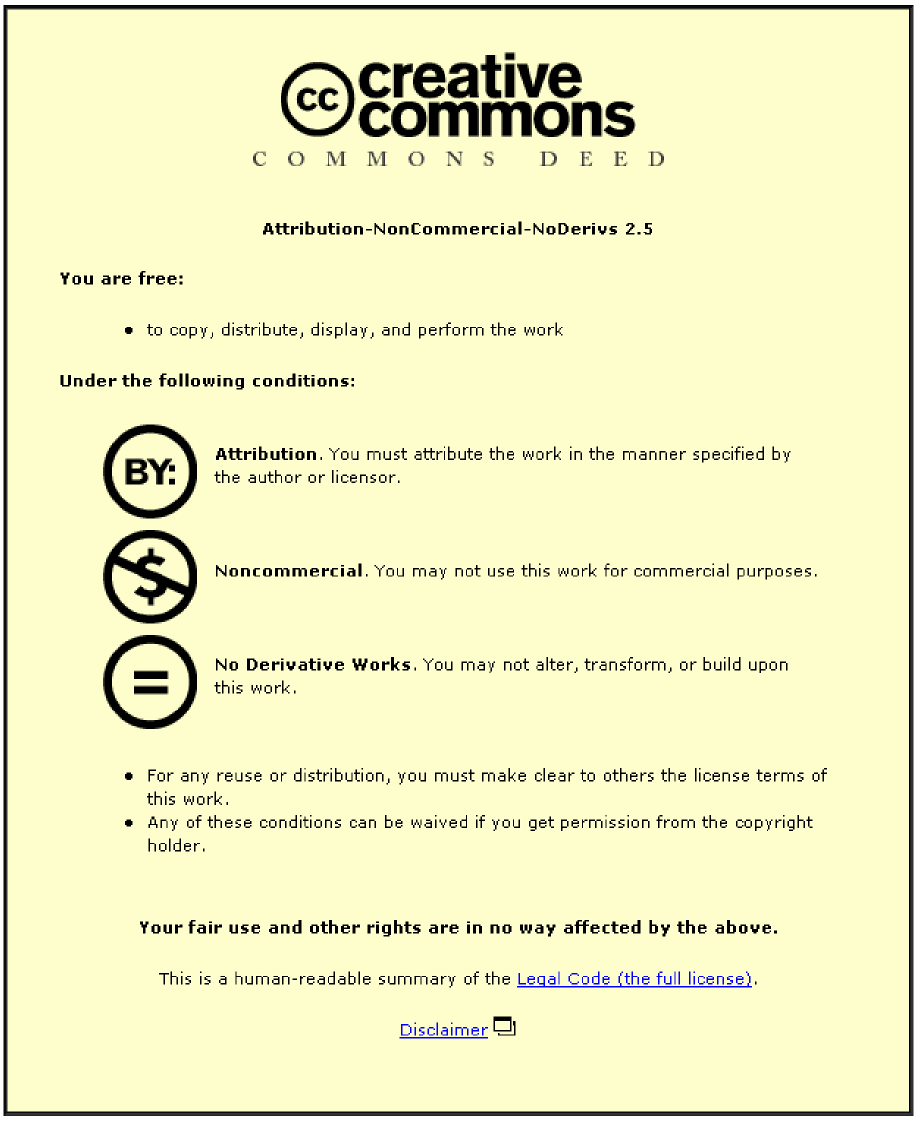

A Set Theoretic View of the ISA Hierarchy

Yee Chung Cheung, Paul Wai Hing Chung and Ana Sălăgean

Department of Computer Science
Loughborough University
Loughborough, UK
P.W.H.Chung@lboro.ac.uk, A.M.Salagean@lboro.ac.uk

Abstract. The ISA (is-a) hierarchies are widely used in the classification and the representation of related objects. In terms of assessing similarity between two nodes, current distance approaches suffer from its nature that only parent-child relationships among nodes are captured in the hierarchy. This paper presents an idea of treating a hierarchy as a set rather than as a tree in the traditional view. The established set theory is applied to provide the foundation where the relations between nodes can be mathematically specified and results in a more powerful and logical assessment of similarity between nodes.

