ELSEVIER

Data & Knowledge Engineering 20 (1996) 227–252

DATA & KNOWLEDGE ENGINEERING

A hierarchical model learning approach for refining and managing concept clusters discovered from databases

Ning Zhong[a,*], Setsuo Ohsuga[b]

[a] *Department of Computer Science and Systems Engineering, Faculty of Engineering, Yamaguchi University, 2557 Tokiwadai, Ube-Shi 755, Japan*

[b] *Department of Information and Computer Science, School of Science and Engineering, Waseda University, 3-4-1 Okubo Shinjuku-Ku, Tokyo 169, Japan*

Received 26 April 1995; revised 17 August 1995; accepted 26 December 1995

Abstract

The contents of most databases are ever-changing, and erroneous data can be a significant problem in real-world databases. Therefore, the process of discovering knowledge from databases is a process based on incipient hypothesis generation/evaluation and refinement/management. Although many systems for knowledge discovery in databases have been proposed, most systems have not addressed the capabilities of refining/managing the discovered knowledge. This paper describes a *hierarchical model learning* approach for refining/managing concept clusters discovered from databases. This approach is the basic one for developing HML (Hierarchical Model Learning), which is one sub-system of our GLS (Global Learning Scheme) discovery system and can be cooperatively used with other sub-systems of GLS such as DBI (Decomposition Based Induction). By means of HML, concept clusters discovered from a database by DBI can be represented as the Multi-Layer Logic formulae with hierarchical models in a knowledge-base and can be easily refined/managed according to data change in a database and/or domain knowledge. HML is based on the model representation of Multi-Layer Logic (MLL). Its key feature is the quantitative evaluation for selecting the best representation of the MLL formulae by using cooperatively a criterion based on information theory and domain knowledge. Experience with a prototype of HML implemented by the knowledge-based system KAUS is discussed.

Keywords: Knowledge discovery in databases; Multi-Layer Logic; Machine learning; Information theory; Hierarchical modeling; Refinement; Management

1. Introduction

Knowledge discovery in databases (KDD) is becoming an important topic in AI and is attracting the attention of leading researchers in databases [19, 7]. This topic is different from traditional researches of machine learning, though it uses their results [10]. In particular, the

* Corresponding author. Email: zhong@ai.csse.yamaguchi-u.ac.jp

SSDI: 0169-023X(96)00003-1

contents of most databases are ever-changing (i.e. data in databases can be often deleted, added or updated), and erroneous data can be a significant problem in real-world databases (i.e. data in databases are generally uncertain and incomplete) [7, 27]. Hence, the process of discovering knowledge from databases is a process based on incipient hypothesis generation/evaluation and refinement/management as shown in Fig. 1 [27]. In this process, it is required to perform multi-aspect intelligent data analysis and multi-level conceptual abstraction/learning in multiple learning phases [30]. Although many systems for knowledge discovery in databases such as INLEN, Forty-Niner, KDW, EXPLORA and DBLEARN have been proposed [9, 35, 18, 4, 3], most systems have not addressed the capabilities of refining/managing the discovered knowledge. For example, although there are knowledge management operators in INLEN, it is not automatically done and the operator for knowledge refinement was not addressed; Forty-Niner can refine regularity by expanding their range and/or strengthening their pattern, however it was not considered how to refine the discovered regularity when data change (e.g. add or delete some data) in databases, and it was also not addressed how to manage the discovered regularities; KDW, EXPLORA and DBLEARN did not address the capabilities of management and refinement.

We have been developing a methodology/system for knowledge discovery in databases, called GLS (Global Learning Scheme) based on this process as shown in Fig. 1 [27, 30]. The GLS system is developed as a toolkit that is composed of several sub-systems. At present, two sub-systems of GLS, DBI (Decomposition Based Induction) and KOSI (Knowledge Oriented Statistic Inference), have been developed for discovering incipient hypotheses from databases [29, 32], and two further sub-systems of GLS, HML (Hierarchical Model Learning) and IIBR (Inheritance Inference Based Refinement), have been developed for refining and managing incipient hypotheses discovered from databases [28, 31]. Furthermore, a discovery process as shown in Fig. 1 can be organized dynamically and performed in succession. For example, DBI can be first used for discovering concept clusters hidden in the data [29], and then by means of HML, the discovered concept clusters can be represented as the Multi-Layer Logic formulae with hierarchical models in a knowledge-base and can be easily refined and managed according to data change in a database and/or domain knowledge.

This paper describes a way of refining/managing concept clusters by using HML. In GLS, the refinement for concept clusters can be divided into two levels. The first one is the *data*

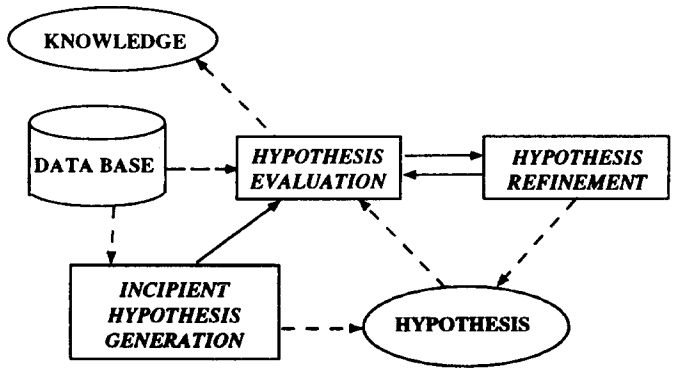

Fig. 1. The process of disxovering knowledge from databases.

level. In this level, concept clusters are refined by the *learning space* defined in DBI for processing the perturbation problem of databases. This has been discussed in our paper [29]. This paper focuses to discuss the other one, i.e. the *rule level* in which concept clusters are refined by using *hierarchical model learning* (HML). Furthermore, the results of the refinement by HML can be further used in the next learning phase for acquiring more high-level knowledge [34]. How to use the results of the refinement by HML for acquiring more high-level knowledge is not discussed here but is left as an independent problem to be discussed elsewhere, because it involves another sub-system of GLS that is being developed by us.

In the following sections, we will describe the details of HML. Section 2 describes main backgrounds on HML including a summary of the features of Multi-Layer Logic and a comparison to related work using information theory for machine learning. Section 3 introduces information theory into logical expression and describes an effective algorithm for calculating the information of the Multi-Layer Logic formula as the theoretical preparation for our application. Section 4 describes the approach of *hierarchical model learning* (*HML*). It mainly includes the knowledge generation, the knowledge refinement and the knowledge management. Finally, Section 5 gives a summary of the features of our approach and future research subjects.

2. Backgrounds on HML

There are two main backgrounds on HML. The first is the model representation of Multi-Layer Logic (MLL) with the hierarchical structure [17, 12]. The other is information theory. The key feature of HML is the quantitative evaluation for selecting the best representation of the MLL formulae by using cooperatively a criterion based on information theory and domain knowledge. This section describes the two backgrounds of HML.

