The Semantics of Confusion in Hierarchies: Theory and Practice

Serguei Levachkine[1], Adolfo Guzmán-Arenas[1,2], and Victor Polo de Gyves[2]

[1]Centre for Computing Research (CIC) - National Polytechnic Institute (IPN)
UPALMZ, CIC Building, 07738, Mexico City, MEXICO
[2]SoftwarePro International
sergei@cic.ipn.mx, a.guzman@acm.org, degyves@gmail.com

Abstract. Conceptual knowledge embedded in symbolic values (such as software, computer systems, Asia, Japan) is formally represented in a tree of sets, also known as a *hierarchy*. Similarities (or rather, errors) among values belonging to a hierarchy are gauged by a function: *conf (r, s)* measures the degree of confusion when (symbolic) value *r* is used instead of (the correct, or intended) value *s*. Intuitively (by people), as well as formally (by *conf*), the error in using nurse instead of physician is smaller than the error if hospital or avocado were used. In addition to being easy to use, *conf* allows the definition of other functions: *Identical*, when two symbolic values are identical. *Substitute*, when a symbolic value may replace another, with confusion 0. *Very similar*. *Similar*. *Somewhat similar*. *Equality up to a given confusion*: $r =_{\varepsilon} s$ (*r* is equal to *s* up to confusion ε). More over, it is possible to extend a predicate P(o) over an object *o*, which is *true* if *o* fulfils P, to "P_{ε} (o)", read "P holds for *o* within confusion ε", which is *true* if *o* fulfils P up to confusion ε. Once we have allowed *confusion* to enter predicates, we can extend a relational database, granting it the capability to retrieve objects within a given confusion. Thus, the theory so far exposed acquires practical dimensions. This is done by introducing *extensions* to SQL, so that a user can express queries involving confusion. Later, these *extensions* are removed by a parser that takes an extended SQL sentence and produces a "pure" SQL sentence, but where confusion is handled satisfactorily, through tables in memory. Our implementation shows that these conceptual structures and formal representations can support human communication and use; in fact, we give (simple) examples of inexact retrieval.

1 Introduction

What wearing apparel do we wear for rainy days? *Raincoat* is a correct answer; *umbrella* is a close miss; *belt* a fair error, and *typewriter* a gross error. What is closer to an *apple,* a *pear* or a *caterpillar?* Can we measure these errors and similarities? How related or close are these words? Some preliminary definitions follow.

Element set. A set[1] E whose elements are explicitly defined. ♦[2] *Example*: {*red, blue, white, black, pale*}.
Ordered set. An element set whose values are ordered by a < ("less than") relation. ♦ *Example*: {*very_cold, cold, warm, hot, very_hot*}.

Covering. K is a covering for set E if K is a set of subsets $s_i \subset E$, such that $\cup\, s_i = E$. ♦ Every element of E is in some subset $s_i \in K$. If K is not a covering of E, we can make it so by adding a new s_j to it, named "others", that contains all other elements of E that do not belong to any of the previous s_i.

Exclusive set. K is an exclusive set if $s_i \cap s_j = \emptyset$, for every $s_i, s_j \in K$. ♦ Its elements are mutually exclusive. If K is not an exclusive set, we can make it so by replacing every two overlapping $s_i, s_j \in K$ with three: $s_i - s_j$, $s_j - s_i$, and $s_i \cap s_j$.
Partition. P is a partition of set E if it is both a covering for E and an exclusive set.
Qualitative variable. A single-valued variable that takes symbolic values. ♦ Its value cannot be a set.[3] By symbolic we mean qualitative, as opposed to numeric, vector or quantitative variables.

A symbolic value v **represents** a set E, written $v \propto E$, if v can be considered a name or a depiction of E. ♦ *Example*: *Pale* ∝ {*white, yellow, orange, beige*}.

1.1 Hierarchy

For an element set E, a **hierarchy** H of E is another element set where each element e_i is a symbolic value that represents either a single element of E or a partition, and $\cup_i \{r_i \mid e_i \propto r_i\} = E$ (The union of all sets represented by the e_i is E). ♦ *Example* (Hierarchy H_1): for E = {*Canada, USA, Mexico, Cuba, Puerto_Rico, Jamaica, Guatemala, Honduras, Costa_Rica*}={a, b, c, d, e, f, g, h, i}, a hierarchy H_1 is {*North_America, Caribbean_Island, Central_America*}={H_1^1, H_1^2, H_1^3}, where *North_America* ∝ {*Canada, USA, Mexico*}; *Caribbean_Island* ∝ {*English_Speaking_Island, Spanish_Speaking_Island*}={H_1^{21}, H_1^{22}}; *English_Speaking_Island* ∝ {*Jamaica*}; *Spanish_Speaking_Island* ∝ {*Cuba, Puerto_Rico*}; *Central_America* ∝ {*Guatemala, Honduras, Costa_Rica*}.

