

# Database Summarization Using Fuzzy ISA Hierarchies

Do Heon LEE and Myoung Ho KIM

*Abstract*— Summary discovery is one of the major components of knowledge discovery in databases, which provides the user with comprehensive information for grasping the essence from a large amount of information in a database. In this paper, we propose an interactive top-down summary discovery process which utilizes fuzzy ISA hierarchies as domain knowledge. We define a generalized tuple as a representational form of a database summary including fuzzy concepts. By virtue of fuzzy ISA hierarchies where fuzzy ISA relationships common in actual domains are naturally expressed, the discovery process comes up with more accurate database summaries. We also present an informativeness measure for distinguishing generalized tuples that delivers much information to users, based on Shannon's information theory.

*Keywords*— data mining, summary discovery, fuzzy set application

## I. INTRODUCTION

AS the rapid growth of database volumes has deepened the gap between data generation and data understanding, *knowledge discovery in databases* has drawn increasing interest from various data-intensive applications[1]. A database summary is one important type of knowledge to be discovered, which provides the user with comprehensive information for grasping the essence of a focused database portion in an understandable manner. It also establishes a starting point to make useful inferences from large collections of data, and facilitates easy communication of observations about the problem domain[2].

Informally, our definition of database summarization is: a task that reduces a large number of actual database tuples into a relatively small number of generalized descriptions, i.e., *generalized tuples*. For example, a computer usage log table whose attribute scheme is (PROGRAM, USER, TIME), containing thousands of usage log records such as <vi, John, 23:20> and <emacs, Tom, 23:31>, could be reduced into a few generalized tuples, say, <editor, programmer, around midnight>, which delivers an assertion that programmers have executed editor programs around midnight.

Among several requirements for effective summary discovery techniques, we concentrate on the following ones: Firstly, it must be allowed to represent database summaries in terms of fuzzy concepts, i.e, concepts with fuzzy boundaries, since crisp concepts are occasionally too restrictive to express complex situations[3], [4]. Secondly, it must be possible to utilize fuzzy domain knowledge, since actual domain knowledge is apt to include fuzziness inherently. Thirdly, users must be able to interact with a summary

The authors are with the Department of Computer Science, Korea Advanced Institute of Science and Technology(KAIST), Taejeon, Korea. E-mail: {dhlee,mhkim}@dbserver.kaist.ac.kr

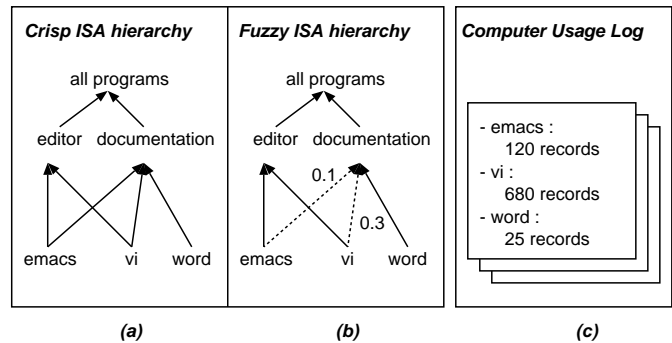


Fig. 1. (a) and (b) show crisp and fuzzy ISA hierarchies: Dotted lines represent partial ISA relationships as strong as the augmented fraction numbers between two incident concept nodes, while solid lines denote complete ISA relationships. (c) depicts a situation where database summarization is to be done.

discovery process to reflect their own discovery purposes.

ISA hierarchies are commonly used to exploit *specialization* relationships among domain concepts. However, ISA hierarchies including only crisp ISA relationships are not sufficient to express actual domain knowledge. For example, suppose we know that programs `emacs` and `vi` are used to edit source code, and a program `word` is used to write documents. In addition, suppose we also know that some users execute `emacs` and `vi` to write documents only on rare occasions. With only crisp relationships, there is no other way to represent the above mentioned domain knowledge except as shown in Figure 1-(a). If fuzzy relationships can be expressed, however, we would have an ISA hierarchy as shown in Figure 1-(b). Let us consider how different are the results that these two ISA hierarchies yield, in database summarization. Suppose a computer usage log table during a certain period contains 120, 680 and 25 log records of `emacs`, `vi` and `word` executions, respectively, as shown in Figure 1-(c), and we want to determine whether `editor` or `documentation` programs are mainly executed in that period. With the crisp ISA hierarchy, we cannot identify the majority of usage since  $120 + 680 = 800$  records are for `editor` programs, and  $120 + 680 + 25 = 825$  records are for `documentation` programs. On the other hand, if we exploit the fuzzy ISA hierarchy, we can conclude that `editor` programs have been mostly executed, since 120 and 680 records of `emacs` and `vi`, respectively, are known mainly for editing source codes not for writing documents.

The limitations of pure statistical analyses and classical inductive machine learning methods, as applied to actual knowledge discovery in databases, are well described in [1]. Recently, there has been database-oriented research on summary discovery techniques[2], [5], [6]. Yager pro-

poses notions of linguistic summaries with fuzzy terms such as “Most employees are *young* with truth value 0.6”. He presents guidelines to evaluate validity measures of a linguistic summary based on fuzzy set theory[2]. However, no specific procedure is given to construct such linguistic summaries themselves. Conjunctive summaries, where multiple attributes are included, are also not considered. DBLEARN adopts an attribute-oriented induction method to extract database summaries[5]. In its attribute-oriented induction, each attribute value of a tuple is substituted with a more general concept. After one pass of the substitution, equivalent classes of generalized tuples are identified and each class is regarded as a candidate summary. This bottom-up procedure is repeated until satisfactory summaries are obtained. However, it does not utilize fuzzy domain knowledge.

This paper is organized as follows: Section 2 defines a generalized tuple as a representation form of a database summary including fuzzy concepts. Fuzzy ISA hierarchies used as domain knowledge are introduced in Section 3. An interactive top-down summary discovery process is proposed in Section 4. Section 5 investigates how much information content a discovered summary delivers, based on Shannon’s information theory. Finally, Section 6 recapitulates and discusses future work.

## II. REPRESENTATION OF DATABASE SUMMARIES

Herein, we define a generalized tuple as a representational form of a database summary, and elaborate how to evaluate the validity of a generalized tuple with respect to a given database. We assume that all attributes appear in a single table, i.e., the universal relation assumption, without loss of generality, to avoid unnecessary complexity of the presentation. However, this work can be applied to any other data models where a database record can be regarded as a series of attribute values.

### A. Generalized tuples

There are many domain concepts having fuzzy boundaries in practice. For example, it is difficult to draw a crisp boundary of **documentation programs** in a set of programs since some programs such as **vi** and **emacs** can be thought as source code editors for some programmers but as word processors for some manual writers. It is more natural and useful to express such domain concepts in terms of fuzzy sets rather than crisp sets[4]. Thus, we use a vector of fuzzy sets to effectively represent a database summary.

A fuzzy set  $f$  on a domain  $D$  is defined by its membership function  $\mu_f(x)$ , which assigns a (positive) fraction number between 0 and 1 as the membership degree to each domain value[3]. Since  $\mu_f(x)$  represents the degree to which an element  $x$  belongs to a fuzzy set  $f$ , a conventional set is regarded as a special case of a fuzzy set whose membership degrees are either one or zero. Given two fuzzy sets,  $f_1$  and  $f_2$ , on a domain  $D$ ,  $f_1$  is called a *subset* of  $f_2$  and denoted as  $f_1 \subseteq f_2$ , iff  $\forall x \in D, \mu_{f_1}(x) \leq \mu_{f_2}(x)$ . In this paper, a special fuzzy set  $f$  such that  $\forall x \in D, \mu_f(x) = 1$ , is denoted as  $\omega$ .

*Definition 1* (Generalized tuple) A *generalized tuple* is defined as an  $m$ -ary tuple  $\langle f_1, \dots, f_m \rangle$  on an attribute scheme  $(A_1, \dots, A_m)$ , where  $f_j$ ’s are fuzzy sets and  $A_j$ ’s are attributes. Given two different generalized tuples,  $g_1 = \langle f_{11}, \dots, f_{1m} \rangle$  and  $g_2 = \langle f_{21}, \dots, f_{2m} \rangle$ , on the same attribute scheme,  $g_1$  is called a *specialization* of  $g_2$ , iff  $\forall j, f_{1j} \subseteq f_{2j}$ .