2.1. Knowledge representation using MLL

MLL (Multi-Layer Logic) is a predicate logic with a syntax that allows some domain(s) of variable(s) to be the variable(s), which extends for MSL (Many-Sorted Logic) in the syntax [17, 12]. This extension in the syntax of MSL gives a great expressive capability for predicate logic involving data structure (set, hierarchy, power set, etc.), especially in manipulation of the hierarchical structure. Since HML is based on the model representation of MLL, we here give a summary of the features of MLL compared with first order logic as a preparation for further describing HML. The details on MLL refer to [17].

The summary can be divided into the following four aspects:

(1) *Structure description*. Structures can be described as *element-of*, *power-set-of*, *component-of* and *product-set-of* relations. Other complex operations can be represented as combinations of these primary operations. For example, let a polyhedron as shown in Fig. 2 be defined as a set of surfaces s_1, s_2, s_3, s_4, and let surfaces be defined by a set of the edge lines using following structure description:

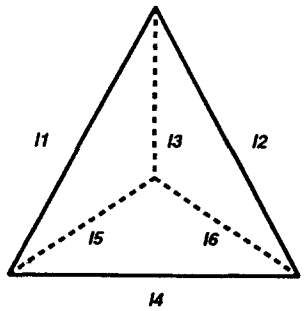

Fig. 2. A polyhedron.

/* Description of general concepts */
!make_p *2vertex, *2line, *2surface;
!ins_e *2vertex line;
!ins_e *2line surface;
!ins_e *2surface polyhedron;
/* Description of specific concepts */
!ins_e line $l_1, l_2, l_3, l_4, l_5, l_6$;
!ins_e surface s_1, s_2, s_3, s_4;
!ins_e polyhedron h_1;
/* Definition of component sets of a specific object with hierarchical structure */
!ins_e h_1 : *surface* s_1, s_2, s_3, s_4;
!ins_e s_1 : *line* l_1, l_2, l_4;
!ins_e s_2 : *line* l_1, l_3, l_5;
!ins_e s_3 : *line* l_2, l_3, l_6;
!ins_e s_4 : *line* l_4, l_5, l_6;

where "!*ins_e* **x** $x_1 \ldots x_n$;" means $x_1 \ldots x_n$ are elements of **x** (i.e. the set-elements relation). "*x" denotes a power set node whose base set is **x**. The base set of the power set is the one from which the extension of the power set is defined. In other words, a power set is composed of all subsets of the base set. However, MLL does not automatically enumerate all elements of the given power set from the given base set. "!*ins_e* *x" defines only parts of members of *x (i.e. a subset of x) by the arguments followed by !*ins_e* *x. Since *x is itself a set, *(*x) can also be defined in the same way, denoted by ***2x**. In general, ***nx** denotes the power set of ***(n − 1)x**. "!*ins_e* **x : a**" describes a component set of **x** (i.e. **a** is a discriminator of the component set). In addition, "!*make_p*" is used for declaring and making power set nodes.

In general, the structure description can be divided into three parts as shown above. That is, the description of general concepts, the description of specific concepts and the

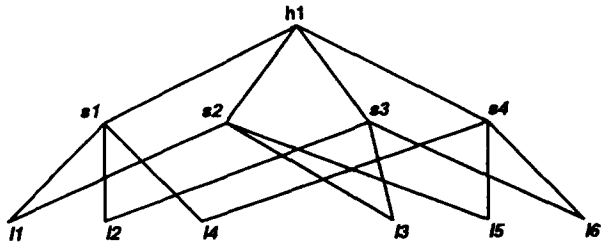

Fig. 3. The hierarchical structure of a polydedron.

definition of component sets (or IS-A relations) of a specific object with hierarchical structure. A component set can be regarded as an IS-A relation (i.e. pseudo IS-A). MLL prepares a syntax to discriminate the real IS-A relation and the pseudo IS-A relation. But it is abbreviated here (i.e. both of them are called the IS-A hierarchy in this paper). Fig. 3 is an equivalent graph of the component sets of this specific object shown above and Fig. 4 shows the relation of general concept in a data structure in MLL.

(2) *Syntax.* A MLL formula consists of a matrix, prefix, AND/OR forms, connectors *and* (&), *or* (|) and *not* (~). Similar to Many-Sorted Logic, a variable in a MLL formula can have its own domain and can be explicitly included in the prefix. For example, by means of the IS-A hierarchy defined above, we can represent the knowledge "There is some surface in a polyhedron h_1 of which the length of all edge lines is 3." in a MLL formula as follows:

$$[\exists S/h_1 : surface][\forall L/S : line]length(L\ 3)\ .$$

The part inside the brackets [ ] in the head of a logic formula is called the prefix in the MLL formula. Here, the domain of variable L is a variable S, the domain of S is h_1 for representing a specific polyhedron.

(3) *Expansion function.* When the domain set of a variable is finite, the MLL formula can be expanded according to the following equivalent expressions:

$$[\forall X/\mathbf{x}]p(X) \cap \boldsymbol{x} = \{x_1, x_2, \ldots, x_n\} \leftrightarrow p(x_1) \cap p(x_2) \cap \cdots \cap p(x_n)\ ,$$

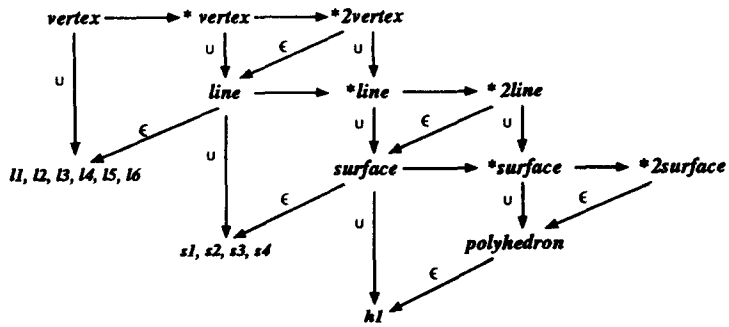

Fig. 4. The data structure for representing a polyhedron.

$$[\exists X/\mathbf{x}]p(X) \cap \boldsymbol{x} = \{x_1, x_2, \ldots, x_n\} \leftrightarrow p(x_1) \cup p(x_2) \cup \cdots \cup p(x_n) .$$