Keywords: fuzzy matching, information retrieval, knowledge representation.

1 Introduction

The ISA hierarchies are widely used to classify and represent domain concepts. In its tree structure, the root represents the most general concept and its child nodes represent more specific concepts. Each node can be further decomposed as necessary. In a hierarchy the parent-child relation is the only one that is explicitly represented between nodes. The mainstream approaches to assess the similarity between nodes in a hierarchy are developed based on the idea of conceptual distance. The conceptual distance between two nodes is defined in terms of the length of the shortest path that connects the nodes in a hierarchy (Rada, [5]). However, similarity assessment based on conceptual distance does not always provide satisfactory results. This paper describes an alternative approach that views a hierarchy as a set which enables richer information to be specified. In this view the established set theory can be applied to both hierarchy specification and similarity assessment. The next section gives a brief description of similarity assessment based on conceptual distance. The third section presents the set theoretic approach. The application of the set theoretic view to similarity assessment is demonstrated in section four. A number of examples of capability matching are used throughout the paper. The paper ends with a discussion and conclusion section.

2 Conceptual Distance Approaches

Figure 1 is a simple capability ontology of programming skills represented as a hierarchy. For example, the term Object-Oriented represents the general concept of object-oriented programming skills; the term VB means Visual Basic programming skills. The parent-child relation between Object-Oriented and VB can be interpreted as VB programming skill is a kind of Object-Oriented programming skill. This capability ontology can be used to describe the skills of agents and the required capability to perform specific tasks.

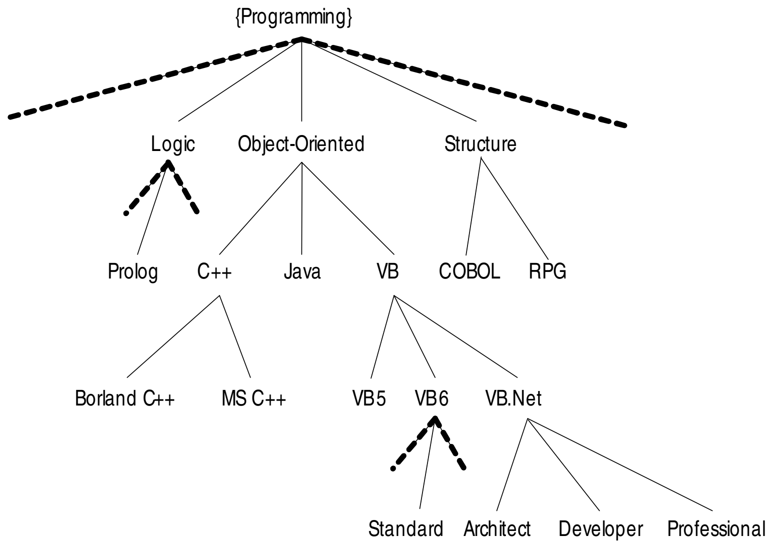

Fig. 1. A simple ISA hierarchy of programming skills

To identify the most appropriate agent for a given task, it is required to assess the goodness of fit (GOF) of an agent's skills against the required capability. The GOF is represented as a number in the interval [0, 100] where the upper limit 100 implies a perfect match. In the following examples, o_a refers to the capability of an available agent and o_r refers to the required capability for performing a task. The following equation, taken from [4], defines GOF based on the distance approach:

$$\text{GOF} = \left(1 - \frac{IP + EP}{IR + ER}\right) \times 100$$

where IP is the number of edges on the path between o_r and the common ancestor of o_r and o_a; EP is the number of edges of the path between o_a and the common ancestor of o_r and o_a; IR is the number of edges on the path between o_r and the root of the hierarchy; ER is the number of edges on the path between o_a and the root of the hierarchy.

