	son_of_relation	0.5
mixer	son_of_electrical	0.5	matter_states	matter_states	1.0
relation	relation	1.0	liquid	liquid	1.0
color	color	1.0	suspension	suspension	1.0
white	white	1.0	gel	gel	1.0
yellow	yellow	1.0	solid	solid	1.0
green	green	1.0	gas	gas	1.0
purple	Purple	1.0			

Figure 17. Now S is asked by P about each concept in O_P. Each answer of S (second and fifth columns; only c_S's with sv≠0 are shown) produces a similarity value (third and last columns). Adding these numbers gives 34.75. Dividing into | P |=68, the degree of understanding of S with respect to P is found to be 0.51.

We can also compute the "cross" knowledge among our two supermarket customers. Figures 18 and 19 show du(Q,P)=0.82. Figures 20 and 21 show that du(P, Q)=0.71. Thus, we can see that P knows more about Q than Q about P.

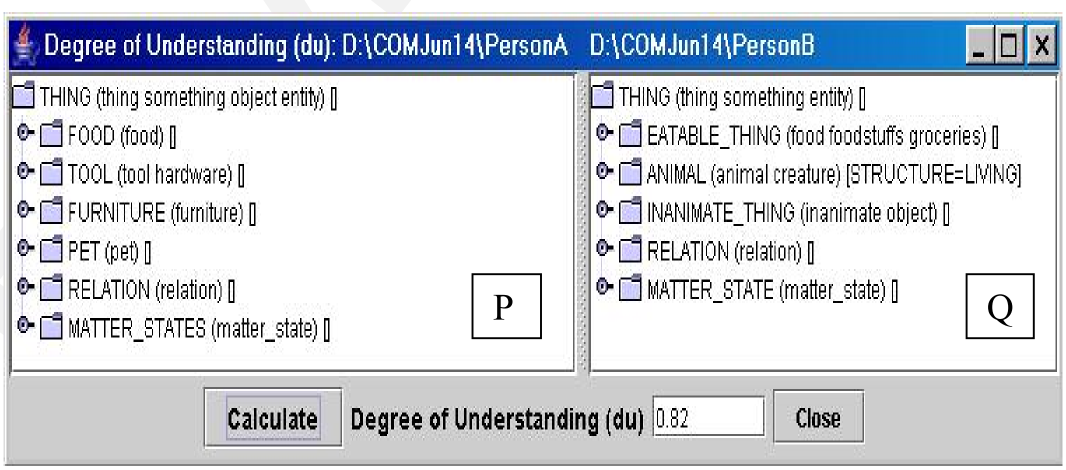


Figure 18. The degree of understanding of Q about P is du(Q, P) = 55.5 / 68 = 0.82. Details in figure 20.

C_P	C_Q = sim (c_P)	sv	C_P	C_Q = sim (c_P)	sv
thing	thing	1.0	green	green	1.0
food	eatable_thing	1.0	purple	purple	1.0
vegetable	vegetable	1.0	orange_color	orange_color	1.0
onion	onion	1.0	shape	shape	1.0
lemon	probably lemon	0.5	circle_shape	circle_shape	1.0
fruit	fruit	1.0	square_shape	square_shape	1.0
orange	probably orange	0.5	taste	son_of_relation	0.5
meat	son_of_eatable_th ing	0.5	size	size	1.0
seed	cereal	1.0	3cm	son_of_size	0.5
rice	rice	1.0	covering	covering	1.0
wheat	wheat	1.0	hair	hair	1.0
tool	tool	1.0	feather	feather	1.0
wrench	wrench	1.0	paint	son_of_covering	0.5
Electri-cal_tool	appliance	1.0	structure	structure	1.0
Dish-washer	dishwasher	1.0	living	living	1.0
blender	blender	1.0	man_made	man_made	1.0
drill	drill	1.0	contain	son_of_relation	0.5
mixer	son_of_appliance	0.5	family	family	1.0
furniture	furniture	1.0	feline	feline	1.0
bed	bed	1.0	canine	canine	1.0
table	table	1.0	use	use	1.0
sofa	sofa	1.0	cleaning	cleaning	1.0
bench	son_of_furniture	0.5	blend	blend	1.0
cat	cat	1.0	mix	mix	1.0
dog	dog	1.0	matter_states	matter_state	1.0
canary	canary	1.0	liquid	liquid	1.0
relation	relation	1.0	suspension	suspension	1.0
color	color	1.0	gel	gel	1.0
white	white	1.0	solid	solid	1.0
yellow	yellow	1.0	gas	gas	1.0