It is called *expansion function of MLL*. This function is used for extracting from a set the elements which possess specified properties. It is syntactically defined by appending "#" after the variable to be expanded in the prefix of the MLL formula. For example, let surfaces be $\{(l_1, l_2, l_4), (l_1, l_3, l_5), (l_2, l_3, l_6), (l_4, l_5, l_6)\}$, then the formula

$$[\exists S\#/h_1 : surface][\forall L\#/S : line]length(L3) .$$

can be expanded into

$$(length(l_1\ 3) \cap length(l_2\ 3) \cap length(l_4\ 3)) \cup$$
$$(length(l_1\ 3) \cap length(l_3\ 3) \cap length(l_5\ 3)) \cup$$
$$(length(l_2\ 3) \cap length(l_3\ 3) \cap length(l_6\ 3)) \cup$$
$$(length(l_4\ 3) \cap length(l_5\ 3) \cap length(l_6\ 3)) .$$

(4) *Higher-order predicate*. Predicate of MLL can include one or more closed formula(s) as a term. A closed formula is a formula which does not include any free variable. The predicate appears as a higher-order predicate but it is inhibited that the same variable is included both inside and outside of an inner predicate. Thus, for example, $[\forall X/d]p(X\ r(X))$ is not allowed but $[\forall X/d]p(X\ [\forall X/c]r(X))$ is because the latter is equivalent to $[\forall X/d]p(X\ [\forall Y/c]r(Y))$. With this restriction, the inner predicates can be any logical formula and the evaluation of the inner formula, r in the above example, can be performed independently from that of the outer predicate p. Thus, it is possible to separate the inner predicates and the outer predicates at the presentation. In general, the set of predicates that does not contain any predicate as a term are located in the object-level, while those that contain some object-level predicate(s) as the term are arranged in a different level immediately above to the object level. The upper level forms the meta-level. It is possible to replace the inner (object-level) predicate by an identifier (ID), for example, by the predicate name. The same holds between the meta-level and meta-meta-level and much higher levels. That is, this extension enables us to realize multiple meta-level architecture [5].

A prototype of HML has been implemented by KAUS. KAUS is a knowledge-based system developed in our laboratory which involves knowledge-bases based on MLL (Multi-Layer Logic) and databases based on the NNF (Non Normal Form) model [17, 26]. KAUS also has the capabilities of multiple meta level reasoning and multiple knowledge worlds. With these characteristics, KAUS enables us to write down the repetitive process as shown in Fig. 1, and can be easily used for many hypothesis generation (discovery/building) as well as representing, transforming and managing both knowledge and data [15, 16].

2.2. Applications of information theory in machine learning

In machine learning, information theory has been recognized as a useful criterion, and several algorithms such as ID3, Prism, CN2, ITRULE and EG2 have been developed [20, 1, 2, 23, 11]. In these algorithms, information theory is used as a measure criterion for

generating inductively knowledge that is represented in decision tree or *if-then* rule. For example, ID3 is a tree-induction algorithm although ID3-induced trees can be transformed into production rules [21]. Like ID3, Prism is classification based and has an information theoretic basis, but it can directly produces their results as a set of production rules. CN2 combines the efficiency and ability to cope with noisy data of ID3 with the if-then rule form (i.e. an ordered list of if-then rules) and flexible search strategy of the AQ family [2, 8]. ID3, Prism and CN2 assume that the training set is a complete one, i.e. they tend to produce only perfect rules. While ITRULE will find these rules, it also generates probabilistic rules. Furthermore, it is stressed that domain knowledge can be used in inductive learning. For example, EG2 can use domain knowledge in decision induction [11].

In these algorithms, however, information theory is only used for generating inductively knowledge that is represented in decision tree or if-then rule, but the issue on refinement/management of knowledge is not considered. Moreover, they are not used for evaluating the information of the logic formula and are not designed for performing multi-aspect intelligent data analysis and multi-level conceptual abstraction/learning in multiple learning phases.

HML can be considered as a typical approach in which information theory is used as a criterion for learning knowledge. An important different point of HML and ID3 including Prism, CN2, ITRULE and EG2 is that HML is designed for refining/managing concept clusters discovered from databases. It can evaluate quantitatively the amount of information of a MLL formula and select the best representation of the MLL formulae by using cooperatively a criterion based on information theory and domain knowledge. Another important different point is that HML is not an isolated algorithm. Its development is based on the GLS methodology [27, 30]. That is, HML is one sub-system of GLS which can be cooperatively used with other sub-systems of GLS such as DBI, and serves as a learning phase in multiple learning phases of GLS for generating, refining and managing concept clusters denoted by using the Multi-Layer Logic formulae with hierarchical models. Furthermore, the results of the refinement by HML can be further used in the next learning phase of GLS for acquiring more high-level knowledge [34].

3. Information of logical expression

This section introduces information theory into the Multi-Layer Logic (MLL) expression. We first define the information of the MLL expression, and then discuss three theorems for effectively calculating the MLL information. Finally, an effective algorithm for evaluating the MLL information is given.

3.1. Definition and theorems

Let us consider a predicate F and let d be a finite base set. For simplicity, we assume F being a single place predicate $F(x)$. It gives a description on an object in d. Or, in other words, $F(x)$ classifies all elements in the set d into two classes: those that satisfy $F(x)$ and those that do not. In the following, $F(x)$ and $\bar{F}(x)$ mean that "$F(x)$: True" and "$F(x)$: False", respectively, for $x \in d$. Let us define a concept "the state of d before and after the formula". It is

assumed that in the prior state, whether $F(x)$ or $\bar{F}(x)$ is not clear for any x in d, while, in the posterior state, either $F(x)$ or $\bar{F}(x)$ is made clear for some or all elements in d. Based on the preparation, we first define the information of MLL.

Definition 1. Information of MLL. Let $d = \{a_1, a_2, \ldots, a_N\}$ be a finite base set. The state of d is defined as the conjunctions of either $F(a_i)$ or $\bar{F}(a_i)$ for every element a_i in d. Before the formula is given, the state of d includes all possibilities of combinations of $F(a_i)$ and $\bar{F}(a_i)$, $i = 1, 2, \ldots, N$ such that $S_1 : \bar{F}(a_1) \wedge \bar{F}(a_2) \wedge \cdots \wedge \bar{F}(a_N)$ through $S_{2^N} : F(a_1) \wedge F(a_2) \wedge \cdots \wedge F(a_N)$. Let the set S_F^1 be defined as the collection of all possible prior states, and the set S_F^2 be defined as the collection of all possible posterior states. Thus, $S_F^1 = \{S_1, \ldots, S_{2^N}\}$. When the formula F is given, the states of some of elements are fixed. Then, S_F^2 becomes a subset of S_F^1 as shown in Fig. 5. Furthermore, let their cardinalities be $|S_F^1|$ and $|S_F^2|$, and their entropies be defined as $I_{S_{F1}} = \log|S_F^1|$ and $I_{S_{F2}} = \log|S_F^2|$, respectively. Thus, the amount of information of the MLL formula F can be defined in the following Eq. (1),

$$K = I_{SF1} - I_{SF2} = \log|S_F^1| - \log|S_F^2| . \tag{1}$$

That is, the difference of I_{SF1} and I_{SF2} is the amount of information K with respect to the predicate symbol F.