Table 1 shows the results of applying this equation to a few examples based on Figure 1. From these examples, it can be seen that this approach does not always produce appropriate GOF values. Consider examples 2 and 3: if the required capability is Java, both available capabilities VB and C++ have the same GOF with value 50. However, an agent who knows C++ may require less effort to learn Java than another agent who knows VB. Another problem can be found in examples 4 and 5. In example 5, the required capability is C++ and the availability is general capability in Object-Oriented. However, it has the same GOF as example 4 where the required capability is any Object-Oriented skills and the available capability is C++. Finally, examples 6 and 7 show a serious problem: when either o_a or o_r is the root, the GOF value is always 0.

Example	o_a	o_r	GOF(o_a, o_r)
1	Java	Java	100
2	VB	Java	50
3	C++	Java	50
4	C++	Object-Oriented	66
5	Object-Oriented	C++	66
6	MS C++	Programming	0
7	Programming	MS C++	0

Table 1. Examples GOF using traditional distance approach

Chung and Jefferson suggested in [3] that different types of relationship between two nodes in a hierarchy have to be dealt with appropriately. They identified four different categories in matching domain concepts in a hierarchy, which are:

1. o_a is the same as o_r;
2. o_a is a descendant of o_r in the hierarchy;
3. o_a is an ancestor of o_r in the hierarchy;
4. o_a and o_r are on different branches in the hierarchy.

In category 1, as o_a is the same as o_r, i.e. $o_a = o_r$, it is obvious that GOF$(o_a, o_r) = 1$. In category 2, o_a is a concept of o_r. Therefore, if a task requires o_r then someone who knows o_a is suitable, and thus GOF$(o_a, o_r) = 1$. In category 3, o_a is more general than o_r. It means that o_a may or may not be what the user is required. A general rule for the domain is required. In category 4, o_a and o_r are on different branches in the hierarchy. Their investigation concludes that it is inappropriate to apply a general rule to determine the GOF value in this category. Nodes on different branches in a hierarchy may or may not be related. It is up to the domain experts to determine how closely two nodes are related or not related at all.

On the other hand, in [6] Sussna identified that besides the length of the path, the specificity of two nodes in the path (measured by the depth in the hierarchy) is an important parameter that affects the distance measure. In his work, a weight is assigned to each edge in the hierarchy and the total weight of the path between two nodes is calculated. The weights try to capture the fact that for the same path length, nodes lower in the hierarchy seem to be conceptually closer. In another similar work, [2], Agirre and Rigau took the density of concepts in the hierarchy into consideration: concepts in a deeper part of the hierarchy should be ranked closer, and the Conceptual Density [1] formula is used to provide more accurate results. Although the above works improve the assessment of similarity, they still suffer from the nature of the ISA hierarchy where only the parent-child relation is captured.

3 Set Theoretic Approach

We propose to view a hierarchy as a collection of sets and their inclusion relationships. Namely to each node in the tree we associate a set and each edge from a parent S to a child A represents the fact that the set A is included in the set S, i.e. $A \subseteq S$. This corresponds to the intuition that the notion A is conceptually included in the more general notion S. Different children of the same parent may or may not overlap. This also corresponds intuitively to the fact that the concepts may or may not have some degree of similarity.

We also quantify the "size" of the sets by defining a measure function on the set of all subsets of the root set. For each such set A its measure is a real number $\mu(A)$ with $\mu(A) \geq 0$. As usual the measure function will have the properties:

1. $\mu(\emptyset) = 0$ (The empty set has size 0)
2. If $A \subseteq B$ then $\mu(A) \leq \mu(B)$
3. If A and B are disjoint then $\mu(A \cup B) = \mu(A) + \mu(B)$.