A generalized tuple  $\langle f_1, \dots, f_m \rangle$  on an attribute scheme  $(A_1, \dots, A_m)$ , is interpreted as an assertion that “each tuple has  $f_1, \dots, f_m$  for attributes  $A_1, \dots, A_m$ , respectively”. Note that an ordinary database tuple is also regarded as a generalized tuple whose fuzzy sets are singleton sets. A *singleton set* is a set having a unique element. An example of a generalized tuple with respect to an attribute scheme (PROGRAM, USER) is  $\langle \text{editor}, \text{programmer} \rangle$ . It implies an assertion that “The program is an editor and its user is a programmer.”, in other words, “A programmer has executed an editor program.”.

### B. Validity of generalized tuples

From the viewpoint of inductive learning, a given database (or a part of it) and a set of possible generalized tuples are regarded as an *instance space* and a *pattern space*, respectively[8]. Our summarization process searches a pattern space to choose valid generalized tuples with respect to a given instance space, i.e., a given database. Recall that the goal of database summarization is to generate representative descriptions embracing as many database tuples as possible. Thus, the validity of a generalized tuple is determined by the number of database tuples with which it is compatible, i.e. *statistical significance*. In the following, this notion is formulated as the support degree.

*Definition 2* (Support degree) The *support degree* of a generalized tuple  $g = \langle f_1, \dots, f_m \rangle$  with respect to a given collection<sup>1</sup> of database tuples  $C$  whose attribute scheme is  $(A_1, \dots, A_m)$ , is defined as follows:

$$SD(g | C) = \frac{|\sum_{t_i \in C} \otimes [\mu_{f_1}(t_i.A_1), \dots, \mu_{f_m}(t_i.A_m)]|}{|C|},$$

where  $\otimes$  is a  $t$ -norm operator[4],  $\mu_{f_j}(t_i.A_j)$  denotes the membership degree of an attribute  $A_j$  of a tuple  $t_i$  with respect to a fuzzy set  $f_j$ , and  $|C|$  denotes the cardinality of the collection  $C$ . We call generalized tuples with higher support degrees than a user-given threshold value, *qualified generalized tuples*.

We will denote  $SD(g | C)$  as  $SD(g)$  for simplicity as long as the data collection  $C$  is obvious in the context. Note that  $t$ -norm operators are used to obtain conjunctive notions of membership degrees in the fuzzy set theory[4]. Examples include MIN and probabilistic product operators.<sup>2</sup> Since  $\mu_{f_j}(t_i.A_j)$  denotes the membership degree of an attribute  $A_j$  of a tuple  $t_i$  with respect to a fuzzy set  $f_j$ , a  $t$ -norm

<sup>1</sup>We do not use the term, *a set of database tuples* or *a database relation* because a collection of database tuples is allowed to have duplicates. (See Section V.B for the detailed reason.)

<sup>2</sup>Since the usage of  $t$ -norm operators in the fuzzy set theory is analogous with that of the product operator in the probability theory, the symbol  $\otimes$ , that is analogous with  $\times$ , is commonly used to denote a specific instance of a  $t$ -norm operator.

TABLE I  
THE SUPPORT STRENGTH OF EXAMPLE DATA TUPLES

Tuple	$A_1$	$A_2$	$\mu_{f_1}(A_1)$	$\mu_{f_2}(A_2)$	$t_i$ supports $\langle f_1, f_2 \rangle$ as strong as
$t_1$	a	$\alpha$	1.0	0.3	$\otimes(1.0, 0.3) = 0.3$
$t_2$	b	$\beta$	0.1	1.0	$\otimes(0.1, 1.0) = 0.1$
$t_3$	b	$\alpha$	0.1	0.3	$\otimes(0.1, 0.3) = 0.1$

value over membership degrees of all attributes of a tuple  $t_i$ , i.e.,  $\otimes[\mu_{f_1}(t_i.A_1), \dots, \mu_{f_m}(t_i.A_m)]$ , represents how strongly the tuple  $t_i$  supports the assertion of a generalized tuple  $\langle f_1, \dots, f_m \rangle$ . As a result, the support degree of a generalized tuple is the normalized sum of such support strength of individual database tuples. In other words, the support degree implies the fraction of *supporting* database tuples to the total data collection.

The defined support degree has the following properties:

- Boundary conditions:
  - Given a generalized tuple  $g$ ,
  - $0 \leq SD(g) \leq 1$ .
  - If all fuzzy sets in  $g$  are  $\omega$ ,  $SD(g) = 1$ .
- Monotonicity:
  - Given two generalized tuples,  $g_1$  and  $g_2$  on the same attribute scheme,  $SD(g_1) \leq SD(g_2)$  if  $g_1$  is a specialization of  $g_2$ .

While boundary conditions are self-evident by definition, the monotonicity property needs some explanation. The following theorem shows the monotonicity property.

*Theorem 1* (Monotonicity of the support degree) Given two generalized tuples,  $g_1 = \langle f_{11}, \dots, f_{1m} \rangle$ , and  $g_2 = \langle f_{21}, \dots, f_{2m} \rangle$ , on an attribute scheme  $(A_1, \dots, A_m)$ ,  $SD(g_1) \leq SD(g_2)$  if  $\forall j \in \{1, \dots, m\}$ ,  $f_{1j} \subseteq f_{2j}$ .

*Proof:* The premise of the theorem implies that  $\forall j \in \{1, \dots, m\}$ ,  $[\forall x \in DOM(A_j), \mu_{f_{1j}}(x) \leq \mu_{f_{2j}}(x)]$ , where  $DOM(A_j)$  denotes the domain of an attribute  $A_j$ .

When a collection of database tuples is  $C$ ,

$$\begin{aligned} SD(g_1) &= \left[ \sum_{t_i \in C} \otimes_{j \in \{1, \dots, m\}} [\mu_{f_{1j}}(t_i.A_j)] \right] / |C| \\ &\leq \left[ \sum_{t_i \in C} \otimes_{j \in \{1, \dots, m\}} [\mu_{f_{2j}}(t_i.A_j)] \right] / |C| \\ &\quad \text{since } \mu_{f_{1j}}(t_i.A_j) \leq \mu_{f_{2j}}(t_i.A_j), \\ &= SD(g_2) \end{aligned}$$

Thus,  $SD(g_1) \leq SD(g_2)$ . ■

Let us look at an example of support degree computation. Suppose a generalized tuple  $g$  on an attribute scheme  $(A_1, A_2)$  is  $\langle f_1, f_2 \rangle$ , where fuzzy sets  $f_1$  and  $f_2$  are  $\{1.0/a, 0.1/b\}$  and  $\{0.3/\alpha, 1.0/\beta\}$ , respectively. If a data collection  $C$  is given as Table I,  $SD(g)$  is computed as follows: The first tuple  $t_1$  supports  $g$  as strong as 0.3, since its first and second attribute values,  $a$  and  $\alpha$ , belong to fuzzy sets,  $f_1$  and  $f_2$ , to the degrees, 1.0 and 0.3, respectively. Note that we use the MIN operator for the t-norm operation just for illustration throughout this paper. Similarly, both the second and third tuples support  $g$  as strong as

0.1. As a result, we can say that the generalized tuple  $g$  is supported by  $0.3 + 0.1 + 0.1 = 0.5$  tuples out of a total of three tuples, i.e., 17 % of the data collection.