From Eq. (1), we can see that more information is obtained by decreasing the posterior entropy I_{SF2} of a MLL formula. Moreover, the above definition is easily extended to the case of n place predicate with n greater than one [12].

Example 1. Assume that we have an IS-A hierarchy shown in Section 2.1 and a MLL formula is given in the formal quantifiers Q_i,

$$[Q_1 S\#/h_1 : surface][Q_2 L\#/S : line]length(L\ 3) ,$$

where Q_i $(i = 1, 2)$ denotes either $\forall$ or $\exists$. Then, $d = \{l_1, l_2, \ldots, l_6\}$ and the different quantifiers in the prefix of this MLL formula have different amounts of information. Before the formula, the predicate $length(l_i\ 3)$ is either true or false for all possible states. Therefore, $|S_F^1| = 2^6$ and $\log|S_F^1| = 6$. Thus,

(1) $[\forall S\#/h_1 : surface][\forall L\#/S : line]length(L\ 3)$,
$K_{\forall\forall} = 6 - \log 1 = 6$;

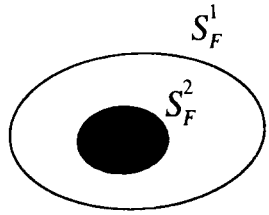

Fig. 5. The prior and the posterior states sets of MLL.

This formula says that all lines are of the length 3. That is, only $length(l_1 3) \wedge \cdots \wedge length(l_6 3)$ is allowed as a particular state. Thus, $|S_F^2| = 1$. In the following cases, the numbers of possible posterior states become 41 and 23, respectively.

(2) $[\forall S\#/h_1 : surface][\exists L\#/S : line]length(L\ 3)$,
$K_{\forall\exists} = 6 - \log 41 = 0.642$;

(3) $[\exists S\#/h_1 : surface][\forall L\#/S : line]length(L\ 3)$,
$K_{\exists\forall} = 6 - \log 23 = 1.476$;

The formula in case (4) says that there is some line of which the length is 3. That is, this formula allows every element in S_F^1 except $\overline{length}(l_1\ 3) \wedge \cdots \wedge \overline{length}(l_N\ 3)$. Thus, $|S_F^2| = 63$.

(4) $[\exists S\#/h_1 : surface][\exists L\#/S : line]length(L\ 3)$,
$K_{\exists\exists} = 6 - \log 63 = 0.023$;

where $K_{Q_1Q_2}$ denotes the amount of information of a MLL formula with quantifiers Q_1Q_2.

Example 1 shows that the MLL formulae with different quantifiers may reveal various different information even if the structure is the same. This is one of two aspects of evaluating the MLL information. Another aspect is to evaluate a MLL formula with different structures. This will be discussed in this section and Section 4.2.

Based on Definition 1, we discuss also three theorems in this section for effectively evaluating the information of the MLL formula.

Theorem 1. *Complement of the posterior cardinality of MLL. The posterior cardinalities of the MLL formulae with contrary quantifiers in their prefixes are complementary. That is*

$$|S_F^2|_{\forall\exists} = |S_F^1| - |S_F^2|_{\exists\forall}\ . \tag{2}$$

Proof. When the domain sets are finite and without loss of generality, let an IS-A hierarchy be defined as

!ins_e $*$d $d_1, d_2, \ldots, d_m$;
!ins_e d_1 $a_1, a_2, \ldots, a_l$;
!ins_e d_2 $a_1, a_3, a_5, \ldots, a_j$;
. . . . . .
!ins_e $d_m a_2, a_3, a_5, \ldots, a_k$;

where elements a_1, a_2, a_3, a_5 in $a_1, a_2, \ldots, a_n$ are "tangled" elements (i.e. one a_i can belong to more than one d_j). And let a MLL formula with the prefix $\forall\exists$ be

$$[\forall Y/*d][\exists X/Y]f(X)\ .$$

Let the probability $P([Q_1Y/*d][Q_2X/Y]f(X))$ be $P_{Q_1Q_2}$, and let $f(a_i)$ be simply represented as b_i. Since the prior probability is not known in advance, it is assumed that the probabilities

of $P(f(X))$ and $P(\sim f(X))$ are equiprobable. Then based on Definition 1, the posterior probability of this MLL formula is

$$P_{\forall\exists} = P((b_1 \cup b_2 \cup \cdots \cup b_l) \cap (b_1 \cup b_3 \cup b_5 \cup \cdots \cup b_j) \cap \cdots \cap (b_2 \cup b_3 \cup b_5 \cdots \cup b_k)) .$$

And let $P(\sim \textit{formula})$ be $\bar{P}(\textit{formula})$, then because

$$\begin{aligned}
\bar{P}_{\forall\exists} &= P(\overline{(b_1 \cup b_2 \cup \cdots \cup b_l) \cap (b_1 \cup b_3 \cup b_5 \cup \cdots \cup b_j)} \\
&\quad \overline{\cap \cdots \cap (b_2 \cup b_3 \cup b_5 \cdots \cup b_k)}) \\
&= P(\overline{b_1 \cup b_2 \cup \cdots \cup b_l} \cup \overline{b_1 \cup b_3 \cup b_5 \cup \cdots \cup b_j} \\
&\quad \cup \cdots \cup \overline{b_2 \cup b_3 \cup b_5 \cup \cdots \cup b_k}) \\
&= P((\bar{b}_1 \cup \bar{b}_2 \cap \cdots \cap \bar{b}_l) \cup (\bar{b}_1 \cap \bar{b}_3 \cap \bar{b}_5 \cap \cdots \cap \bar{b}_j) \\
&\quad \cup \cdots \cup (\bar{b}_2 \cap \bar{b}_3 \cap \bar{b}_5 \cap \cdots \cap \bar{b}_k)) \\
&= P((b_1 \cap b_2 \cap \cdots \cap b_l) \cup (b_1 \cap b_3 \cap b_5 \cap \cdots \cap b_j) \\
&\quad \cup \cdots \cup (b_2 \cap b_3 \cap b_5 \cap \cdots \cap b_k)) \\
&= P_{\exists\forall} , \qquad (3)
\end{aligned}$$

$$P_{\forall\exists} + \bar{P}_{\forall\exists} = P_{\forall\exists} + P_{\exists\forall} = 1 .$$