Figure 19. To compute du(Q, P), every $c_P \in P$ (first and fourth columns) is visited and the most similar $c_Q \in Q$ (second and fifth columns) is found, as well as its similarity value sv (third and sixth columns). Only rows where sv≠0 are shown. The sum of all these sv's (55.5) divided by the number of concepts in P (68) gives du(Q, P) = 0.81

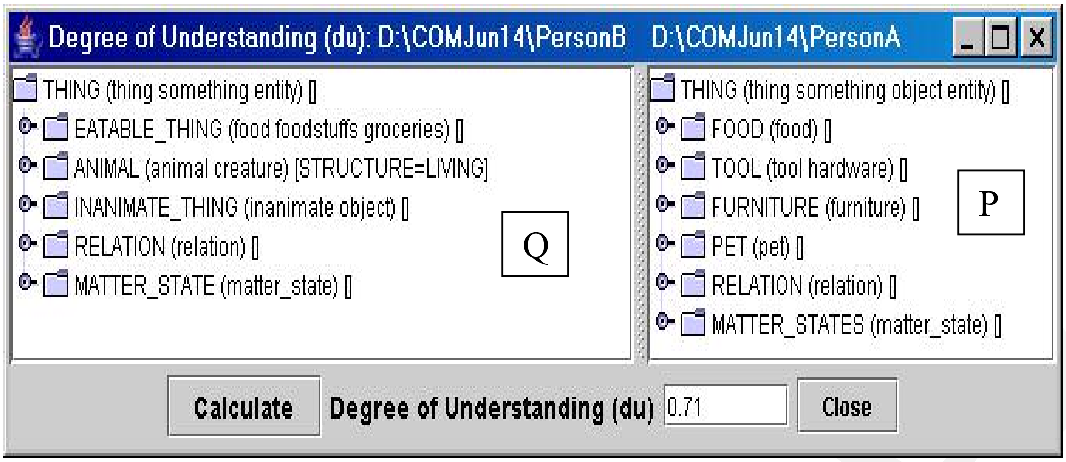
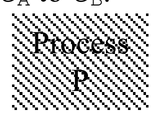


Figure 20. du(P, Q) = 54.1 / 75 = 0.71. P understands 72% of Q's knowledge. Details in figure 21.

3.3 Can I understand the source of our disagreement? Can I mend it?

§3.1.1 shows that A can not perceive or see c_B directly. Given $c_A \in O_A$ and its most similar concept $c_B \in O_B$, can A perceive in what way c_B differs from its c_A? After all, A knows from the value *sv* returned by sim(c_A), how imperfect is the matching of c_A to c_B.



The answer is *yes*, and it will be explained qualitatively. A can ask B about the facts involving c_B. It will receive the answers in words. Then, A can process them (through *sim*, perhaps) to see how c_A's facts differ from those received. It can do the same with the father_of (c_B), and with the sons_of (c_B). And so on. Some words received will refer to concepts of which A is not sure (it has no knowledge of them, or there is ambiguity), so that more processing (Process P is called again) *on these concepts* is needed. Also, occasionally A will receive from B relations involving c_B which A has as false for c_A.

3.3.1 An agent learning from another agent

A can collect all this information in a note N attached to c_A: "N is what B knows about my concept c_A, which differs from my knowledge". It is like having A compute a belief: "B believes or knows N about c_A, and N is not what I know about c_A." Or, perhaps, A can go ahead and decide to *internalize* N, that is, to "make it its knowledge": to *learn* N about c_A from O_B. For this to happen, A needs to incorporate the relations in N in its O_A,[22] and to *resolve the ambiguities and inconsistencies* coming from N (some of N's relations are known to A to be false; others make little sense to A). This has been solved for an *agent* teaching a *person*, but not yet for an agent teaching another agent. It can be done, perhaps, by using other *knowledge services* in the Web to referee disagreements between A and B and help A decide *who is wrong* about what (the "what" is already captured in N). We call this *ontology merging:* if A learns O_B from B, its new O_A will be its old O_A merged (as informally described here) with O_B. Cuevas' Ph.D. Thesis [2] is along these lines.

[22] This (to regard N as facts) is not difficult, once A decides to do it.