### III. FUZZY DOMAIN KNOWLEDGE

An ISA hierarchy is an acyclic digraph  $(N, A)$ , where  $N$  and  $A$  are a set of concept nodes and a set of ISA arrows, respectively. If there is an ISA arrow from a concept node  $n_1$  to an other concept node  $n_2$ , we say that  $n_1$  ISA  $n_2$ , in other words,  $n_1$  is a specialized concept of  $n_2$ . While conventional ISA hierarchies have only crisp ISA arrows, fuzzy ISA hierarchies include fuzzy ISA arrows. The meaning of a fuzzy ISA arrow from  $n_1$  to  $n_2$  can be interpreted as  $n_1$  is a *partially* specialized concept of  $n_2$ . Without loss of generality, we suppose that the root node of any fuzzy ISA hierarchy is the special fuzzy set  $\omega$ , and each terminal node is a singleton set whose unique element is an atomic domain value, i.e., values appearing in actual database tuples. (Examples of a conventional crisp ISA hierarchy and a fuzzy ISA hierarchy can be found in Introduction of this paper.)

Since fuzzy ISA hierarchies are too flexible of a structure to be used directly in database summarization, we provide a method to resolve a given fuzzy ISA hierarchy into a collection of fuzzy sets defined on the same domain, and a *fuzzy set hierarchy* that focuses on the complete inclusion relationships.

*Definition 3* (Fuzzy set hierarchy) A *fuzzy set hierarchy* is a partially ordered set,  $(\Gamma, \subseteq)$  where  $\Gamma$  is a set of fuzzy sets defined on the domain  $D$ . The binary relation  $\subseteq$  is the (complete) set inclusion relationship between two fuzzy sets. A fuzzy set  $f_i$  is called a *direct subset* of another fuzzy set  $f_j$  if  $f_i \subseteq f_j$  and there is no other  $f_k$  such as  $f_i \subseteq f_k \subseteq f_j$ .

Recall that a fuzzy set  $f_i$  is said to be (completely) included by a fuzzy set  $f_j$  on the same domain, if for each domain element  $x$ ,  $\mu_{f_i}(x) \leq \mu_{f_j}(x)$ .

#### A. Transforming a fuzzy ISA hierarchy to a fuzzy set hierarchy

In fuzzy set theory[9], the elements of a fuzzy set can themselves be fuzzy sets, rather than atomic domain values. Ordinary fuzzy sets whose elements are atomic values are called level-1 fuzzy sets. Fuzzy sets whose elements are level- $(k-1)$  fuzzy sets are called level- $k$  fuzzy sets. Table II depicts some level- $k$  fuzzy sets. If two fuzzy sets have different levels, we cannot directly determine the inclusion relationship between them, since the domains are different. However, the level of a fuzzy set can be either *upgraded* or *downgraded* by some fuzzy set-theoretic treatments. Thus, if we want to determine the inclusion relationship between two fuzzy sets with different levels, we have to adjust their levels to the same through upgrading or downgrading levels.

*Upgrading* the level of a fuzzy set is trivial, since a level- $k$  fuzzy set can be simply rewritten as a level- $(k+1)$  singleton set whose unique element is the original level- $k$  fuzzy set. For example, a level-2 fuzzy set editor in Table II can be

TABLE II  
LEVEL-K FUZZY SETS

set label	membership function	level
engi	{1.0/editor, 1.0/docu, 0.8/spread}	3
busi	{1.0/docu, 1.0/spread}	3
editor	{1.0/emacs, 1.0/vi}	2
docu	{0.1/emacs, 0.3/vi, 1.0/word}	2
spread	{0.1/word, 1.0/wright}	2
emacs	{1.0/emacs}	1
vi	{1.0/vi}	1
word	{1.0/word}	1
wright	{1.0/wright}	1

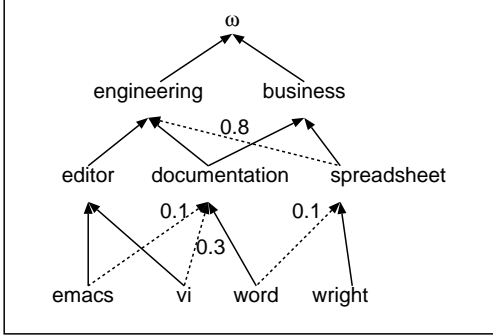


Fig. 2. A fuzzy ISA hierarchy on computer programs

thought as a level-3 fuzzy set such as  $\{1.0/\text{editor}\}$ . *Downgrading* the level of a fuzzy set is done by a *support fuzzification* technique based on the extension principle[4]. Rather than spending a large space to explain support fuzzification precisely, we will explain it by example. Interested readers are recommended to refer to Zadeh's original paper[10].

The transformation procedure of a fuzzy ISA hierarchy to a fuzzy set hierarchy, is composed of three steps as follow:

1. Downgrade several levels of fuzzy sets in a fuzzy ISA hierarchy to level-1 fuzzy sets.
2. By probing pairwise inclusion relationships, elicit a partial order relation on those level-1 fuzzy sets obtained in the previous step.
3. Draw arrows between fuzzy sets and their direct subsets based on the partial order relation.

Let us demonstrate the transformation procedure step by step with an example. Figure 2 shows an example fuzzy ISA hierarchy on computer programs that could be used in computer usage analysis. Note that a fuzzy set  $f$  in a fuzzy ISA hierarchy is a level- $k$  fuzzy set, if the maximal path length from  $f$  to terminal nodes is  $k - 1$ . The several levels of fuzzy sets in the fuzzy ISA hierarchy of Figure 2 are in Table II.

In the first step, all level- $k$  ( $k > 1$ ) fuzzy sets are downgraded to level-1 fuzzy sets through support fuzzification. For example, a level-3 fuzzy set **engineering** in Table II is transformed to a level-2 fuzzy set as follows:

$$\text{engineering} = \{1.0/\text{editor}, 1.0/\text{documentation}, 0.8/\text{spreadsheet}\}$$

$$\begin{aligned} &= \{1.0/\{1.0/\text{emacs}, 1.0/\text{vi}\}, \\ &\quad 1.0/\{0.1/\text{emacs}, 0.3/\text{vi}, 1.0/\text{word}\}, \\ &\quad 0.8/\{0.1/\text{word}, 1.0/\text{wright}\}\} \\ &= \{\otimes(1.0, 1.0)/\text{emacs}, \otimes(1.0, 1.0)/\text{vi}, \\ &\quad \otimes(1.0, 0.1)/\text{emacs}, \otimes(1.0, 0.3)/\text{vi}, \\ &\quad \otimes(1.0, 1.0)/\text{word}, \otimes(0.8, 0.1)/\text{word}, \\ &\quad \otimes(0.8, 1.0)/\text{wright}\} \\ &= \{1.0/\text{emacs}, 1.0/\text{vi}, 0.1/\text{emacs}, 0.3/\text{vi}, \\ &\quad 1.0/\text{word}, 0.1/\text{word}, 0.8/\text{wright}\} \\ &= \{\oplus(1.0, 0.1)/\text{emacs}, \oplus(1.0, 0.3)/\text{vi}, \\ &\quad \oplus(1.0, 0.1)/\text{word}, 0.8/\text{wright}\} \\ &= \{1.0/\text{emacs}, 1.0/\text{vi}, 1.0/\text{word}, 0.8/\text{wright}\} \end{aligned}$$

where  $\otimes$  and  $\oplus$  are a t-norm and a t-conorm operators, respectively. In contrast with t-norm operators, t-conorm operators are used to obtain disjunctive combinations of membership degrees. Herein, we use MIN and MAX for t-norm and t-conorm operations just for illustration. But, there are also several alternatives for t-conorm operators.

To envisage the implication of support fuzzification, let us consider the reason why the membership degree of the element **word** is determined as 1.0. Since **documentation** is a member of **engineering**, and **word** is a member of **documentation**, **word** is also regarded as a member of **engineering**. By this transitivity, the membership degree of **word** to **engineering** is determined as  $\otimes(\mu_{\text{engineering}}(\text{documentation}), \mu_{\text{documentation}}(\text{word})) = \otimes(1.0, 1.0) = 1.0$ . Meanwhile, the alternative transitivity that **spreadsheet** is a member of **engineering**, and **word** is a member of **spreadsheet**, also implies that **word** is regarded as a member of **engineering**. If we follow the latter transitivity, the membership degree of **word** to **engineering** is determined as  $\otimes(\mu_{\text{engineering}}(\text{spreadsheet}), \mu_{\text{spreadsheet}}(\text{word})) = \otimes(0.8, 0.1) = 0.1$ . Note that as far as either of such two transitivity relationships exists, i.e., the disjunctive combination of two facts, the membership of **word** to **engineering** holds. Thus, the membership degree of **word** to **engineering** is concluded as  $\oplus(1.0, 0.1) = 1.0$ .