Furthermore, since the posterior cardinalities of the MLL formulae with the prefixes $\forall\exists$ and $\exists\forall$ are

$$|S_F^2|_{\forall\exists} = P_{\forall\exists} \times H , \qquad (4)$$

$$|S_F^2|_{\exists\forall} = P_{\exists\forall} \times H , \qquad (5)$$

where H is the maximal number of possible elements and H is also the prior cardinality (i.e. $|S_F^1| = H$),

$$\begin{aligned}
|S_F^2|_{\forall\exists} &= P_{\forall\exists} \times H = (1 - P_{\exists\forall}) \times H = H - P_{\exists\forall} \times H \\
&= |S_F^1| - |S_F^2|_{\exists\forall} . \quad \square
\end{aligned}$$

Moreover, by using the same method stated above, we can also prove

$$|S_F^2|_{\exists\exists} = |S_F^1| - |S_F^2|_{\forall\forall} .$$

Furthermore, since

$$\begin{aligned}
K &= \log|S_F^1| - \log|S_F^2| = \log H - \log P \times H \\
&= \log \frac{H}{P \times H} = \log \frac{1}{P} , \qquad (6)
\end{aligned}$$

and based on Theorem 1, we can calculate the information of the MLL formulae with quantifiers $\exists\forall$ and $\forall\exists$ by using

$$K_{\forall\exists} = \log \frac{1}{1 - P_{\exists\forall}} \tag{7}$$

and

$$K_{\exists\forall} = \log \frac{1}{P_{\exists\forall}} . \tag{8}$$

In order to calculate $P_{\exists\forall}$, the following Eq. (9) is used if there are "tangled" elements among sub-sets of the base set of an IS-A hierarchy (e.g. the calculation of Eq. (3)),

$$\begin{aligned} &P(e_1 \cup e_2 \cup \cdots \cup e_{n-1} \cup e_n) \\ &\quad = \sum_{i=1}^{n} P(e_i) - \sum_{1 \leq i < j \leq n} P(e_i \cap e_j) \\ &\qquad + \sum_{1 \leq i < j \leq n} P(e_i \cap e_j \cap e_k) - \cdots + (-1)^{n-1} P(e_1 \cap e_2 \cap \cdots \cap e_{n-1} \cap e_n) , \end{aligned} \tag{9}$$

else Eq. (10) is used,

$$\begin{aligned} &P(e_1 \cup e_2 \cup \cdots \cup e_{n-1} \cup e_n) \\ &\quad = 1 - \bar{P}(e_1 \cup e_2 \cup \cdots \cup e_{n-1} \cup e_n) \\ &\quad = 1 - P(\bar{e}_1 \cap \bar{e}_2 \cap \cdots \cap \bar{e}_{n-1} \cap \bar{e}_n) . \end{aligned} \tag{10}$$

From Eqs. (7) and (8), we can further see another aspect of evaluating the MLL information, i.e. a MLL formula with different structures may also reveal various different information. The example about this will be shown in Section 4.2.

Theorem 2. *The equivalence of information of MLL. In the case of the IS-A hierarchy, it is possible to refine a hierarchical structure by defining new intermediate nodes. Then the prefix sequence becomes longer. If the same quantifiers appeared in succession in the prefix of a MLL formula such as*

(1-1) $[QX^{n-1}/s][\exists X^{n-2}/X^{n-1}] \cdots [\exists X^2/X^3][\exists X^1/X^2] f(X^1)$,

(2-1) $[QX^{n-1}/s][\forall X^{n-2}/X^{n-1}] \cdots [\forall X^2/X^3][\forall X^1/X^2] f(X^1)$,

then they can be, respectively, regarded as

(1-2) $[QY/s][\exists X/Y] f(X)$,

(2-2) $[QY/s][\forall X/Y] f(X)$,

when calculating their information.

Proof. We would like to prove Theorem 2, only prove the information of the formulae (1-1), (1-2) and (2-1), (2-2),

$$K_{(1-1)} = K_{(1-2)} \quad \text{and} \quad K_{(2-1)} = K_{(2-2)} .$$

Furthermore, based on Eq. (6), only prove their probabilities

$$P_{Q\exists\cdots\exists} = P_{Q\exists} \quad \text{and} \quad P_{Q\forall\cdots\forall} = P_{Q\forall}.$$

Without loss of generality, let the prefix of a MLL formula involve n same quantifiers appeared in succession, and let the base set d involve M elements. It is clear that

$$P_{\exists_1\exists_2\cdots\exists_n} = P_{\exists_1\exists_2\cdots\exists_{n-1}} = \cdots = P_{\exists\exists} = P_{\exists} = \frac{2^M - 1}{2^M},$$

$$P_{\forall_1\forall_2\cdots\forall_n} = P_{\forall_1\forall_2\cdots\forall_{n-1}} = \cdots = P_{\forall\forall} = P_{\forall} = \frac{1}{2^M},$$

Based on this, the following equalities also hold,

$$P_{\forall\exists_1\exists_2\cdots\exists_n} = P_{\forall\exists_1\exists_2\cdots\exists_{n-1}} = \cdots = P_{\forall\exists\exists} = P_{\forall\exists},$$

$$P_{\exists\forall_1\forall_2\cdots\forall_n} = P_{\exists\forall_1\forall_2\cdots\forall_{n-1}} = \cdots = P_{\exists\forall\forall} = P_{\exists\forall}. \quad \square$$

Theorem 2 shows the equivalence of information of MLL with the same quantifiers appeared in succession in the prefix of a formula. It can be used for convenience in the calculation. The example about this will be shown in Section 4.2.2.

Theorem 3. *Information of Tautology of MLL. Let K_A and K_B denote, respectively, the amounts of information of the MLL formulae A and B with different quantifiers, and let $A \Rightarrow B$ denote that if A is true then B is true, we have*

$$\text{IF } A \Rightarrow B, \text{ THEN } K_A \geq K_B.$$

Proof. Assume this is not true. Then there must be some B such that $A \Rightarrow B$ and $\sim(K_A \geq K_B)$ or $A \Rightarrow B$ and $K_A < K_B$. By Definition 1, we have $K_A = I_{S_{A1}} - I_{S_{A2}}$. Without loss of generality, let $I_{S_{A1}} = I_{S_{B1}}$. Then, $K_A < K_B$ is equivalent with $I_{S_{A2}} > I_{S_{B2}}$ or $S_A^2 > S_B^2$. This means that there is some posterior state which is included in S_A^2 but S_B^2, for which A is true but B is not true. This is a contradiction. $\square$

It is possible to evaluate the amounts of information of the MLL formulae with the different sequence of quantifiers in the prefix. For example, let the quantifiers set of the MLL formulae $\mathbf{Q} = \{Q_1, Q_2, Q_3\}$, and Q_i $(i = 1, 2, 3)$ denote either $\forall$ or $\exists$. Based on this, we define a state graph of quantifiers of the MLL formulae as shown in Fig. 6, and let d be a finite base set for presenting an IS-A hierarchy and let f be a predicate. Thus, Fig. 6 denotes all different combinations of quantifiers of the MLL formula,

$$[Q_1 Z/ *2d][Q_2 Y/Z][Q_3 X/Y] f(X).$$

And a partially ordered set of $\mathbf{Q}$ is $<Power(\mathbf{Q}), \subseteq>$, i.e.