C_P	$C_Q = \mathrm{sim}(c_P)$	sv	C_P	$C_Q = \mathrm{sim}(c_P)$	sv
thing	thing	1.0	purple	purple	1.0
Eat-able_thing	food	1.0	green	green	1.0
plant	son_of_food	0.5	light_brown	son_of_color	0.5
vegetable	vegetable	1.0	orange_color	son_of_color	0.5
lettuce	son_of_vegetable	0.5	shape	shape	1.0
tomato	son_of_vegetable	0.5	circle_shape	circle_shape	1.0
onion	onion	1.0	square_shape	square_shape	1.0
potato	son_of_vegetable	0.5	size	size	1.0
lemon	lemon	1.0	small	son_of_size	0.5
cereal	seed	1.0	medium	son_of_size	0.5
rice	rice	1.0	structure	structure	1.0
wheat	wheat	1.0	living	living	1.0
canary	probably canary	0.34	man_made	man_made	1.0
cat	probably cat	0.67	covering	covering	1.0
dog	probably dog	0.67	hair	hair	1.0
tool	tool	1.0	feather	feather	1.0
manual	son_of_tool	0.5	family	family	1.0
appliance	electrical_tool	1.0	feline	feline	1.0
drill	drill	1.0	canine	canine	1.0
Electri-cal_saw	son_of_electrical_tool	0.5	use	use	1.0
dishwasher	dishwasher	1.0	cleaning	cleaning	1.0
blender	blender	1.0	blend	blend	1.0
furniture	furniture	1.0	mix	mix	1.0
bed	bed	1.0	matter_state	matter_states	1.0
table	table	1.0	liquid	liquid	1.0
sofa	sofa	1.0	suspension	suspension	1.0
chair	sofa	1.0	gel	gel	1.0
relation	relation	1.0	solid	solid	1.0
color	color	1.0	gas	gas	1.0
white	white	1.0	plasma	son_of_matter_states	0.5
yellow	yellow	1.0			

Fig. 21. To compute du(P, Q), concepts in Q (columns 2 y 5) most similar to each concept in P (columns 1 and 4), as well as the similarity value sv, are computed. Only rows with sv≠0 are shown.

3.4 Conclusions

Methods are given to allow interaction and understanding between agents with different ontologies, so that there is no need to agree first on a standard set of concept definitions. Given a concept and associated words, a procedure for finding the most similar concept in another ontology is shown, with examples, as well as a measure of the degree of understanding between two agents. It remains to test our methods with large, vastly different, or practical ontologies.

Work reported is a step towards free will interactions[4] among agents, instead of using canned interactions. Also, towards agents "strange to each other"[3] that try to interact and

21 of 24

make sense of their utterances, opposing the current trend where only agents written by the same person or group, or following the same data exchange standards, can interact.

Interaction through standards will still dominate the market for some time: it is easier to define and follow standards than to be “flexible, uncompromising and willing to try to understand new concepts.” A standard ontology in a discipline is a good thing, although it feels rigid and archaic after a while.[23] Nevertheless, knowledge can not be standardized, since each day more sprouts; standardization will always fall behind. It is preferable, I think, for me to be flexible and have general ways of trying to understand what you have to say (even new or unusual things), instead of forcing you to use a standard for concept-sharing with me (I already force you to use a shared communication language, perhaps a natural language, but that is unavoidable). Our work shows that *a standard ontology for concept-sharing is not needed;* it will be impossible in general, anyway.

3.4.1 Suggestions for further work

From text document to ontology. To help ontology merging (§3.3.1), write a converter that forms ontologies out of text documents.

Notation to describe any complex concept. How do you describe complex concepts, such as “clintonize” of §2.2? Idea: express it as relations composed by relations. See [19] for a first try.

Describe actions. Extend ontologies to make possible to describe a sequence of events [22].

Better notation for ontologies. • Tree notation (figure 1) is cumbersome, since only one “subset” relation is represented, and often a set S is partitioned in several partitions. Thus, a better notation could be:

```
person {partition sex (=M : male_person) (=F : female_person) }
         {partition age (≤20 : young_person)
                        (20< age ≤ 60 : adult_person)
                        (>60 : old_person) }
```

• Similarly, graphic notations make cumbersome to represent n-ary relations. • When characterizing the relations (as another branch of the ontology), you need to define types of partitioning relations (`sex`, `age`…), or whether the partition is a “natural” one, like partitioning `vertebrate` into `fish`, `bird`, `reptile`, `batrachians` and `mammal`.

Agent interaction. Establish necessary or sufficient conditions for agent interaction that do not have a communication agreement, as mentioned in §1.