Hierarchies make it easier to compare qualitative values belonging to the same hierarchy (§3), and even to different hierarchies (procedure *sim* in [3]).

A **hierarchical variable** is a qualitative variable whose values belong to a hierarchy (The data type of a hierarchical variable is hierarchy). ♦ *Example*: *place_of_origin* that takes values from H_1. Note: hierarchical variables are single-valued.

[1] Perhaps infinite, perhaps empty.

[2] The symbol ♦ means: end of definition.

[3] Variable, attribute and property are used interchangeably. An object may have an attribute (Ex: weight) while others do not: the weight of blue *does not make sense*, as opposed to saying that the weight of blue *is unknown* or not given. A variable (*color*, *height*) describes an aspect of an object; its value (*blue*, *2 Kg*) is such description or measurement.

Thus, a value for *place_of_origin* can be *North_America* or *Mexico,* but not {*Canada, USA, Mexico*}, although *North_America* ∝ {*Canada, USA, Mexico*}.

1.2 Notation

The sets represented by each element of a hierarchy form a tree under the relation subset. *Example*: for H_1, such tree is given in Figure 1.

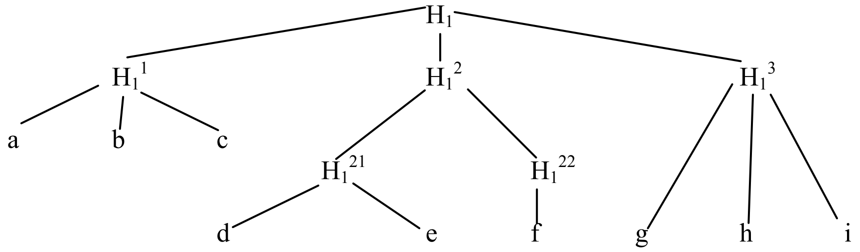

Fig. 1. The tree induced by hierarchy H_1.

We will also write a hierarchy such as H_1 thus: {*North_America* ∝ {*Canada USA Mexico*} *Caribbean Island* ∝ {*Spanish_Speaking_Island* ∝ {*Cuba Puerto_Rico*} *English_Speaking_Island* ∝{*Jamaica*} } *Central_America* ∝ {*Guatemala Honduras Costa_Rica*} }.

father_of (v). In a tree representing a hierarchy (such as H_1), the father_of a node is the node from which it hangs. ♦ Similarly, the **sons_of** (v) are the values hanging from v. The nodes with the same father are **siblings**. ♦ Similarly, **grand_father_of**, **brothers_of, aunt**, **ascendants**, **descendants**... are defined, when they exist. ♦ The **root** is the node that has no father.♦

2 Previous Work

Clasitex [1] finds the themes of an article written in Spanish or English, performing a task equivalent to disambiguation of a word into its different senses. It uses the concept tree, and a word (words lie outside the context tree) *suggests the topic of* one or more concepts in the tree. A document that talks about Cervantes, horses and corruption will be classified (indexed) in these three nodes in the tree. In [2], each agent possesses its own ontology of concepts, but must map these into natural language words for communication [3]. Thus LIA, a language for agent interaction [2], has an ontology comparator COM, that maps a concept from one ontology into the closest corresponding concept of another ontology. COM achieves communication without need of a common or *standard ontology*; it is used in *sim* of §3.4.

A datum makes sense only within a context. Intuitively, we know that “computer” is closer to “office” than to “ocean” or to “dog.” A “cat” is closer to “dog” than to “bus station.” “Burning” is closer to “hot” than to “icy.” How can we measure these similarities?

A hierarchy describes the structure of qualitative values in a set S. A **(simple, normal) hierarchy** is a tree with root S and if a node has children, these form a partition of the father [4]. A simple hierarchy describes a hierarchy where S is a set (thus its elements are not repeated, not ordered). For example, live being{animal{mammal, fish, reptile, other animal}, plant{tree, other plant}}. In a **percentage hierarchy** [6], the size of each set is known[4]. For instance, AmericanContinent(640M){North America(430M) {USA(300M), Canada(30M), Mexico(100M)} Central America (10M), South America(200M)}. In an **ordered hierarchy** [5], the nodes of some partitions obey an ordering relation: object{tiny, small, medium, large}* [5]. Finally, a **mixed hierarchy** combines the three former types. Other works related to retrieval of approximate answers are referenced in [7].

For these four types of hierarchies we define *conf(r, s)* as the confusion or error in using value r instead of s, the intended or correct value. These definitions agree with the human sense of estimation in closeness for several wrong but approximate answers to a given question; each is applicable to particular endeavors.

Then, we define an enriched SQL syntax that deals with approximate queries on elements in a database holding qualitative values hierarchically structured. This enriched SQL uses precision-controlled predicates. Next, we explain how the extension (to precision-controlled retrieval) of *any* database is possible. Finally, we give some examples.