$$Power(\mathbf{Q}) = \{\forall\forall\forall, \forall\forall\exists, \forall\exists\forall, \exists\forall\forall, \forall\exists\exists, \exists\forall\exists, \exists\exists\forall, \exists\exists\exists\}.$$

Thus, we can see that Fig. 6 is just its Hasse diagram presented a lattice structure. Therefore, the cover relation of $Power(\mathbf{Q})$ is

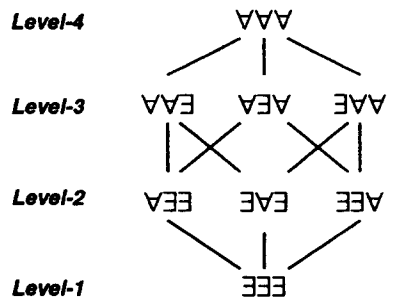

Fig. 6. A state space of quantifiers of MLL.

$$
\begin{aligned}
covPower(\mathbf{Q}) = \{&\langle \exists\exists\exists, \exists\exists\forall\rangle, \langle \exists\exists\exists, \exists\forall\exists\rangle,\\
&\langle \exists\exists\exists, \forall\exists\exists\rangle, \langle \exists\exists\forall, \exists\forall\forall\rangle,\\
&\langle \exists\exists\forall, \forall\exists\forall\rangle, \langle \exists\forall\exists, \exists\forall\forall\rangle,\\
&\langle \exists\forall\exists, \forall\forall\exists\rangle, \langle \forall\exists\exists, \forall\exists\forall\rangle,\\
&\langle \forall\exists\exists, \forall\forall\exists\rangle, \langle \exists\forall\forall, \forall\forall\forall\rangle,\\
&\langle \forall\exists\forall, \forall\forall\forall\rangle, \langle \forall\forall\exists, \forall\forall\forall\rangle\}.
\end{aligned}
$$

Based on this, let $\langle B, A\rangle$ denote any cover relation in $covPower(\mathbf{Q})$. Thus, based on Theorem 3, we have $K_B \leqslant K_A$ in which the formulae B, A satisfy $\langle B, A\rangle$. Furthermore, according to Eq. (6), then $K_B \leqslant K_A$ is equivalent with $P_B \geqslant P_A$, i.e.

$$
\begin{aligned}
&P_{\exists\exists\exists} \geqslant P_{\exists\exists\forall}\\
&P_{\exists\exists\exists} \geqslant P_{\exists\forall\exists}\\
&P_{\exists\exists\exists} \geqslant P_{\forall\exists\exists}\\
&\cdots\cdots\\
&P_{\exists\forall\forall} \geqslant P_{\forall\forall\forall}\\
&P_{\forall\exists\forall} \geqslant P_{\forall\forall\forall}\\
&P_{\forall\forall\exists} \geqslant P_{\forall\forall\forall}
\end{aligned}
$$

Example 2. According to Theorem 3, because

$$
\begin{aligned}
&[\forall S\#/h_1 : surface][\forall L\#/S : line]length(L\ 3) \Rightarrow\\
&[\forall S\#/h_1 : surface][\exists L\#/S : line]length(L\ 3),\\
&K_{\forall\forall} \geqslant K_{\forall\exists}.
\end{aligned}
$$

In general, we can create a state graph of quantifiers of the MLL formula for learning its quantifiers. The state graph as shown in Fig. 6 is the one in which the number of quantifiers is equal to 3. This state graph is divided into four levels. The uppermost level is with the most information. The further example about using Theorem 3 will be shown in Section 4.1.

3.2. An algorithm

Based on the definition and theorems stated above, we developed an effective algorithm for calculating the MLL information. At first, we only consider how to calculate the amounts of information of the MLL formulae with the prefixes ∀∀, ∀∃, ∃∀ or ∃∃. That is:

If we would like to calculate the amounts of information of the MLL formulae with the prefixes ∀∀, ∀∃, ∃∀ and ∃∃, only calculate ∀ and ∃∀ according to Theorem 1 and Theorem 2.

If the quantifiers in the MLL prefix either ∀∀, ∀∃, ∃∀ or ∃∃ can be used, select ∀∀ according to Theorem 3.

Furthermore, Eq. (9) or (10) can be easily used for calculating the probability of the sum of "tangled" or dependent elements. That is, if we calculate the amount of information of the MLL formula with the prefix ∀∃, then

Step 1: Calculate the probability of the MLL formula with the prefix ∀∃ in Eq. (9) or (10) according to whether there are "tangled" elements among sub-sets of the base set of an IS-A hierarchy.

Step 2: Calculate the amount of information K in Eq. (7).

Example 3. We use (2) in Example 1 as an example of using this algorithm and let $length(l_i 3)$ be simply represented as b_i. Thus, if we would like to calculate the amount of information of the MLL formula with the prefix ∀∃, then first, calculate the probability of the MLL formula with the prefix ∃∀ in Eq. (9). That is,

$$P_{\exists\forall} = P((b_1 \cap b_2 \cap b_4) \cup (b_1 \cap b_3 \cap b_5) \cup (b_2 \cap b_3 \cap b_6) \cup (b_4 \cap b_5 \cap b_6)) = \frac{23}{2^6},$$

next, calculate the amount of information

$$K_{\forall\exists} = \log \frac{1}{1 - P_{\exists\forall}} = 0.642 .$$

Although most applications only need to consider the MLL formulae with 2 alternating quantifiers or the same quantifiers appeared in succession (see Section 4), we will now discuss how to extend the above algorithm for processing more than 2 alternating quantifiers like ∃∀∃. ∀∃∀, etc. This extension can be easily done because we can expand an expression of probability corresponding to a complex sequence of quantifiers of a MLL formula into a simpler form (i.e. the same form as ∃∀), so that Eq. (9) can be used for calculating its probability. We describe it by the following example.

Example 4. If we would like to calculate the amount of information of the following MLL formula with the prefix ∀∃∀ and the IS-A hierarchy shown in Fig. 7,

$$[\forall Z/ *2d][\exists Y/Z][\forall X/Y] f(X) ,$$

then first, write out the expression of probability corresponding to the MLL formula as follows:

$$P_{\forall\exists\forall} = P((a_1 \cap a_2 \cup a_3) \cap a_4) .$$

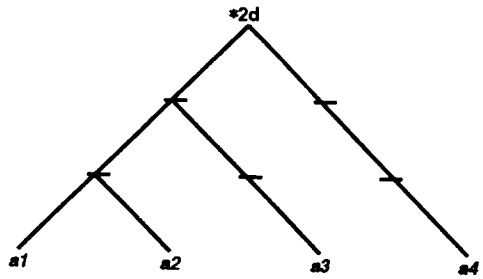

Fig. 7. A sample IS-A hierarchy to describe the more complex MLL prefix.