Note that the elements **emacs**, **vi**, **word** and **wright** appearing in the final line of the above formula, are not atomic domain values. They are level-1 fuzzy sets as depicted in Table II. That is, they can be rewritten as  $\{1.0/\text{emacs}\}$ ,  $\{1.0/\text{vi}\}$ ,  $\{1.0/\text{word}\}$  and  $\{1.0/\text{wright}\}$ . If we downgrade again the obtained level-2 fuzzy set **engineering** to level-1, we have the same looking fuzzy set such as  $\{1.0/\text{emacs}, 1.0/\text{vi}, 1.0/\text{word}, 0.8/\text{wright}\}$ . But now, the elements are atomic domain values. The reason why we treat the terminal nodes in a fuzzy ISA hierarchy as level-1 singleton sets rather than just atomic domain values, is in order to achieve a unified representation of domain concepts and domain values. As a result of this first step, we have a collection of level-1 fuzzy sets as in Table III.

In the second step, we compare the obtained level-1 fuzzy sets in a pairwise manner to probe the inclusion relationships. Finally, we draw arrows between fuzzy sets and their

TABLE III

LEVEL-1 FUZZY SETS OBTAINED THROUGH SUPPORT FUZZIFICATION FROM LEVEL- $k$  ( $k > 1$ ) FUZZY SETS: THE LEFTMOST COLUMN ENUMERATES ATOMIC DOMAIN VALUES AND OTHER COLUMNS REPRESENT MEMBERSHIP DEGREES OF ATOMIC DOMAIN VALUES FOR THE FUZZY SETS LISTED IN THE FIRST ROW.

value	editor	docu	spread	engi	busi	$\omega$
emacs	1.0	0.1	0.0	1.0	0.1	1.0
vi	1.0	0.3	0.0	1.0	0.3	1.0
word	0.0	1.0	0.1	1.0	1.0	1.0
wright	0.0	0.0	1.0	0.8	1.0	1.0

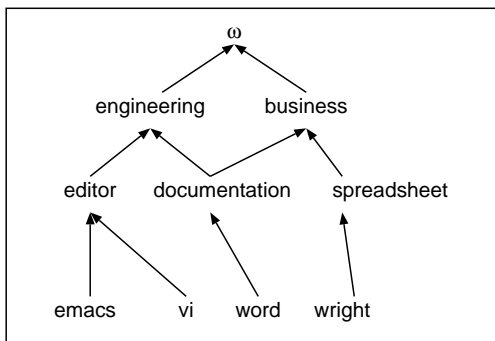


Fig. 3. The fuzzy set hierarchy derived from a fuzzy ISA hierarchy

direct subsets based on the obtained partial order relation. Figure 3 depicts the fuzzy set hierarchy obtained.

#### IV. A SUMMARY DISCOVERY PROCESS

Now we present a process to discover qualified generalized tuples based on given fuzzy domain knowledge, i.e., fuzzy set hierarchies. In short, our summary discovery process looks for qualified generalized tuples in a top-down manner. It initially hypothesizes the most generalized tuple, i.e., a generalized tuple whose fuzzy sets are all  $\omega$ 's. The process specializes the most generalized tuple, i.e.,  $\langle \omega, \dots, \omega \rangle$ , based on the given fuzzy set hierarchies to search for more specific generalized tuples while remaining qualified. The specialization is done *minimally* in the sense that only *one* fuzzy set is specialized into its *direct* subset. By evaluating support degrees of those specializations with respect to a given collection of database tuples, qualified generalized tuples are identified. At this point, human users can interact with the discovery process. They might choose only some qualified generalized tuples for the further consideration if they are not interested in the others or want to trade in the search completeness for reducing the search cost. The process minimally specializes again only user-chosen qualified generalized tuples. Those specializations become hypotheses for the next phase. After some repetitive steps of such specialization, the process yields a specialization hierarchy of qualified generalized tuples i.e., significant database summaries. Figure 4 diagrams the process, and Figure 5 depicts the steps in detail.

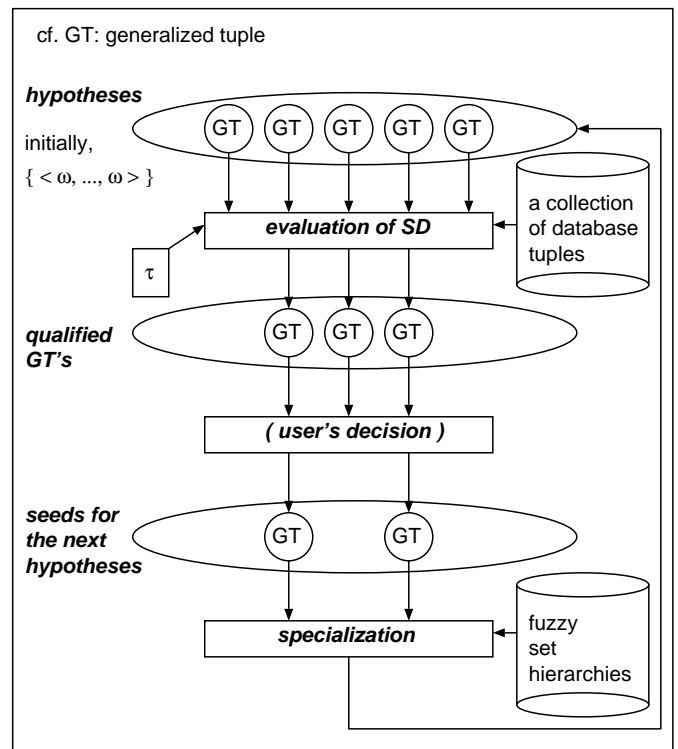


Fig. 4. A brief diagram of the summary discovery process

Note that the monotonicity property of the support degree in Theorem 1 guarantees that any specializations of *unqualified* generalized tuples cannot be qualified, and as a result, the process never misses qualified generalized tuples, even though it does not specialize unqualified generalized tuples.

In Figure 5, the support degree of each generalized tuple is computed from Line 5 to Line 7. This is the most time-consuming part of the process except the user interaction (Line 12), since the process must scan disks to read each database tuple. In Line 9, qualified generalized tuples with respect to  $\tau$ , are identified and they are put into *result*. After users choose only interesting generalized tuples in Line 12, they are minimally specialized in the sub-function `specialize()`. The process comes back to Line 4 with those specializations to repeat the steps.

Let us consider the efficiency of the process in terms of the number of disk accesses. Because it is hard to estimate how much time it takes to interact with human users, let us assume that users choose all qualified generalized tuples in Line 12 for the efficiency analysis. It is common to analyze the efficiency of most disk-based database applications in terms of disk access costs [11]. It is because the cost of disk accesses is much more expensive than that of in-memory operations. The disk access cost actually determines the efficiency of a process if the number of in-memory operations does not increase exponentially along with the input size, and no external network-based communication is involved.

The number of generalized tuples whose support degrees are evaluated in one specialization phase, i.e., the size of `curr` in Line 6, has nothing to do with the number

Input:  
 (i) a collection of database tuples  $C$ , (ii) fuzzy set hierarchies for attributes, (iii) a support degree threshold value  $\tau$

Output:  
 a specialization hierarchy of qualified generalized tuples

```

(1) SPGT()
(2) {
(3)   result =  $\phi$ ; curr = {  $\langle \omega, \dots, \omega \rangle$  };
(4)   while(curr  $\neq \phi$ ) {
(5)     foreach  $t$  in  $C$ 
(6)       foreach  $g$  in curr
(7)         accumulate  $SD(g)$  ;
(8)     foreach  $g$  in curr
(9)       if  $SD(g) > \tau$  then result = result  $\cup g$ ;
(10)      else curr = curr -  $g$ ;
(11)     foreach  $g$  in curr {
(12)       if the USER marks  $g$  then specialize(curr,  $g$ );
(13)       curr = curr -  $g$ ;
(14)     }
(15)   }
(16) }

(17) specialize(set,  $g = \langle a_1, \dots, a_m \rangle$ )
(18) {
(19)   for  $i = 1$  to  $m$ 
(20)     foreach direct subset  $sa_j$  of  $a_j$ 
      (in the fuzzy set hierarchy
      for the  $j$ th attribute)
(21)     set = set  $\cup \langle a_1, \dots, sa_j, \dots, a_m \rangle$ ;
(22) }

```

Fig. 5. A Specialization Process of Generalized Tuples (SPGT)

of database tuples. Rather, it is determined by the average fan-out of given fuzzy set hierarchies. Empirically, we expect that the system memory buffer space can hold all those generalized tuples in one specialization phase.