3 Properties and Functions on Hierarchies

I ask for a *European car*, and I get a *German car*. Is there an error? Now, I ask for a *German car*, and a *European car* comes. Can we measure this error? Can we systematize or organize these values? Hierarchies of symbolic values allow measuring the similarity between these values, and the error when one is used instead of another.

3.1 Confusion in using r instead of s, for simple hierarchies

If r, s $\in$ H, then the **confusion** in using r instead of s, written conf(r, s), is:

- conf (r, r) = conf (r, s) = 0, where s is any ascendant of r; **(1)**
- conf (r, s) = 1 + conf (r, father_of(s)) . ♦ **(2)**

[4] Notation: the size of each set is written in parenthesis after the set. Here we write number of inhabitants.

[5] Notation: an * is placed at the end of the partition, to signify that it is an *ordered* partition.

To measure conf, count the *descending* links from r to s, the replaced value. conf is not a *distance*, nor *ultradistance*. To differentiate, we prefer to use **confusion** instead of other linguistic terms like relatedness or closeness.

Example (Hierarchy H_2): conf(r, s) for H_2 of Figure 2 is given in Table 1:

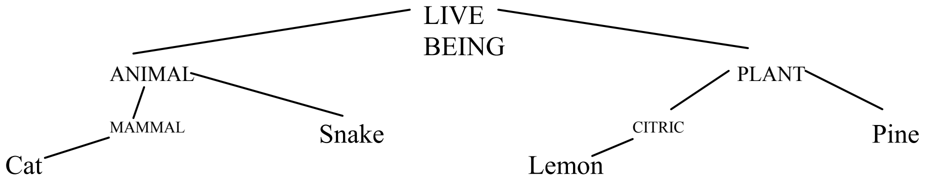

Fig. 2. A hierarchy H_2 of live beings.

Table 1. conf(r, s): Confusion in using r instead of s for the live beings of H_2.

→ ***s***

↓ ***r***

Conf	Live b.	Animal	Plant	Mam.	Snake	Citric	Pine	Cat	Lemon
Live b.	0	1	1	2	2	2	2	3	3
Animal	0	0	1	1	1	2	2	2	3
Plant	0	1	0	2	2	1	1	3	2
Mam.	0	0	1	0	1	2	2	1	3
Snake	0	0	1	1	0	2	2	2	3
Citric	0	1	0	2	2	0	1	3	1
Pine	0	1	0	2	2	1	0	3	2
Cat	0	0	1	0	1	2	2	0	3
Lemon	0	1	0	2	2	0	1	3	0

The confusion thus introduced *resembles reality* and *catches the hierarchy semantics*. For example, conf (*animal, live_being*) = 0: if they ask you for a live being and you give them an animal, the error of using animal instead of live being is 0, since all animals are live beings. Giving a live being when asked for an animal has error 1; conf (*live_being, animal*) = 1. The confusion among two brothers (say, dog and cat) is 1; using a son instead of the father produces conf=0; using the father instead of the son makes conf = 1. conf is *not* a symmetric property. Using *general things* (see row 'live being') instead of *specific things* produces *high errors*. Using *specific things* (see row 'lemon') instead of *general things* produces *low errors*. The table's lower triangular half has *smaller errors* than its upper triangular half[6].

[6] These triangular parts would result to be equal for a distance. Thus, distance represents a context-looseness measure in this case.

3.1.1 Confusion in using r instead of s, for ordered hierarchies

For hierarchies formed by sets that are lists (ordered sets; example Temp={icy, cold, normal, warm, hot, burning}), the **confusion** in using r instead of s, conf’’ (r, s), is defined as:

- conf’’ (r, r) = conf (r, any ascendant of r) = 0;
- If r and s are distinct brothers, conf’’ (r, s) = 1 if the father is not an ordered set; else, conf’’ (r, s) = the relative distance from r to s = the number of steps needed to jump from r to s in the ordering, divided by the cardinality-1 of the father;

  (3)
- conf’’ (r, s) = 1 + conf’’(r, father_of(s)). ♦

This is like conf for *hierarchies formed by sets*, except that there the error between two brothers is 1, and here it is a number ≤ 1. *Example*: in the list *Temp*, conf’’ (*icy, cold*) = 1/5, while conf’’ (*icy, burning*) = 5/5.

3.1.2 Confusion in using r instead of s, for percentage hierarchies

Now consider a hierarchy H (of an element set E) but composed of bags (unordered collection where repetitions are allowed) instead of sets.

For bags, the **similarity** in using r instead of s, sim^b (r, s), is:

- sim^b (r, r) = sim^b (r, any ascendant_of (r)) = 1;
- if r is ascendant of s, sim^b(r, s)= number of elements of $S \cap r \cap s$ / number of elements of $S \cap r$ = relative popularity of s in r. ♦[7] **(4)**

Example: If *baseball_player* = {*pitcher catcher base_player* $\propto$ {*baseman baseman baseman*} *field_player* $\propto$ {*fielder fielder fielder*} *shortstop*} then (a) conf’ (*fielder, baseball_player*) = 1 – sim^b (*fielder, baseball_player*) = 0; (b) conf’ (*baseball_player, fielder*) = 1 – 1/3 = 2/3; (c) conf’ (*baseball_player, left_fielder*) = 8/9 (a *left_fielder* is one of those three fielders); (d) conf’ (*base_player, fielder*) = 2/3.