Then, expand this expression into a simpler form as follows:

$$P_{\forall\exists\forall} = P((a_1 \cap a_2 \cup a_3) \cap a_4) = P((a_1 \cap a_2 \cap a_4) \cup (a_3 \cap a_4)) .$$

That is, this is the same form as $\exists\forall$. Thus, Eq. (9) can be used for calculating its probability. That is

$$P_{\forall\exists\forall} = P((a_1 \cap a_2 \cap a_4) \cup (a_3 \cap a_4)) = \frac{5}{2^4} ,$$

and the amount of information is

$$K_{\forall\exists\forall} = \log \frac{1}{P_{\forall\exists\forall}} = 1.678 .$$

4. Hierarchical modeling learning

Based on the preparation in Sections 2 and 3, this section introduces the approach of *hierarchical model learning* (HML). It is mainly composed of three functions:

- Representing the concept clusters discovered from a database by DBI as the MLL formulae with hierarchical models in a knowledge-base. It includes hierarchical modeling for concept clusters and automatic selection of quantifiers in the prefixes of the MLL formulae.
- Refining the hierarchical models by using domain knowledge and/or when new concept clusters are discovered along with data change in a database. This is to select automatically a best (or more refined) hierarchical model from more hierarchical models belonging to a family.
- Managing the hierarchical models by using the set chains of hierarchical models and their inheritance graphs.

We would like to use a breast cancer database [6], which calls *breast-cancer* as shown in Table 1, as an example for showing our approach. In this database, each tuple corresponds to one patient and values of 10 attributes are given for each patient. The domain of every attribute is given by the sets of 9 quantized values that are classified as a case of benign or malignant cancer, resulting from clinical examinations related to this disease. The meanings of the attributes used in Table 1 are as follows:

Table 1
DB: breast-cancer

id	code-n	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9
1	160296	4	5	8	8	10	5	10	8	10	3
2	342245	2	1	1	3	1	2	1	1	1	1
3	428598	2	1	1	3	1	1	1	2	1	1
4	492561	2	4	3	2	1	3	1	2	1	1
5	493452	2	1	1	3	1	2	1	1	1	1
6	493452	2	4	1	2	1	2	1	2	1	1
7	521441	2	5	1	1	2	2	1	2	1	1
8	560680	2	3	1	2	1	2	1	2	1	1
9	636437	2	1	1	1	1	2	1	1	1	1
10	640712	2	1	1	1	1	2	1	2	1	1
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
·	1266154	4	8	7	8	2	4	2	5	10	1
·	1272039	2	1	1	1	1	2	1	2	1	1
·	1276091	2	2	1	1	1	2	1	2	1	1
·	1276091	2	1	3	1	1	2	1	2	2	1
·	1276091	2	5	1	1	3	4	1	3	2	1
·	1277629	2	5	1	1	1	2	1	2	2	1
·	1293439	2	3	2	2	3	2	1	1	1	1
·	1293439	2	6	9	7	5	5	8	4	2	1
·	1294562	4	10	8	10	1	3	10	5	1	1
·	1295186	4	10	10	10	1	6	1	2	8	1

code-n – code-number (Sample Code Number)

a0 – b-cancer-type (Breast Cancer Type: 2 for benign, 4 for malignant)

a1 – clump-t (Clump Thickness)

a2 – u-cell-size (Uniformity of Cell Size)

a3 – u-cell-shape (Uniformity of Cell Shape)

a4 – marginal-adhesion (Marginal Adhesion)

a5 – s-e-cell-size (Single Epithelial Cell Size)

a6 – bare-nuclei (Bare Nuclei)

a7 – bland-chromatin (Bland Chromatin)

a8 – normal-nucleoli (Normal Nucleoli)

a9 – mitoses (Mitoses).

Furthermore, in order to describe the change of data in database, the breast cancer database is divided into two groups: *group* 1 for fundamental data and *group* 2 for its variation. The objective is to find which conditions of the 9 attributes indicate malignant and which no malignant cancer, and form concept clusters by decomposing this breast cancer

database, so that finally to represent the results as the MLL formulae with hierarchical models in a knowledge-base and refine/manage them by HML [29].

4.1. Knowledge generation

The process of knowledge generation can be divided into two main stages by using cooperatively DBI and HML. The **first stage** is to decompose a database for forming concept clusters by using DBI. This has been described in our paper [29]. To make this paper self-content, we describe briefly the main steps of forming concept clusters in DBI as follows:

Step 1: Create a *Probability Distribution Matrix* (*PDM*). There are many kinds of methods for creating the PDM, depending on their purposes. For our application, the dependency relations between any two attributes are considered, their probability distributions are calculated and recorded in a PDM. Let $\mathbf{a} = \{a_1, a_2, \ldots, a_n\}$ and $\mathbf{b} = \{b_1, b_2, \ldots, b_m\}$ be the sets of different values of any two attributes in a database that has been preprocessed. Using conditional probability, we have

$$p(x_i \mid x_j) = \frac{p(x_i \cap x_j)}{p(x_j)} \quad x_i, x_j \in \mathbf{a}, \mathbf{b}\,. \tag{11}$$

From this we define p_{ij}, the probability distributions, to be $p(x_i \mid x_j)/N$, where N is the number of attributes. These p_{ij} constitute the entities of the PDM.

Step2: Form the diagonal matrix. It is a step as pre-processing before decomposing the PDM. Two methods, *diagonalization by a special attribute* as a supervised method and *diagonalization by the optimum decomposition* as an unsupervised method, can be used for this according to the cases in which a criterion of forming the diagonal PDM is given by the user or not. Since there is obviously a special attribute (i.e., the attribute *b-cancer-type*, or e.g. c_1, c_2 as shown in Fig. 8) that can be chosen by the user as a criterion of forming the diagonal PDM for this breast cancer database, the method, *diagonalization by a special attribute*, is used for obtaining the diagonal PDM shown in Fig. 8.