We are interested not so much in the sizes of the sets but rather in their relative sizes. For each set A except the root we define the quantity $P(A)$ representing the relative size of A against the size of its parent set S, i.e. $P(A) = \mu(A)/\mu(S)$. Intuitively this quantifies what proportion of the general concept S is covered by the concept A. Obviously, since $A \subseteq S$, we have $0 \leq P(A) \leq 1$. For each parent S having children $A_1, A_2, \ldots, A_k$ we assume we are given $P(A_1), P(A_2), \ldots, P(A_k)$ and $P(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_t})$ for all $2 \leq t \leq k$ and $1 \leq i_1 < i_2 < \ldots < i_t \leq k$.

We make an important simplifying assumption, namely that each child is, in a sense "uniformly distributed" throughout its parent set. More precisely, if a node S has children A_1 and A_2 which are not disjoint (i.e. $A_1 \cap A_2 \neq \emptyset$) and furthermore A_1 has children B_1 and B_2, then say B_1 appears in the same proportion in $A_1 \cap A_2$ as in A_1, that is $\mu(B_1 \cap A_1 \cap A_2)/\mu(A_1 \cap A_2) = P(B_1)$. In the sequel we will call this assumption "the uniformity property".

We are now ready to define $\mathrm{GOF}(o_a, o_r)$ for our model. Intuitively, we want to measure what proportion of the required notion o_r is covered by the available notion o_a. Therefore, we define

$$\mathrm{GOF}(o_a, o_r) = 100 \frac{\mu(o_a \cap o_r)}{\mu(o_r)}. \tag{1}$$

We will look in more detail at how can this be computed according to the positions of o_a and o_r in the hierarchy. A summary will be given in Table 2.

If $o_r = o_a$ or $o_r \subset o_a$ then $o_a \cap o_r = o_r$ so (1) becomes $\mathrm{GOF}(o_a, o_r) = 100$. This fits well with the intuition that we have a perfect match in this case.

If $o_a \subset o_r$ then $o_a \cap o_r = o_a$ so (1) becomes $\mathrm{GOF}(o_a, o_r) = 100\mu(o_a)/\mu(o_r)$. This can be computed as follows: assume the path in the tree from o_a to its ancestor o_r consists of the sets $o_a \subseteq B_1 \subseteq B_2 \subseteq \ldots \subseteq B_u \subseteq o_r$. Then

$$\mathrm{GOF}(o_a, o_r) = 100\frac{\mu(o_a)}{\mu(o_r)} = \frac{\mu(o_a)}{\mu(B_1)} \cdot \frac{\mu(B_1)}{\mu(B_2)} \cdot \ldots \cdot \frac{\mu(B_u)}{\mu(o_r)}$$

hence

$$\mathrm{GOF}(o_a, o_r) = 100P(o_a)P(B_1)\cdots P(B_u)$$

Finally we have the case when none of o_a and o_r are included in the other. We look first at the situation where o_a and o_r are siblings, i.e. both are children of the same parent S. We have:

$$\mathrm{GOF}(o_a, o_r) = 100\frac{\mu(o_a \cap o_r)}{\mu(o_r)} = 100\frac{\frac{\mu(o_a \cap o_r)}{\mu(S)}}{\frac{\mu(o_r)}{\mu(S)}} = 100\frac{P(o_a \cap o_r)}{P(o_r)}$$

For the more general case when none of o_a and o_r are included in the other and they are not siblings, $\mathrm{GOF}(o_a, o_r)$ can be computed as follows: let S be the common ancestor of o_a and o_r and let the path from o_a to S consist of the sets $o_a \subseteq B_1 \subseteq B_2 \subseteq \ldots \subseteq B_u \subseteq S$ and the path from o_r to S consist of the sets $o_r \subseteq C_1 \subseteq C_2 \subseteq \ldots \subseteq C_v \subseteq S$. Then, as before, we have $\frac{\mu(o_a)}{\mu(B_u)} = P(o_a)P(B_1)\cdots P(B_{u-1})$. Due to the uniformity property, o_a occupies uniformly a proportion $\mu(o_a)/\mu(B_u)$ of any subset of B_u, in particular of $B_u \cap o_r$. This means