3.1.3 Confusion in using r instead of s, for mixed hierarchies

To compute *sim(r, s)* in a mixed hierarchy, proceed as follows:

- apply rule (1) to the *ascending* path from *r* to *s*;
- in the descending path, use rule (3) instead of rule (2), if p is an ordered set[8]; or use rule (4) instead of (2), when sizes of p and q are known. ♦ That is, use (4) instead of (2) for percentage hierarchies.

This definition is consistent with and reduces to previous definitions for simple, ordered, and percentage hierarchies.

[7] Number of elements of S that are in r and that also occur in s / number of elements of S that are also in r = relative popularity or percentage of s in r.

[8] Here, p and q are two consecutive elements in the path from r to s, where q immediately follows p. That is, r → …p→q… →s.

The rest of the paper will derive results for conf; those for conf' and conf'' can be similarly derived.

3.2 The set of values that are equal to another, up to a given confusion

A **value u is equal to value v, within a given confusion** ε, written u $=_\varepsilon$ v, iff conf(u, v) ≤ ε (It means that value u can be used instead of v, within error ε). ♦

Example: If v = *lemon* (Figure 2), then (a) the set of values equal to v with confusion 0 is {*lemon*}; (b) the set of values equal to v with confusion 1 is {*citric lemon*}; (c) the set of values equal to v with confusion 2 is {*plant citric pine lemon*}. Notice that $=_\varepsilon$ is neither *symmetric* nor *transitive*.

3.2.1 Queries

Objects possessing several properties (or variables), some of them perhaps hierarchical variables, can best be stored as rows of a table in a relational database. We now extend the notion of queries to tables with hierarchical variables,[9] by defining the set S of objects that satisfy predicate P within a given confusion ε.

P holds for object o with confusion ε, or P holds for o within ε, iff
(1) if P is formed by non-hierarchical variables, iff P is true for o;
(2) for pr a hierarchical variable and P of the form (pr = c), iff for value v of property pr in object o, v $=_\varepsilon$ c (if the value v of the object can be used instead of c with confusion ε);
(3) if P is of the form P1 ∨ P2, iff P1 holds for o within ε or P2 holds for o within ε;
(4) if P is of the form P1 ∧ P2, iff P1 holds for o within ε and P2 holds for o within ε;
(5) if P is of the form ¬P1, iff P1 does not hold for o within ε. ♦

Example 1 (refer to hierarchies H_1 and H_2 above): Let the *predicates* be: P = (*lives_in* = *USA*) ∨ (*pet* = *cat*), Q = (*lives_in* = *USA*) ∧ (*pet* = *cat*), R = ¬ (*lives_in* = *Spanish_Speaking_Island*); and the *objects* be (Ann (*lives_in USA*) (*pet snake*)), (Bill (*lives_in English_Speaking_Island*) (*pet citric*)), (Fred (*lives_in USA*) (*pet cat*)), (Tom (*lives_in Mexico*) (*pet cat*)), (Sam (*lives_in Cuba*) (*pet pine*)). Then we have the following results (Table 2):

Table 2. How the predicates P, Q and R of example 1 hold for several objects.

	P holds within ε for:	**Q holds within ε for:**	**R holds within ε for:**
ε = 0	Ann, Fred, Tom	Fred	Ann, Bill, Fred, Tom
ε = 1	Ann, Fred, Tom	Fred, Tom	Ann, Fred, Tom
ε = 2	Ann, Fred, Tom, Sam	Ann, Fred, Tom	Nobody

[9] For non-hierarchical variables, a match in value means conf = 0; a mismatch means conf = ∞

3.2.2 The smallest ε for which P(o) is true

How close is Tom to be like Ann in Example 1? Ann lives in the USA and her pet is a snake, while Tom lives in Mexico and his pet is a cat. When we apply S = (*lives_in* = *USA*) $\wedge$ (*pet* = *snake*) to Tom, we see that S starts holding for $\varepsilon=1$. The answer to "How close is Tom to Ann?" is 1. Notice that this is not a *symmetric* property.

Ann is close to Tom starting from $\varepsilon=2$; that is, (*lives_in* = *Mexico*) $\wedge$ (*pet* = *cat*) does not hold for Ann at $\varepsilon=1$, but it starts holding for her at $\varepsilon=2$. This defines the "*closeness to.*"

Object *o* **ε-fulfills** predicate P at threshold ε, if ε is the smallest number for which P holds for *o* within ε. ♦ Such smallest ε is the **closeness** of *o* to P. ♦

Closeness is an integer number defined between an object and a predicate. The closer is ε to 0, the "tighter" P holds. Compare with the *membership function* for fuzzy sets.