Then, the total number of disk accesses is *the number of specialization phases multiplied by the number of disk accesses to read database tuples in  $C$* . Let us denote the maximal path length of a fuzzy set hierarchy on the  $j$ th attribute as  $l_j$ . Since the specialization of generalized tuples are done attribute by attribute (See line 16 to 21), the number of specialization phases cannot be greater than  $\sum_j (l_j)$ . As a result, the number of disk accesses is no greater than  $\sum_j (l_j) \times p$ , where  $p$  denotes the number of disk pages containing the collection  $C$  of database tuples. Note that like the size of curr,  $\sum_j (l_j)$  is also determined by given fuzzy set hierarchies not by the number of database tuples. Thus, we claim that the average cost of our summary discovery process increases linearly along with the number of database tuples in a collection if we ignore human user's interaction.

#### A. An example of summary discovery

Suppose that we have a collection of computer usage records whose attributes are PROGRAM and USER as shown in Table IV. Given fuzzy ISA hierarchies on attributes, PROGRAM and USER, are supposed to be resolved to fuzzy sets in Table V and fuzzy set hierarchies in Figure 6. Fuzzy sets in Table V are represented in the form of semantic relations [12]. *Semantic relations* represent several fuzzy sets on the same domain in the relational form. If the domain is a continuous interval, a semantic rela-

TABLE IV  
 AN EXAMPLE COLLECTION OF COMPUTER USAGE RECORDS

PROGRAM	USER	PROGRAM	USER
emacs	John	emacs	Tom
vi	Tom	gcc	John
emacs	Tom	wright	Steve
vi	John	emacs	Jane
word	Kimberly	emacs	Mary
emacs	John	tetris	John
word	Mary	emacs	Jane
emacs	John	ultima	Mary
word	Kimberly	emacs	John
word	Kimberly	emacs	John
emacs	John	ultima	Mary
word	Mary	emacs	Jane
emacs	John	tetris	John
word	Kimberly	emacs	Mary
vi	John	emacs	Jane
emacs	Tom	wright	Steve
vi	Tom	gcc	John
emacs	John	emacs	Tom

TABLE V  
 SEMANTIC RELATIONS REPRESENTING FUZZY SETS IN THE FUZZY SET HIERARCHIES

For PROGRAM<sub>01</sub>

value	compiler	editor	docu	spread	engt	busi	game	$\omega$
gcc	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
cc	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
f77	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
emacs	0.0	1.0	0.1	0.0	1.0	0.1	0.0	1.0
vi	0.0	1.0	0.3	0.0	1.0	0.3	0.0	1.0
word	0.0	0.0	1.0	0.1	1.0	1.0	0.0	1.0
wright	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
tetris	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
ultima	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

For USER<sub>01</sub>

value	prog	writer	seller	account	develop	market	$\omega$
John	1.0	0.0	0.0	0.0	1.0	0.0	1.0
Tom	1.0	0.0	0.0	0.0	1.0	0.0	1.0
Mary	0.2	0.8	0.0	0.0	1.0	0.8	1.0
Kimberly	0.0	1.0	0.1	0.0	1.0	1.0	1.0
Steve	0.0	0.0	1.0	0.0	0.0	1.0	1.0
Jane	0.0	0.0	0.4	1.0	0.0	1.0	1.0
Bob	0.0	0.0	0.0	1.0	0.0	1.0	1.0

tion partitions the domain into disjoint sub-intervals and assigns a representative membership degree to each sub-interval. Semantic relations are treated in the same way as ordinary database tables, and as a result, we do not have to deviate from the framework of conventional data models. Even though we adopt and recommend semantic relations as a proper representation of fuzzy sets for the benefit of homogeneity, our discovery process is not tied to a specific fuzzy set representation in principle.

Figures 7 to 10 shows the behavior of the summary discovery process *SPGT* along with the specialization phases. The threshold value of support degrees is given as 0.4. Each figure corresponds to a single while loop in Figure 5. In the first specialization, among six generalized tuples derivable from the root generalized tuple, only three depicted in Figure 7 are qualified. If users are interested in the computer usage of *developers*, they would mark the middle one, i.e.,  $\langle -, \text{developer} \rangle$ , for further consideration. By specializing the marked one and evaluating support degrees of derived

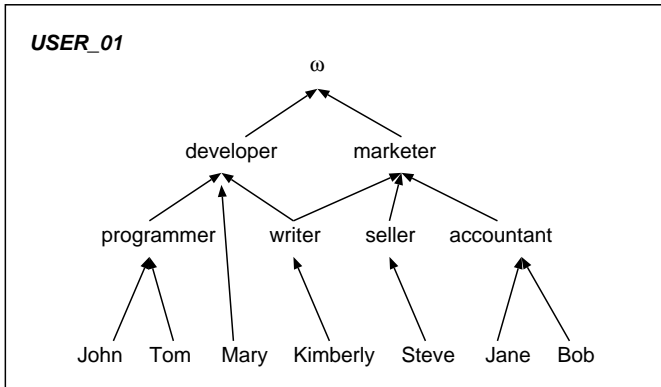
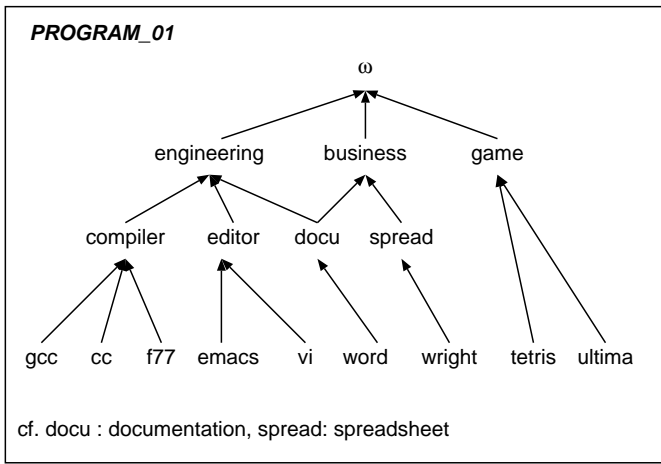


Fig. 6. Fuzzy set hierarchies for PROGRAM and USER. As different users have different ISA relationships between domain concepts in mind, there can be several fuzzy set hierarchies for an attribute domain. Thus, we postfix ‘\_01’ to each fuzzy set hierarchy name to denote that it is chosen among several available ones.

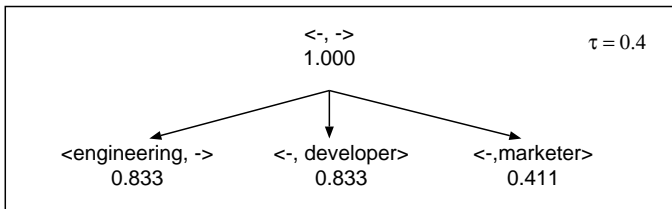


Fig. 7. The 1st specialization of the root generalized tuple

hypotheses, the process yields qualified generalized tuples as shown in Figure 8. Figures 9 and 10 show subsequent specializations.

From the final hierarchy of generalized tuples in Figure 7, we can conclude that developers used engineering programs mostly. In particular, programmers have been heavily executed editor programs.