$$\frac{\mu(o_a \cap o_r)}{\mu(B_u \cap o_r)} = \frac{\mu(o_a \cap B_u \cap o_r)}{\mu(B_u \cap o_r)} = \frac{\mu(o_a)}{\mu(B_u)}$$

so $\mu(o_a \cap o_r) = \mu(B_u \cap o_r)\mu(o_a)/\mu(B_u)$. We still have to compute $\mu(B_u \cap o_r)$. Since $o_r \subseteq C_v$ we have $\mu(B_u \cap o_r) = \mu(B_u \cap C_v \cap o_r)$. Again by the uniformity property, $B_u \cap C_v$ occupies uniformly a proportion $\mu(B_u \cap C_v)/\mu(C_v)$ of any subset of C_v, in particular of o_r. Hence

$$\frac{\mu(B_u \cap o_r)}{\mu(o_r)} = \frac{\mu(B_u \cap C_v \cap o_r)}{\mu(o_r)} = \frac{\mu(B_u \cap C_v)}{\mu(C_v)}$$

so $\mu(B_u \cap o_r) = \mu(o_r)\mu(B_u \cap C_v)/\mu(C_v)$. So we can compute

$$\mathrm{GOF}(o_a, o_r) = 100\frac{\mu(o_a \cap o_r)}{\mu(o_r)} = 100\frac{\mu(B_u \cap o_r)\mu(o_a)}{\mu(B_u)\mu(o_r)} = 100\frac{\mu(B_u \cap C_v)\mu(o_a)}{\mu(C_v)\mu(B_u)}$$

and finally

$$\mathrm{GOF}(o_a, o_r) = 100\frac{P(B_u \cap C_v)}{P(C_v)}P(o_a)P(B_1)\cdots P(B_{u-1}).$$

	GOF(o_a, o_r)	Explanations
General formula	$100\frac{\mu(o_a \cap o_r)}{\mu(o_r)}$	
$o_a = o_r$	100	
$o_a \supset o_r$	100	
$o_a \subset o_r$	$100P(o_a)P(B_1)\cdots P(B_u)$	$o_a \subseteq B_1 \subseteq \ldots \subseteq B_u \subseteq o_r$
o_a and o_r siblings	$100\frac{P(o_a \cap o_r)}{P(o_r)}$	
o_a and o_r arbitrary	$100\frac{P(B_u \cap C_v)}{P(C_v)}P(o_a)P(B_1)\cdots P(B_{u-1})$	$o_a \subseteq B_1 \subseteq \ldots \subseteq B_u \subseteq S$ $o_r \subseteq C_1 \subseteq \ldots \subseteq C_v \subseteq S$

Table 2. Formulae for GOF in the set approach

Table 2 summarises the formulae for computing GOF using the set approach. Examples of computations of GOF using this definition will be given in the next section.

We note that storing at each node all the quantities $P(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_t})$ for all $1 \leq t \leq k$ and $1 \leq i_1 < i_2 < \ldots < i_t \leq k$ can lead to excessive memory usage. It also provides a level of detailed information that often will be neither available nor needed. We can simplify the representation as follows. For each parent S having children $A_1, A_2, \ldots, A_k$ we assume we are given $P(A_1 \cap A_2 \cap \ldots \cap A_k)$ and, optionally $P(A_1), P(A_2), \ldots, P(A_k)$ and $P(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_t})$ for all $2 \leq t < k$ and $1 \leq i_1 < i_2 < \ldots < i_t \leq k$. If we are not given $P(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_t})$ for some t and some $1 \leq i_1 < i_2 < \ldots < i_t \leq k$, we assume by default that $A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_t} = A_1 \cap A_2 \cap \ldots \cap A_k$, and therefore $P(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_t}) = P(A_1 \cap A_2 \cap \ldots \cap A_k)$. If $P(A_1), P(A_2), \ldots, P(A_k)$ are not given then we assume by default that $P(A_1) = P(A_2) = \ldots = P(A_k)$ and $S = A_1 \cup A_2 \cup \ldots A_k$. We can then deduce the values $P(A_i)$ using the inclusion-exclusion formula:

$$\begin{aligned}\mu(A_1 \cup A_2 \cup \ldots A_k) = &\sum_{i=1}^{k} \mu(A_i) - \sum_{1 \leq i_1 < i_2 \leq k} \mu(A_{i_1} \cap A_{i_2}) + \ldots + \\ &+(-1)^{t+1} \sum_{1 \leq i_1 < i_2 < \ldots < i_t \leq k} \mu(A_{i_1} \cap \ldots \cap A_{i_t}) + \ldots + \\ &+(-1)^{k+1} \mu(A_1 \cap A_2 \cap \ldots \cap A_k)\end{aligned}$$

Dividing by $\mu(S)$ we obtain

$$1 = \sum_{i=1}^{k} P(A_i) - \sum_{1 \leq i_1 < i_2 \leq k} P(A_{i_1} \cap A_{i_2}) + \ldots +$$

$$+(-1)^{t+1} \sum_{1 \le i_1 < i_2 < \ldots < i_t \le k} P(A_{i_1} \cap \ldots \cap A_{i_t}) + \ldots +$$
$$+(-1)^{k+1} P(A_1 \cap A_2 \cap \ldots \cap A_k) \qquad (2)$$

When only $P(A_1 \cap A_2 \cap \ldots \cap A_k)$ is given we have therefore $1 = kP(A_1) - (k-1)P(A_1 \cap A_2 \cap \ldots \cap A_k)$, hence for all i we have

$$P(A_i) = \frac{1 + (k-1)P(A_1 \cap A_2 \cap \ldots \cap A_k)}{k}. \qquad (3)$$

4 Example

The hierarchy given in Figure 1 can be enriched using some sample similarity figures, obtaining the hierarchy given in Figure 2.