3.3 Confusion between variables (not values) that form a hierarchy

What could be the error in "Sue directed the thesis of Fred", if all we know is "Sue was in the examination committee of Fred"? Up to now, the *values* of a hierarchical variable form a hierarchy (Cf. §1.1). Now, consider the case where the *variables* (or relations) form a hierarchy. For instance, relative and brother, in a universe of kinship relations E = {*sister, aunt…*}. Consider *hierarchies* H_3 *and* H_4: (H_3) *relative* $\propto$ {*close_relative* $\propto$ {*father mother son daughter brother sister*} *mid_relative* $\propto$ {*aunt uncle niece cousin*} *far_relative* $\propto$ {*grandfather grandmother grandson granddaughter grandaunt granduncle grandcousin grandniece*} }, (H_4) *player* $\propto$ {*socker_player* $\propto$ {*John Ed*} *basketball_player* $\propto$ {*Susan Fred*} }.

In hierarchy H_3, conf (*son, relative*) = 0; conf (*relative, son*) = 2. We know that, for object (Kim (*close_relative Ed*) (*pet cat*)), the predicate V = (*close_relative Ed*) holds with confusion 0. It is reasonable to assume that W = (*son Ed*) holds for Kim with confusion 1[10]; that X = (*relative Ed*) holds for Kim with confusion 0. Moreover, since Ed is a member of hierarchy H_4, it is reasonable to assume that for object (Carl (*close_relative socker_player*) (*pet pine*)) the predicate V holds with confusion 1, X holds with confusion 1 and W holds with confusion 1+1 = 2. Thus, we can extend the definition to variables that are members of a hierarchy, by adding another bullet to the definition of §3.2.1, thus:

If P is of the form (var = c), for var a variable member of a hierarchy, iff ∃ variable var_2 for which (var_2=c) holds for o within ε – conf (var, var_2), where var_2 also belongs to the hierarchy of var.♦

The confusion of the variables *adds* to the confusion of the values. *Example*: For (Burt (*relative basketball_player*) (*pet cat*)), V holds with confusion 1+2=3, W with confusion 2+2=4, and X with confusion 0+2=2.

[10] We are looking for a person that is a son of Ed, and we find Kim, a close relative of Ed.

3.4 Similarity for values in different hierarchies and in different ontologies

When v_l belongs to a hierarchy H_l and v_2 to another hierarchy H_2, both with the same element set E, it is best to construct an *ontology* O_U from E, and then to use it to measure the similarity sim'(v_1, v_2), as follows: sim' (c_U, d_U) for two concepts belonging to the *same ontology* O_U, is defined as the 1/(1 + length of the path going from c_U to d_U in the O_U tree). ♦ sim' is defined for *concepts,* not for symbolic values.

Also, for concepts c_A, d_B belonging to *different ontologies* O_A, O_B, we define: sim'' (c_A, d_B) when d_B is *not* the most similar concept in O_B to $c_A \in O_A$, is equal to $s_1 s_2$, where s_l = sim (c_A, O_A, O_B) [sim gives the similarity between c_A and its most similar concept c_B in O_B; sim also finds c_B], and s_2 = sim' (c_B, d_B). ♦

3.5 Object's similarity and accumulated confusion

Let us consider the following three hierarchies with the idea to introduce more new concepts such as identical, substitute, similar, etc. and accumulated confusion.

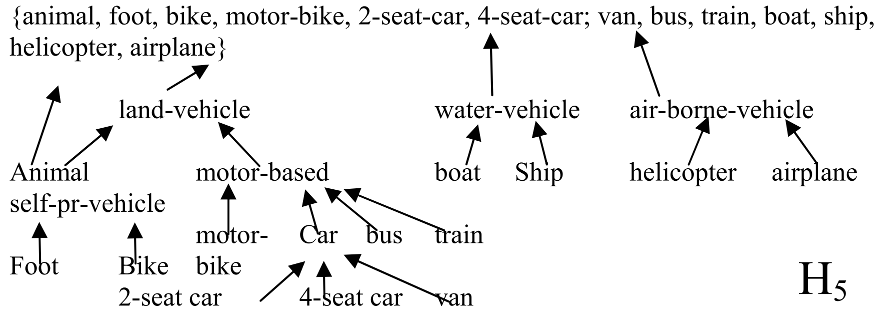

Fig. 3. A hierarchy H_5 of transportation vehicles. Some qualitative values, like air-borne-vehicle, represent sets: {helicopter, airplane} in our example