V. INFORMATIVENESS OF GENERALIZED TUPLES

This section derives a measure for informativeness of generalized tuples based on Shannon’s information theory[7]. Though the proposed summary discovery process comes up with several qualified generalized tuples, the quantity of information we can obtain from each generalized tuple may be different. In the previous example, <editor,

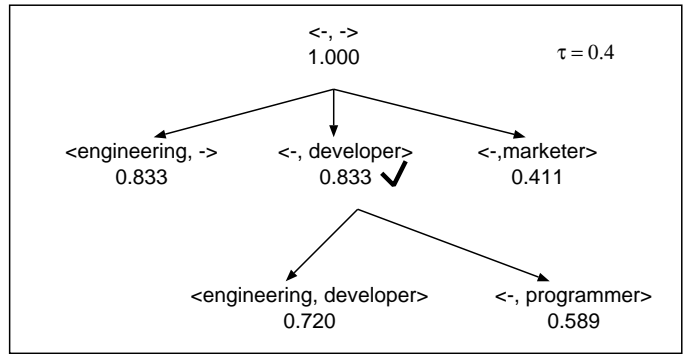


Fig. 8. The 2nd specialization of generalized tuples

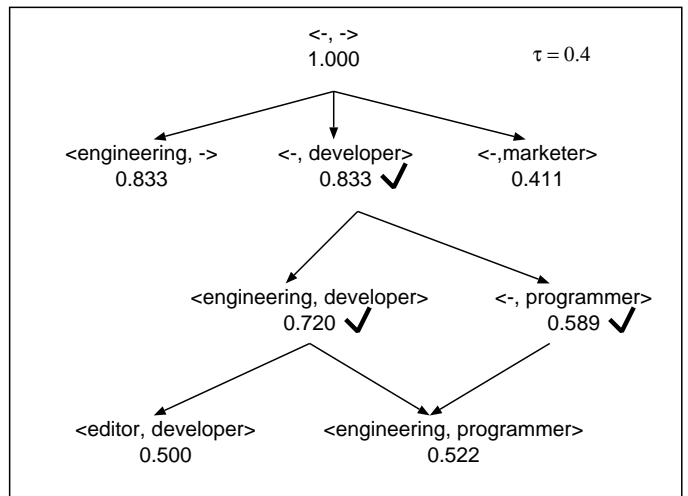


Fig. 9. The 3rd specialization of generalized tuples

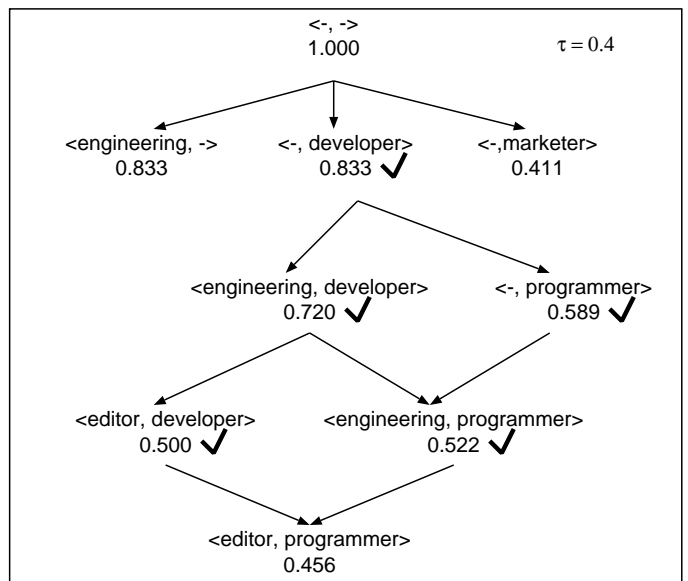


Fig. 10. The final hierarchy of generalized tuples

`programmer`> seems more informative than `<engineering, programmer>`. The reason is that since a fuzzy set `editor` is more specific than a fuzzy set `engineering`, we have less uncertainty to figure out the original collection of database tuples from `<editor, programmer>` than from `<engineering, programmer>`.

Along with the specificity of generalized tuples, support degrees also affect the information values. For example, `<editor, programmer>` with the support degree 0.9 could be regarded more informative than the same generalized tuple with the support degree 0.3. It is because the former seems to give information about 90% of the original collection of database tuples, while the latter seems to give information about only 30% of them<sup>3</sup>

### A. The notion of Shannon's entropy

In Shannon's information theory, the amount of information carried by a message is measured as *the amount of uncertainty* reduced by the existence or the occurrence of that message[7]. When there are  $n$  possible alternatives in the system on hand, Shannon's entropy measures the amount of uncertainty of the system as  $\log_2 n$ . If a message arrives, the number of alternatives might be reduced to  $m$  ( $m \leq n$ ) owing to the information carried by the message. Then the amount of uncertainty is reduced to  $\log_2 m$ . In this situation, we say that the message reduces uncertainty, in other words, it delivers information as much as  $\log_2 n - \log_2 m = \log_2 n/m$ . This notion of Shannon's entropy is adopted to analyze how much information content a generalized tuple delivers.

Let us denote the set of all possible data collections on a given attribute scheme, as  $\Omega(\cdot)$ , and a set of data collections on the same attribute scheme that make it possible for a qualified generalized tuple  $g$  to be discovered, as  $\Omega(g)$  ( $\Omega(g) \subseteq \Omega(\cdot)$ ). Then, the amount of the information that the generalized tuple  $g$  delivers, is  $\log_2 \frac{|\Omega(\cdot)|}{|\Omega(g)|}$ , where  $|A|$  denotes the cardinality of a set  $A$ .

### B. An informativeness measure for generalized tuples

Prior to deriving formulae for  $|\Omega(\cdot)|$ ,  $|\Omega(g)|$ , and in turn  $\log_2 \frac{|\Omega(\cdot)|}{|\Omega(g)|}$ , let us observe some characteristics of database summarization.

Firstly, a data collection  $C$  for summarization can have *duplicates*. It holds even for relational database systems whose underlying data model represents a data collection as a *relation*. To justify this argument, let us consider an example. Suppose that a computer log relation has an attribute scheme such as `<PROGRAM, USER, TTY, DURATION, START_TIME>` as in the `/var/adm/pacct` file in the UNIX system and users want to obtain summaries on which users have executed which computer programs. They have to project the relation onto those two attributes `PROGRAM` and `USER` before summarization. If duplicate records are eliminated after projection, unfit results may be obtained

<sup>3</sup>Strictly speaking, this argument alone may mislead. Both the specificity and the support degree should be considered simultaneously to obtain the more exact information value as detailed in the following sections.

since several number of executions of a program by the same user look like just one execution in the duplicate-eliminated table. Most of current relational database languages such as SQL permit duplicates in data tables[13].

Secondly, we assume that attribute dependencies such as functional dependencies and multivalued dependencies, are not known in advance. Finally, we suppose that the cardinality or the range of each attribute domain is known. Note that the *support set*[4] of the fuzzy set  $\omega$  on an attribute domain is equivalent to the attribute domain itself.

Now, we explain how to obtain  $|\Omega(\cdot)|$  and  $|\Omega(g)|$  in our context. Suppose that a given data collection  $C$  has an attribute scheme  $(A_1, \dots, A_m)$ . Let  $\Psi[j]$  denote the domain of the  $j$ th attribute. Then, the set of all possible tuples that can be composed from the domains, denoted as  $\Psi$ , becomes  $\Psi[1] \times \dots \times \Psi[m]$ , under the ignorance assumption of attribute dependencies. Consequently,

$$|\Omega(\cdot)| = |\Psi|^{|C|}, \dots \dots \dots (1.1)$$

since duplicates in the data collection  $C$  are allowed.