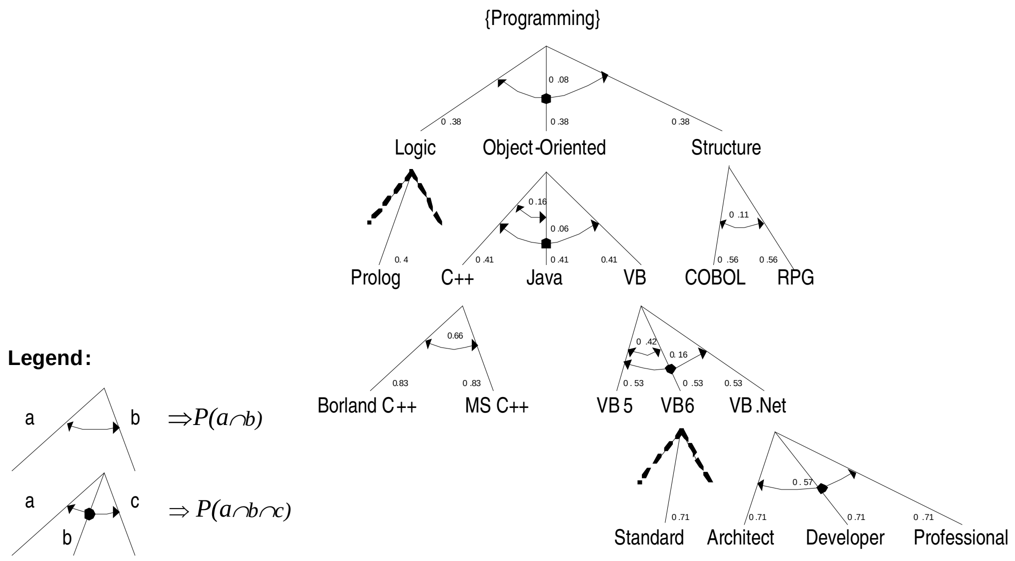

Fig. 2. Enriched hierarchy

The numerical values given in Figure 2 are obtained as follows. Take for example the concept Object-Oriented and its three children, the concepts C++, Java and VB. We assume that the three children have the same size i.e. $P(\text{C++}) = P(\text{Java}) = P(\text{VB}) = x$. The amount of overlap of the concepts has to be obtained from domain experts. Assume the experts indicate that C++ and Java are 40% similar, i.e. $P(\text{C++} \cap \text{Java}) = 0.4P(\text{Java}) = 0.4x$ and that the similarity of all the three notions is 15%, i.e. $P(\text{C++} \cap \text{Java} \cap \text{VB}) = 0.15P(\text{Java}) = 0.15x$. The similarity of VB with C++ and of VB with Java are not given so it is assumed that they equal the similarity of all the three concepts, i.e. $P(\text{C++} \cap \text{VB}) =$

$P(\text{Java} \cap \text{VB}) = P(\text{C++} \cap \text{Java} \cap \text{VB}) = 0.15x$. Using formula (2) we can then compute

$$\begin{aligned}
1 &= P(\text{C++}) + P(\text{Java}) + P(\text{VB}) - P(\text{C++} \cap \text{Java}) - P(\text{Java} \cap \text{VB}) - \\
&\quad - P(\text{C++} \cap \text{VB}) + P(\text{C++} \cap \text{Java} \cap \text{VB}) \\
&= 3x - 0.4x - 0.15x - 0.15x + 0.15x = 2.45x
\end{aligned}$$

Hence $x = 1/2.45 = 0.41$, $P(\text{C++} \cap \text{Java}) = 0.4x = 0.16$ and $P(\text{C++} \cap \text{Java} \cap \text{VB}) = 0.15x = 0.06$.

We compute GOF for several combinations of o_a and o_r using the formulae introduced in the previous section.

$$\begin{aligned}
\text{GOF}(\text{Java}, \text{Java}) &= 100 \\
\text{GOF}(\text{VB}, \text{Java}) &= 100\frac{P(\text{VB} \cap \text{Java})}{P(\text{Java})} = 100\frac{0.06}{0.41} = 15 \\
\text{GOF}(\text{C++}, \text{Java}) &= 100\frac{P(\text{C++} \cap \text{Java})}{P(\text{Java})} = 100\frac{0.16}{0.41} = 39 \\
\text{GOF}(\text{MS C++}, \text{Programming}) &= 100P(\text{MS C++})P(\text{ C++})P(\text{Object-oriented}) \\
&= 100 \times 0.83 \times 0.41 \times 0.38 = 13 \\
\text{GOF}(\text{Prolog}, \text{MS C++}) &= 100\frac{P(\text{Logic} \cap \text{Object-Oriented})}{P(\text{Object-Oriented})}P(\text{Prolog}) \\
&= 100\frac{0.08}{0.39}0.4 = 8 \\
\text{GOF}(\text{MS C++}, \text{Prolog}) &= \frac{P(\text{Object-Oriented} \cap \text{Logic})}{P(\text{Logic})}P(\text{MS C++})P(\text{C++}) \\
&= 100\frac{0.08}{0.39}0.83 \times 0.41 = 7
\end{aligned}$$

Example	o_a	o_r	GOF(o_a, o_r) Distance	GOF(o_a, o_r) Set view
1	Java	Java	100	100
2	VB	Java	50	15
3	C++	Java	50	39
4	C++	Object-Oriented	66	41
5	Object-Oriented	C++	66	100
6	MS C++	Programming	0	13
7	Programming	MS C++	0	100
8	Prolog	MS C++	0	8
9	MS C++	Prolog	0	7