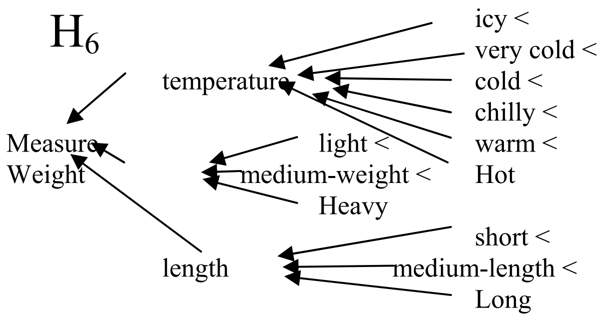

Fig. 4. A hierarchy having some ordered sets: (short < medium-length < long), (light < medium-weight < heavy), (icy < very cold < cold < chilly < warm < hot)

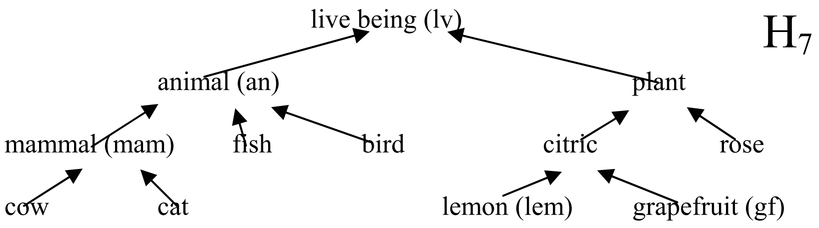

Fig. 5. A hierarchy H$_7$ of living creatures.

3.5.1 Identical, very similar, somewhat similar objects.

Objects are entities described by a set of (property, value) pairs, which in our notation we refer to as (variable, value) pairs. They are also called (relationship, attribute) pairs in databases. An object *o* with k (variable, value) pairs is written as (o (v_1 a_1) (v_2 a_2)... (v_k a_k)). Example: (Bob (*travels-by* boat) (*owns* bird) (*weighs* heavy))

We want to estimate the error in using object *o'* instead of object *o*. For an object *o* with k (perhaps hierarchical) variables v_1, v_2,.., v_k and values a_1, a_2, ..., a_k, we say about another object *o'* with same variables v_1...v_k but with values a_1', a_2',... a_k', the following statements:

- *o'* is **identical** to *o* if $a_i' = a_i$ for all $1 \le i \le k$. All corresponding values are identical. ♦ If all we know about *o* and *o'* are their values on variables v_1,...v_k, and both objects have these values pairwise identical, then we can say that "for all we know," *o* and *o'* are the same.
- *o'* is **a substitute** for *o* if conf (a_i', a_i) = 0 for all $1 \le i \le k$. ♦ There is no confusion between a value of an attribute of *o'* and the corresponding value for *o*. We can use *o'* instead of the (correct, intended) *o* with confusion 0.
- o' is **very similar** to o if Σ conf (a_i', a_i) = 1. ♦ The sum of all confusions is 1.
- o' is **similar** to o if Σ conf (a_i', a_i) = 2. ♦

- *o'* is **somewhat similar** to *o* if Σ conf (a_i', a_i) = 3. ♦
- In general, *o'* is **similar$_n$** to o if Σ conf (a_i', a_i) = n. ♦

These relations are not symmetric.

Example 2 (We use hierarchies H_5, H_6 and H_7). Consider the objects

(Ann	(*travels-by* land-vehicle)	(*owns* animal)	(*weighs* weight))
(Bob	(*travels-by* boat)	(*owns* bird)	(*weighs* heavy))
(Ed	(*travels-by* water-vehicle)	(*owns* plant)	(*weighs* medium-weight))
(John	(*travels-by* car)	(*owns* cow)	(*weighs* light)).

Then Ann is similar$_4$ to Bob; Bob is very similar to Ann; Ann is somewhat similar to Ed; Ed is similar$_{3.5}$ to Bob;[11] Bob is similar$_6$ to John, etc. See Table 3.

Table 3. Relations between objects of Example 2. This table gives the relation obtained when using object o' (running down the table) instead of object o' (running across the table)

	Ann	Bob	Ed	John
Ann	Identical	similar$_4$	somewhat similar	Similar$_5$
Bob	very similar	identical	Very similar	Similar$_6$
Ed	Similar	similar$_{3.5}$	Identical	Similar$_6$
John	substitute	similar$_4$	Similar$_{2.5}$	identical

3.5.2 Accumulated confusion

For compound predicates, a tighter control of the error or confusion is possible if we require that the accumulated error does not exceed a threshold ε. This is accomplished by the following definition.

P **holds for object o with accumulated confusion** ε, written *P^ε holds for o*, iff

- If P^ε is formed by non-hierarchical variables, iff P is true for o.
- For *pr* a hierarchical variable and P^ε of the form (*pr* c), iff for value v of property *pr* in object o, v $=_\varepsilon$ c. [if the value v can be used instead of c with confusion ε]
- If P^ε is of the form P1 ∨ P2, iff $P1^\varepsilon$ holds for o or $P2^\varepsilon$ holds for o.
- If P^ε is of the form P1 ∧ P2, iff there exist confusions a and b such that a+b = ε and $P1^a$ holds for o and $P2^b$ holds for o.
- If P^ε is of the form ¬P1, iff $P1^\varepsilon$ does not hold for o. ♦

Example 3: For Q = (*travels-by* helicopter) ∧ (*owns* cat), we see that Q^0 holds for nobody; Q^1 holds for nobody; Q^2 holds for nobody; Q^3 holds for John; Q^4 holds for {Ann, Bob, John}; Q^5 holds for {Ann, Bob, Ed, John}, as well as Q^6, Q^7...