With respect to a generalized tuple  $g$ , the given data collection  $C$  can be thought to being divided into two parts, denoted as  $C_g$  and  $C_{\bar{g}}$ .  $C_g$  is a set of database tuples supporting  $g$ , and  $C_{\bar{g}}$  is its complement. Recall that the support degree implies the fraction of supporting database tuples to the total data collection, i.e.,  $\frac{|C_g|}{|C|}$ . Thus,

$$|C_g| = |C|SD(g), |C_{\bar{g}}| = |C|(1 - SD(g)) \dots (1.2)$$

Also,  $\Psi$  can be thought as being divided into two parts, denoted as  $\Psi_g$  and  $\Psi_{\bar{g}}$ , with respect to  $g$ .  $\Psi_g$  is a set of tuples consistent with a generalized tuple  $g$ , and  $\Psi_{\bar{g}} = \Psi - \Psi_g$ . If we define the *coverage degree*,  $CV$ , of a generalized tuple  $g$  as  $\frac{|\Psi_g|}{|\Psi|}$  to measure the fraction of  $\Psi$  that is consistent with  $g$ ,  $\Psi_g$  and  $\Psi_{\bar{g}}$  can be written as,

$$|\Psi_g| = |\Psi|CV(g), |\Psi_{\bar{g}}| = |\Psi|(1 - CV(g)) \dots (1.3)$$

The coverage degree will be precisely defined in the next subsection. At the moment, let us assume that it is given from somewhere. It is obvious that the coverage degree of a generalized tuple is, by definition, greater than zero. Complete coverage degree, i.e., one, of a generalized tuple implies that the generalized tuple is `<  $\omega, \dots, \omega$  >`. By definition,  $\Omega(\langle \omega, \dots, \omega \rangle)$  is the same as  $\Omega(\cdot)$ .

As depicted in Figure 11, database tuples in  $C_g$  and  $C_{\bar{g}}$  are thought to be selected from  $\Psi_g$  and  $\Psi_{\bar{g}}$ , respectively. Thus, we can formulate  $|\Omega(g)|$  by (1.2) and (1.3) as follows:

$$|\Omega(g)| = |\Psi_g|^{|C_g|} |\Psi_{\bar{g}}|^{|C_{\bar{g}}|} \\ = [|\Psi|CV(g)]^{|C|SD(g)} [|\Psi|(1 - CV(g))]^{|C|(1 - SD(g))}, (1.4)$$

when  $0 < CV(g) < 1$ .

If  $CV(g) = 1$  then  $|\Omega(g)| = \Omega(\cdot)$ .

As a result, the information content of a generalized tuple  $g$  is formulated as follows:

$$INFO(g) = \log_2 \frac{|\Omega(\cdot)|}{|\Omega(g)|} \\ = \log_2 \frac{|\Psi|^{|C|}}{[|\Psi|CV(g)]^{|C|SD(g)} [|\Psi|(1 - CV(g))]^{|C|(1 - SD(g))}}, \\ \text{by (1.1) and (1.4)} \\ = \log_2 \frac{1}{CV(g)^{|C|SD(g)}(1 - CV(g))^{|C|(1 - SD(g))}} \dots \dots (1.5)$$



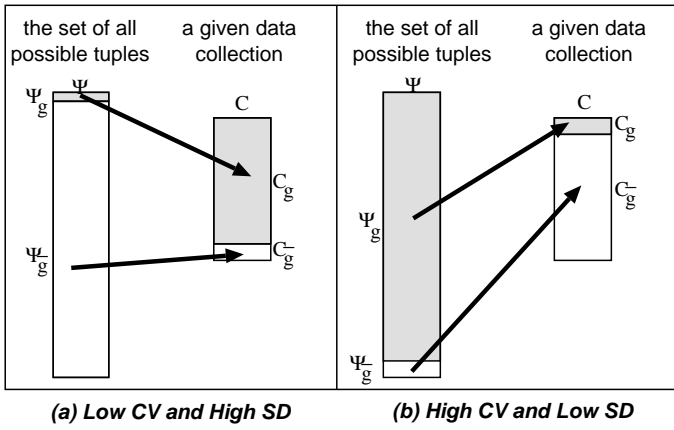


Fig. 11. Tuple selections when a qualified generalized tuple  $g$  is known. (a) denotes a case where the coverage degree of  $g$  is very low but the support degree is high and, (b) denotes the opposite case.

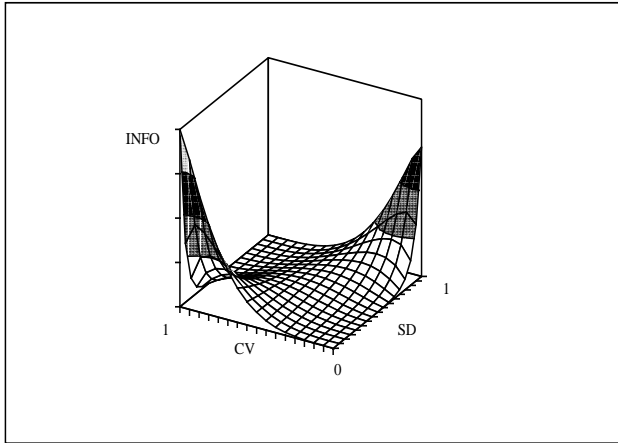


Fig. 12. Informativeness for various SD-CV combinations

when  $0 < CV(g) < 1$ .

If  $CV(g) = 1$  then  $INFO(g) = \log_2 1 = 0$ .

Let us observe the behavior of  $INFO$ . Figure 12 depicts the values of  $INFO$  with respect to  $SD$  and  $CV$ . There are two areas where informativeness becomes high. One area includes cases where  $CV$  is low and  $SD$  is high (See the right peak in Figure 12). It shows that a *specific* generalized tuple with a high support degree delivers much information. It is because the fact of low  $CV$  and high  $SD$  implies that most records (i.e.,  $|C|SD(g)$ ) in the original data collection  $C$  come from a small number of alternatives covered by the generalized tuple (i.e.,  $|\Psi|CV(g)$ ): See Figure 11-(a).

The other includes cases where  $CV$  is high and  $SD$  is low (See the left peak in Figure 12). It shows that though the support degree is low, a generalized tuple can deliver much information if it is very non-specific. As Figure 11-(b) shows, it is also informative since the fact of high  $CV$  and low  $SD$  implies that most records (i.e.,  $|C|(1-SD(g))$ ) in the data collection  $C$  come from a small number of alternatives

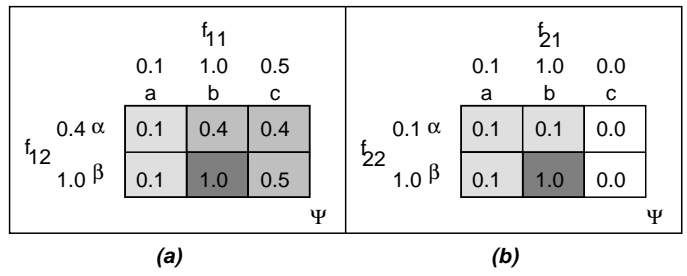


Fig. 13. Coverage of generalized tuples. (a) is for  $g_1 = \langle f_{11}, f_{12} \rangle$ , where  $f_{11} = \{0.1/a, 1.0/b, 0.5/c\}$ , and  $f_{12} = \{0.4/\alpha, 1.0/\beta\}$ . (b) is for  $g_2 = \langle f_{21}, f_{22} \rangle$ , where  $f_{21} = \{0.1/a, 1.0/b, 0.0/c\}$ , and  $f_{22} = \{0.1/\alpha, 1.0/\beta\}$ . In each diagram, the whole six-block area represents  $\Psi$ , and the shaded area represents  $\Psi_{g_1}$  (or  $\Psi_{g_2}$ ). The density of shade implies how completely the generalized tuple covers the corresponding area.

not covered by the generalized tuple (i.e.,  $|\Psi|(1 - CV(g))$ ).

C. The coverage degrees of generalized tuples

So far, we have assumed that the coverage degree of a generalized tuple is given from somewhere. Now let us consider how to actually obtain the coverage degree of a generalized tuple  $g$ .