Table 3. Examples GOF using the set view approach

These results and a few more are summarised in Table 3. We note that the results obtained using the set approach are more reasonable than the ones using distance. Examples 2 and 3 show that in the set theoretic approach we can model the fact that C++ is more similar to Java than VB is to Java. In examples 4 through to 7, one of the concepts is a descendant of the other. While the distance approach was unable to differentiate between the situation when the required concept is a descendant of the available one or the other way around, the set approach does, yielding a GOF of 100 in the first case and an intermediate GOF in the converse situation. Also, in the latter situation, the set approach GOF will be progressively lower as the available concept is further down the tree from the required concept (compare examples 4 and 6). When one of the concepts is the root of the tree, the distance approach yields a GOF of 0, which is certainly not the expected behaviour, while the set theoretic view gives again a GOF of 100 in the case when the required concept is a descendant of the available one, and an intermediate GOF in the converse situation. Finally, in examples 8 and 9 the common ancestor of the two concepts is the root. The distance approach produces a GOF of 0 in all such situations, while the set approach produces some small GOF as intuitively expected, as the two concepts, although distantly related, do still have some small degree of similarity.

5 Group matching

Supporting group matching that assesses the GOF of a group of objects against another group is an additional advantage of the set theoretic view. Agent selection for tasks is an example of such matching. To perform a task an agent may require multiple capabilities which form a capability set. Similarly, each agent may possess a set of capabilities. To find out who is the most appropriate agent for a given task, the capability sets possessed by the agents have to be matched against the required capability set.

If an agent has capabilities $A_1, \dots, A_k$ and the required capabilities are $R_1, \dots, R_t$ then GOF will be defined as the proportion of the union of required capabilities which is covered by the union of the available capabilities, i.e.

$$\text{GOF}(\cup_{i=1}^{k} A_i, \cup_{j=1}^{t} R_j) = 100 \frac{\mu((\cup_{i=1}^{k} A_i) \cap (\cup_{j=1}^{t} R_j))}{\mu(\cup_{j=1}^{t} R_j)}$$

The computation is in this case more complex and will be the subject of further work.

6 Discussion and Conclusion

The is-a hierarchy is a useful representation that is widely used. However, in terms of assessing the similarity between nodes, it is limited by its nature that the parent-child relation is the only one that is explicitly represented. The set theoretic view is therefore proposed as a solution to this problem.

The basic idea is viewing a hierarchy as a universal set and the child nodes are subsets of this universal set. The similarity is represented by the overlapping between sets. This approach offers a number of advantages over the traditional hierarchy.

First, well-established set theory can be applied to describe hierarchies and the relation between nodes. It allows the similarity between the nodes to be precisely specified while traditional hierarchy only has parent-child connections.

Secondly, the set theory can be used to assess the similarity between any two nodes for which the similarity is not given directly. The outcomes are more reasonable as a result of the enrich relation provided in a set-based description.

Thirdly, the set theoretic approach supports group matching. Current distance based approaches support only one to one matching and lack a theory to support combining the individual results.

Fourthly, the hierarchy description and set description can both be used in the same application. The tree view in hierarchy provides a simple visual explanation to the user and the set description enables a precise assessment at the back end.

The above sections demonstrate how a set theoretic view can improve the traditional hierarchy. Providing appropriate similarity figures is critical to generating a precise set-based description for a hierarchy. Experience shows that reasonable similarity values between different nodes can be obtained from domain experts. Although the given figures vary slightly between experts, the trends are similar. For example, there is general agreement that the similarity between C++ and Java is higher than between C++ and VB.

In conclusion, the set theoretic approach enables more precise description and accurate assessment over the traditional distance-based approaches. As the hierarchical and set theoretic views can both be used in the same application, the simplicity of hierarchies will not be impaired while the set theoretic view provides additional support.

References

1. E. Agirre and G. Rigau. A proposal for word sense disambiguation using conceptual distance. In *International Conference on Recent Advances in Natural Language Processing, Tzigov Chark, Bulgaria*, September 1995.
2. E. Agirre and G. Rigau. Word sense disambiguation using conceptual density. In *Proceedings of COLING-96*, 1996.
3. P.W.H. Chung and M. Jefferson. A fuzzy approach to accessing accident databases. *Applied Intelligence*, 9:129–137, 1998.
4. Cognitive Systems Inc., Boston. *ReMind Reference Manual*, 1992.
5. R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development an application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.
6. M. Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the Second International Conference on Information and knowledge Management*, Arlington, Virginia USA, 1993.