[11] conf (water-vehicle, boat) = 1; conf (plant, bird) = 2; conf (medium-weight, heavy) = 0.5; they add to 3.5.

Figure 6. Query *(address = california)$_1$* returns customers in California with confusion 1

```
select customer.name, customer.address
from customer
where conf(customer.address,'california')<=1

NAME             ADDRESS
East coast meat  florida
Media Tools      new york
Tom's Hamburgers pasadena
Microsol         silicon valley
Tampa tobacco    tampa
Texas fruits     texas
```

4 Querying a Database with Predicates that are imperfectly fulfilled

Extended SQL. To query with controlled precision a table T of a database, SQL is extended by these constructs:

- `conf(R,s)`≤ε, a SQL representation for $(R=s)_\varepsilon$, is a condition procedure used in a `WHERE` or `HAVING` clause, which is true iff *conf*(r, s)≤*ε. R* is the name of a column of T that is a hierarchical variable (a variable or column having hierarchical values), *r* is each of these values, and *s* is the intended or expected qualitative value. ♦ *Example*: `conf(address, mexico)≤ 0` represents in extended SQL the predicate $(\text{address} = \text{mexico})_0$ and will select rows from Fig 7 whose address is Mexico with confusion 0; that is, all rows where `(address = r)` and *conf*(r, mexico)≤ 0. It returns rows 2 and 7.
- `conf(R)` is a SQL expression [a shorthand for `conf(R, s)`], used in ‘`SELECT conf(R)`’, or ‘`GROUP BY conf(R)`’ or ‘`ORDER BY conf(R)`’, which returns for each row of table T, *conf(R, s).* ♦ T. That is, `conf(R)` returns for table T a list of numbers corresponding to the confusion of the value of property R for each row of T. *Example*: see Fig. 8.

Writing queries in extended SQL. The algorithm `EXPR` = replace(P) to replace a precision-controlled predicate P by its equivalent extended SQL expression `EXPR` is:

- *(R = s)$_\varepsilon$* should be replaced by ‘`conf(`’ *R* ‘`,`’ *s* ‘`)≤`’ ε, when R is the name of a column of a table, and s a symbolic value.
- *(P1 ∨ P2)$_\varepsilon$* should be replaced by ‘`(`’ replace(*P1$_\varepsilon$*) ‘`OR`’ replace (*P2$_\varepsilon$*) ‘`)`’.
- *(P1 ∧ P2)$_\varepsilon$* should be replaced by ‘`(`’replace(*P1$_\varepsilon$*) ‘`AND`’ replace (*P2$_\varepsilon$*)‘`)`’.
- *¬P* should be replaced by ‘`NOT` (’ replace (*P*) ‘`)`’.
- *(P1 ∨ P2)$^\varepsilon$* should be replaced by ‘`(`’replace(*P1*) ‘ `AND` ’ replace(*P2*) ‘ `AND (conf(`’ *P1* ‘`) +conf(`’ *P2* ‘`))≤`’ ε‘`)`’. ♦

Example: $(\text{industrial branch} = \text{food})_0 \wedge [(\text{address} = \text{pasadena}) \vee (\text{address} = \text{mexico city})]_1$ is replaced by `conf (industrial_branch, food)≤ 0 AND (conf(address, pasadena)≤ 1 OR conf (address, mexico city)≤ 1).` Example: *(address = Mexico City ∧ industrial branch = computer)^1*

is replaced by `(conf(address, Mexico City)≤1 AND conf (industrial_branch, computer)≤1 AND conf(address) + conf (industrial_branch) ≤1).`

Figure 7. Table of customers

```
         name          | industrial_branch |     address     | discount
-----------------------+-------------------+-----------------+---------
 Media Tools           | computers         | new york        |        0
 Garcia Productores    | tequila           | mexico city     |        0
 Tom's Hamburgers      | food              | pasadena        |        0
 Microsol              | software          | silicon valley  |        0
 East coast meat       | meat              | florida         |        0
 Luigi's italian food  | italian food      | north america   |        0
 Mole Doña Rosa        | mexican food      | mexico          |        0
 Texas fruits          | fruits            | texas           |        0
 Tampa tobacco         | cigars            | tampa           |        0
 Canada seeds          | food              | canada          |        0
```