*Definition 4:* The coverage degree of a generalized tuple  $g = \langle f_1, \dots, f_m \rangle$  on an attribute scheme  $(A_1, \dots, A_m)$  is defined as follows:

$$CV(g) = \frac{\sum_{x_1 \in \Psi_1, \dots, x_m \in \Psi_m} \otimes(\mu_{f_1}(x_1), \dots, \mu_{f_m}(x_m))}{|\Psi_1| \times \dots \times |\Psi_m|}$$

Note that by substituting  $\sum$  operations by  $\int$  operations, the formula can be adapted to the case where continuous domains of attributes are involved. Let us see an example of the coverage degree computation: Suppose that we are considering two generalized tuples on an attribute scheme having two attributes, and attribute domains are  $\{a, b, c\}$  and  $\{\alpha, \beta\}$  for the first and second attributes. Those two generalized tuples are given as:

$$\begin{aligned} g_1 &= \langle f_{11}, f_{12} \rangle, \text{ where} \\ f_{11} &= \{0.1/a, 1.0/b, 0.5/c\}, f_{12} = \{0.4/\alpha, 1.0/\beta\}, \\ g_2 &= \langle f_{21}, f_{22} \rangle, \text{ where} \\ f_{21} &= \{0.1/a, 1.0/b, 0.0/c\}, f_{22} = \{0.1/\alpha, 1.0/\beta\}. \end{aligned}$$

Then the coverage degrees of those generalized tuples are computed as follows:

$$\begin{aligned} |\Psi_1| \times |\Psi_2| &= |\{a, b, c\}| \times |\{\alpha, \beta\}| = 3 \times 2 = 6, \quad (2.1) \\ \sum_{x_1 \in \Psi_1, x_2 \in \Psi_2} \otimes(\mu_{f_{11}}(x_1), \mu_{f_{12}}(x_2)) &= \otimes(\mu_{f_{11}}(a), \mu_{f_{12}}(\alpha)) + \otimes(\mu_{f_{11}}(a), \mu_{f_{12}}(\beta)) \\ &+ \otimes(\mu_{f_{11}}(b), \mu_{f_{12}}(\alpha)) + \otimes(\mu_{f_{11}}(b), \mu_{f_{12}}(\beta)) \\ &+ \otimes(\mu_{f_{11}}(c), \mu_{f_{12}}(\alpha)) + \otimes(\mu_{f_{11}}(c), \mu_{f_{12}}(\beta)) \\ &= \otimes(0.1, 0.4) + \otimes(0.1, 1.0) + \otimes(1.0, 0.4) \\ &+ \otimes(1.0, 1.0) + \otimes(0.5, 0.4) + \otimes(0.5, 1.0) \\ &= 0.1 + 0.1 + 0.4 + 1.0 + 0.4 + 0.5 \\ &= 2.5 \dots \dots \dots (2.2) \end{aligned}$$

$$\begin{aligned} \text{Similarly,} \\ \sum_{x_1 \in \Psi_1, x_2 \in \Psi_2} \otimes(\mu_{f_{21}}(x_1), \mu_{f_{22}}(x_2)) &= 1.3 \dots (2.3) \\ \text{By (2.1) and (2.2), } CV(g_1) &= 2.5/6 = 0.42, \text{ and} \\ \text{by (2.1) and (2.3), } CV(g_2) &= 1.3/6 = 0.22. \end{aligned}$$

As a result, we can say that  $g_1$  and  $g_2$  cover the  $\Psi$  as much as 42% and 22%, respectively. Figure 13 depicts how those generalized tuples cover  $\Psi$  graphically.

## VI. CONCLUDING REMARKS

## A. Summary

In this paper, we have proposed an interactive top-down summary discovery process which utilizes fuzzy ISA hierarchies as the domain knowledge. We have defined a *generalized tuple* as a representational form of a database summary including fuzzy concepts. By virtue of fuzzy ISA hierarchies where fuzzy ISA relationships in the actual domain are naturally expressed, the discovery process yields more accurate database summaries. Rather than using fuzzy ISA hierarchies directly, we provide a method to resolve a given fuzzy ISA hierarchy into a collection of fuzzy sets defined on the same domain, and a fuzzy set hierarchy. Fuzzy set hierarchies make it possible to prune unnecessary hypothesis derivations without missing any potentially qualified generalized tuples. We have also presented an informativeness measure for distinguishing generalized tuples that deliver much information to users, based on Shannon's information theory.

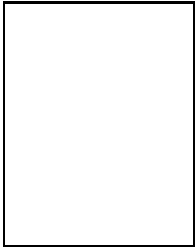
In the classification of inductive learning paradigms, this work belongs to the *model-driven generate-and-test* approach. Fuzzy set hierarchies given as the domain knowledge *guide* the learning process. The process *generates* hypothetical summaries and *tests* their validity over the given instance space. Thus, the process shares the well-known advantages of the model-driven approach against the data-driven one. One of them is *noise robustness*, i.e., a few exceptional training instances do not affect the major results of the process. Since the process is done in a top-down manner unlike the bottom-up method proposed in [5], users can easily control the selection of search paths as well as the search depth. In actual database applications, where a great number of possible search paths exist, such kind of path control is quite necessary.

## B. Future Research

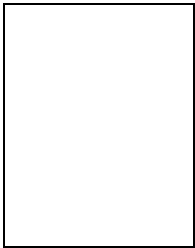
There remain some open issues to be addressed. First of all, systematic procedures to elicit domain knowledge such as fuzzy ISA hierarchies, should be conceived for practical applications, even though term thesauruses for information retrieval systems could be adapted in some cases. In addition, since discovered summaries themselves can be considered as useful information to be stored and managed, methods to maintain consistency between the raw database and the derived summary data set should be addressed to cope with changes of the original raw database.

## REFERENCES

- [1] W. Frawley, G. Piatetsky-Shapiro and C. Matheus, "Knowledge Discovery in Databases: An Overview", *Knowledge Discovery in Databases*, AAAI Press, 1991, pp.1-27
- [2] R. Yager, "On Linguistic Summaries of Data", *Knowledge Discovery in Databases*, AAAI Press, 1991, pp. 347-363
- [3] L. Zadeh, "Fuzzy Sets", *Information and Control*, Vol. 8, pp. 338-353, 1965
- [4] H. Zimmermann, *Fuzzy Set Theory and Its Applications*, Kluwer-Nijhoff Pub., 1985
- [5] J. Han, Y. Cai and N. Cercone, "Knowledge Discovery in Databases: An Attribute-Oriented Approach", in *Proc. the 18th VLDB Conference*, 1992, pp. 547-559
- [6] D.H. Lee and M.H. Kim, "Discovering Database Summaries through Refinements of Fuzzy Hypotheses", in *Proc. 10th Int'l Conf. on Data Engineering*, 1994, pp. 223-230
- [7] C.Shannon, "The Mathematical Theory of Communication", *The Bell System Tech. J.*, Vol. 27, pp. 379-423, 623-656, 1948
- [8] P.Cohen and E.Feigenbaum, *The Handbook of Artificial Intelligence*, Vol.3, William Kaufmann Pub., 1982, pp. 411-415
- [9] G.J. Klir and T.A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall Int'l, Inc., 1988, pp. 260-265
- [10] L. Zadeh "A Fuzzy Set Theoretic Interpretation of Linguistic Hedges", *Journal of Cybernetics*, Vol. 2, No. 2, pp. 4-34, 1972
- [11] J. Ullman, *Principles of Databases and Knowledge-Base Systems*, Computer Science Press, 1988
- [12] D.H. Lee and M.H. Kim, "Accommodating Subjective Vagueness through a Fuzzy Extension to the Relational Data Model", *Information Systems*, Vol. 18, No. 6, pp. 363-374, 1993
- [13] ISO 9075 : Information Processing Systems - Database Language SQL, 1992



**Do Heon LEE** received the B.S., M.S., and Ph.D. degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST), Taejeon, Korea, in 1990, 1992, and 1995, respectively. Since 1991, he has worked on combining AI and DB technology. His current primary research interests are in the areas of data mining and fuzzy database.



**Myoung Ho KIM** received the B.S. and M.S. degrees in computer engineering from Seoul National University, Seoul, Korea, in 1982 and 1984, respectively, and the Ph.D. degree in computer science from Michigan State University, East Lansing, MI, in 1989. In 1989 he joined the faculty of the Department of Computer Science at KAIST, Taejeon, Korea, where he currently is an associate professor. His research interests include distributed and parallel database, data mining, multidatabase, information retrieval and real-time database.

Dr.Kim is a member of the Association for Computing Machinery and IEEE Computer Society.