3.5 Acknowledgments

Helpful discussions were held with Profs. Serguei Levachkine (CIC), Michael N. Huhns (U. of South Carolina), Alexander Gelbukh (CIC), and Hal Berghel (U. of Nevada Las Vegas). Work herein reported was partially supported by NSF-CONACYT Grant 32973-A and Project CGPI-IPN 2004442). Authors have SNI *National Scientist* Award (CONACYT)

[23] Compare the UNESCO Catalog of Sciences (which is 30-years obsolete in Computer Science) with the ACM Computing Classification System, which is 2-years obsolete.

3.6 References

1. V. Alexandrov, S. Levachkine and A. Guzman. “Data Dynamical Structures for Image Treatment with Applications to Digital Cartography”. Book in preparation.
2. A. Cuevas-Rasgado. *Automatic learning by an agent through ontology merging*. Ph. D. Thesis (in preparation; in Spanish), CIC-IPN.
3. J. Everett, D. Bobrow, R. Stolle, R. Crouch, V. de Paiva, C Condoravdi, M van den Berg, and L Polyani. Making ontologies work for resolving redundancies across documents. *Comm. ACM* **45 (**2) (2002) 55-60.
4. K. Forbus, B. Falkenhainer and D. Gentner. The structure mapping engine: algorithms and examples. *Artificial Intelligence* **41** (1) (1989) 1-63.
5. A. Gelbukh, G. Sidorov and A. Guzman-Arenas. Use of a weighted document topic hierarchy for document classification. *Text, Speech, Dialogue* (Pilsen, Chech Republic, 1999)133-138.
6. A. Gelbukh, G. Sidorov, and A. Guzman-Arenas. Document comparison with a weighted topic hierarchy. *DEXA-99, 10th International Conference on Database and Expert System applications,* Workshop on Document Analysis and Understanding for Document Databases (Florence, Italy, 1999) 566-570.
7. T. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing, in: Nicola Guarino and Roberto Poli (eds.), *Formal Ontology in Conceptual Analysis and Knowledge Representation* (Kluwer Academic Publishers 1993).
8. A. Guzman-Arenas. Finding the main themes in a Spanish document. *Journal Expert Systems with Applications,* **14** (1/2) (1998) 139-148.
9. A. Guzman-Arenas, J Olivares, A. Demetrio and C. Dominguez. Interaction of purposeful agents that use different ontologies. In: *Lecture Notes in Artificial Intelligence* **1793** (Springer 2000) 557-573.
10. A. Guzman-Arenas, C. Dominguez and J. Olivares. Reacting to unexpected events and communicating in spite of mixed ontologies In: *Lecture Notes in Artificial Intelligence* **2313** (Springer Heidelberg 2002) 377-386.
11. C. Holsapple and K. Joshi. A collaborative approach to ontology design. *Comm. ACM* **45 (**2) (2002) 42-47.
12. M. N. Huhns, M. P. Singh and T. Ksiezyk. Global Information Management Via Local Autonomous Agents. In: M. N. Huhns and M. P. Singh (eds.): *Readings in Agents* (Morgan Kauffmann 1997).
13. H. Kim. (2002) Predicting how ontologies for the semantic web will evolve. *Comm. ACM* **45** (2) (2002) 48-54.
14. F. Kon, F. Costa, G. Blair and R. H. Campbell. The case for reflective middleware. *Comm. ACM* **45** (6) (2002) 33-38.
15. D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt and M. Shepherd. Cyc: Toward Programs with Common Sense, *Comm. of the ACM* **33** (9) (1990) 30-49.
16. S. Levachkine and A. Guzman-Arenas. Hierarchies Measuring Qualitative Variables. In: *Lecture Notes in Computer Science* **2945** (Springer-Verlag 2004) 262-274.
17. S. Levachkine and A. Guzman-Arenas. Hierarchy as a new data type for qualitative variables. Submitted to *Data and Knowledge Engineering.*

23 of 24

18. M. Montes-y-Gomez, A. Lopez-Lopez and A. Gelbukh. Information Retrieval with Conceptual Graph Matching. In: *Lecture Notes in Computer Science* **1873** (Springer-Verlag 2000) 312-321.
19. J. Olivares. *An Interaction Model among Purposeful Agents, Mixed Ontologies and Unexpected Events.* Ph. D. Thesis, CIC-IPN, 2002. (In Spanish) Available on line at http://www.jesusolivares.com/interaction/publica
20. Jesus M. Olivares-Ceja, Adolfo Guzman-Arenas. Concept similarity measures the understanding between two agents. Accepted in *NLDB 04.* To appear in *Lecture Notes in Computer Science* (Springer 2004).
21. Jan Kupper, Horacio Saggion, *et al.* Multi-source in formation extraction and merging. *Proc. IJCAI 03,* 409-414