Figure 8. Querying, sorting and showing values for *(address = california)*$_1$

```
select customer.name, customer.address,
conf(customer.address)
  from customer
  where conf(customer.address,'california')<=1
  order by conf(customer.address)

  NAME             ADDRESS        CALIFORNIA
  Tom's Hamburgers pasadena       0
  Microsol         silicon valley 0
  Media Tools      new york       1
  Tampa tobacco    tampa          1
  Texas fruits     texas          1
  East coast meat  florida        1
```

Queries: retrieving objects that match P_ε

Example (refer to Fig. 9): *(address = usa)*$_1$ will return any object whose value of property `address` can be used instead of `usa` with confusion 1. *Example*: Fig. 6 shows customers (of Fig. 7) for which *(address = california)*$_1$. This returns every record, except for Mole Doña Rosa customer because the customer's address is somewhere in Mexico and conf(mexico, california) has a value of 2 (by Fig. 9); except for Garcia Productores because the address is in Mexico City and conf(mexico city, california) is 2. Except for Luigi's Italian food because the address of the customer is somewhere in North Amerca and conf(north america, california) is 2, and so for Canada seeds, because conf(canada, california) is 2. For the customers of Fig. 7, we show in Fig. 9 the hierarchy for properties `address,` and for `industrial branch` we show in Fig. 10 its percentage hierarchy. *Example:* Fig. 8 shows how to sort the answers to *(address = california)*$_1$ by ascending confusion.

Figure 9. The addresses of customers form a simple hierarchy

```
Property: address;
hierarchy:
world{
   north_america{
      usa{
        california{
            silicon valley,
            pasadena, },
        new york{
            new york city },
        florida{
            miami,
            tampa },
        texas }
      canada,
      mexico{
        mexico city,
        jalisco{
            guadalajara } } } }
```

Figure 10. Hierarchy of industrial branch for customers, using percentage values. The values represent the products consumed in a business organization

```
Property: industrial branch;
hierarchy:
industrial branch(1){
    computer(.3){
        software(.12),
        hardware(.18)
    },
    human consumption(.7){
        food(.56){
            prepared food(.112){
                mexican food(.0448),
                italian food(.0672)
            },
            meat(.168),
            fruits(.28)
        }
        drinks and cigars(.14){
            drinks(.056){
                whiskey(.0112),
                beer(.028),
                tequila(.0168)
            },
            cigars(.084) } } }
```

5 Conclusion

The notions of hierarchy and hierarchical variable make it possible to measure the *confusion* when a value is used instead of another. This makes a natural generalization for predicates and queries. The notions were introduced and developed for arbitrary hierarchies formed by sets, bags, and lists, but they can be extended to mixed hierarchies too.

The concepts given herein have practical applications, since they mimic the manner in which people process qualitative values and disambiguate senses). Predicates with controlled precision $P_\varepsilon(o)$ (called "P holds for o with confusion (precision) ε") and $P^\varepsilon(o)$ (called "P holds for o with accumulated confusion (precision) ε") allow us to define precision-controlled retrieval of hierarchical values. These predicates permit "loose retrieval" (retrieval with defined confusion bounds) of objects that sit in a relational database. Moreover, such database could be an existing "normal" database (where no precision-controlled retrieval was defined), to which one or more definitions of hierarchies are attached. This in fact provides a procedure (a "kit") to extend *any* (existing) database to another in which imprecise retrievals are possible. Furthermore, this extension can be done without recompiling application programs. Old programs (with no precision retrieval) still work as before, whereas new application programs can exploit the "normal" database as if it were precision-controlled. In fact, a "normal" database now becomes a "precision-controlled" database when the extension (the kit) is applied to it. Some examples are given.

References

1. Guzman, A.: Finding the Main Themes in a Spanish Document. *Journal Expert Systems with Applications,* Vol. **14**, No. 1/2 (1998) 139-148
2. Guzman, A., Dominguez, C., Olivares, J.: Reacting to Unexpected Events and Communicating in spite of Mixed Ontologies. *Lecture Notes in Artificial Intelligence*, Vol. **2313**. Springer-Verlag, Berlin Heidelberg New York (2002) 377-386
3. Jesus M. Olivares-Ceja, Adolfo Guzman-Arenas. Concept similarity measures the understanding between two agents. *Lecture Notes in Computer Science* **3136** (Springer Verlag 2004) 182-194
4. S. Levachkine, A. Guzman-Arenas, Hierarchies measuring qualitative variables. *Lecture Notes in Computer Science*, Vol. **2945**, Springer-Verlag (2004) 258-270
5. A. Guzman-Arenas, S. Levachkine, Graduated errors in approximate queries using hierarchies and ordered sets. *Lecture Notes in Artificial Intelligence*, Vol. **2972**, Springer-Verlag (2004) 119-128
6. S. Levachkine and A. Guzman-Arenas. *Confusion between hierarchies partitioned by a percentage rule*. Unpublished manuscript.
7. S. Levachkine and A. Guzman-Arenas. Hierarchy as a new data type for qualitative variables. Submitted to *Data and Knowledge Engineering*.