Step3: Decompose the diagonal PDM by a decomposing algorithm. In decomposing, primary factors for describing some concepts are aggregated by selecting proper attributes,

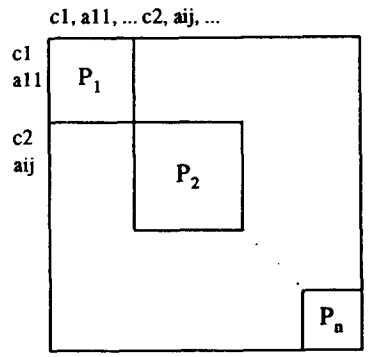

Fig. 8. Forming the diagonal PDM by a special attribute.

and two kinds of noises, *minor elements* and *irrelevant elements* are neglected. Where, the *minor elements* are those whose probability values are much smaller than other related values. For example, in the *breast cancer database*, if bare-nuclei = 1 then the probabilities of malignant and benign are 0.02 and 0.98, respectively. Thus, bare-nuclei = 1 cannot be used as one of the conditions of malignant cancer. Therefore, it is neglected as a minor factor. The *irrelevant elements* are those who cannot be used to differentiate concepts. For example, in the *breast cancer database*, if mitoses = 3 then the probabilities of malignant and benign are both nearly 0.5, even though some data are not appearing in Table 1. Thus, mitoses = 3 cannot be used to differentiate malignant from benign. That is, mitoses = 3 is useless for knowledge discovery and cannot be classified into a cluster. Therefore, they should also be omitted as an irrelevant element although its probability value may be fairly large. As the result of decomposing a PDM, several sub-matrices are formed. These results are applied back to a database that has been preprocessed and the job of decomposing the database is thus completed. As the result of decomposing the database, concept clusters are formed.

As its consequence, the following two concept clusters can be discovered from *group 1* of the breast cancer database,

1. The conditions of benign cancer:

   clump-t: 1, 4, 2, 6 .
   u-cell-size: 1, 3, 2, 9.
   u-cell-shape: 2, 1, 4.
   s-e-cell-size: 1.
   bare-nuclei: 0, 3, 8.
   bland-chromatin: 1, 3, 4.
   mitoses: 2.

2. The conditions of malignant cancer:

   clump-t: 10, 7, 8, 9.
   u-cell-size: 8, 6, 10, 5, 7.
   u-cell-shape: 8, 10.
   marginal-adhesion: 10, 7.
   s-e-cell-size: 6, 8, 10.
   bare-nuclei: 10, 9.
   bland-chromatin: 8, 7, 9, 6, 10, 5.
   normal-nucleoli: 10, 8, 9, 6.
   mitoses: 4, 8.

That is, the elements in above two clusters can be respectively used as the conditions of indicating malignant and which no malignant cancer.

Based on the results stated above, the **second stage** of knowledge generation is to represent the concept clusters as the MLL formulae with hierarchical models in a knowledge-base by using HML. For example, the hierarchical model corresponding to *cluster-1*, which represents the conditions of benign cancer, can be generated as shown in Fig. 9, and the corresponding MLL formula can be created as *Rule* 1:

Rule 1: /* The rule for diagnosing breast cancer */

[∀Y# / benign : symptom][∃X# / Y]p-breast-cancer(Y X).

Rule 1 reads "*if the symptoms recorded in the set-elements relations about benign are satisfied, then the breast cancer is benign.*"; The similar rule for the malignant cancer can be created to mean "*if the symptoms recorded in the set-elements relations about malignant are satisfied, then the breast cancer is malignant.*".

Thus, there are two important jobs in knowledge generation in HML. The first is *hierarchical modeling*. Where, the process of representing a concept cluster by an IS-A hierarchy (i.e. the hierarchical model represented by the set-elements relation in MLL, and Fig. 9 is an example of its equivalent graph) is called *hierarchical modeling*. For example, two concept clusters discovered from *group* 1 of the breast cancer database are represented by two IS-A hierarchies.

Another job in knowledge generation in HML is to select quantifiers in the MLL prefix. Since the choice of the MLL prefix is sensitive to the relationships among the conditions of indicating malignant and which no malignant cancer, we have to interpret that the relationships among the conditions are either conjuctive or disjunctive. First, according to the principle that values in an attribute do not happen at the same time for a tuple in universal relation, the relationships among the conditions belonging to an attribute in a cluster are disjunctive. Thus, for the conditions belonging to an attribute, we use the quantifier ∃. Furthermore, you can ask that the relationships among the conditions belonging to different attributes should be either conjunctive or disjunctive. This mainly depends on the method of creating PDM in DBI. Since the method of creating PDM described in this paper only considers the dependency relations between any two attributes, and the attribute "b-cancer-type" is used as a special attribute for forming the diagonal PDM, the elements in every cluster can be interpreted as the conditions of indicating malignant and which no malignant cancer. However, the relationships (i.e. conjunctive or disjunctive) among the conditions

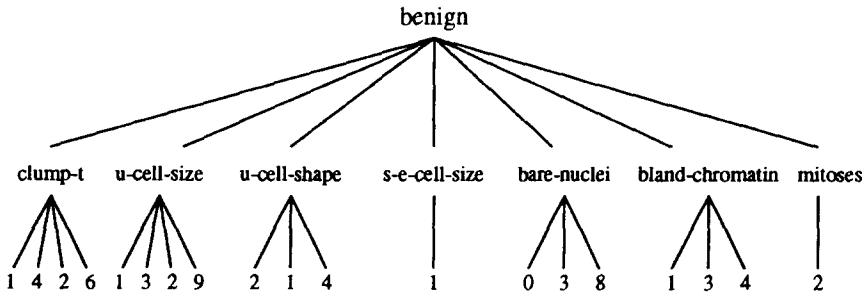

Fig. 9. The hierarchical model of *group 1*.

belonging to different attributes are indefinite. In other words, both of the quantifiers ∀ and ∃ can be used for the conditions belonging to different attributes. Thus, both of the MLL prefixes,

(1) $[\exists Y\# / \text{benign} : \text{symptom}][\exists X\# / Y]$

and

(2) $[\forall Y\# / \text{benign} : \text{symptom}][\exists X\# / Y]$,

can be used for representing the acquired concept clusters. Where, we can see that if the MLL formula with the prefix (2) is true then the one with the prefix (1) is true, that is,

$$[\forall \text{Y}\# / \text{benign} : \text{symptom}][\exists \text{X}\# / \text{Y}]\text{p-breast-cancer(Y X)} \Rightarrow$$
$$[\exists \text{Y}\# / \text{benign} : \text{symptom}][\exists \text{X}\# / \text{Y}]\text{p-breast-cancer(Y X)},$$

from Theorem 3 stated in Section 3.1, we know that the amount of information of the MLL formula with the prefix (2) is larger than the one with the prefix (1). Therefore, the prefix (2) is selected for representing the concept clusters.

The main reasons why the discovered concept clusters are represented as the MLL formulae with hierarchical models are

- We need to use a kind of model representation in our system for both compact representing and flexibly revising the bulky concept clusters discovered from a database;
- By representing the discovered concept clusters as the MLL formulae, we can further combine knowledge-based system techniques with statistical/decision analysis methods for flexibly managing, refining and using them.

Knowledge representation in MLL is a kind of model representation. That is, the representation of the structure and the functionality (property, function, etc.) description is separated so that by changing the structure, the representation with the more information is reached. For example, the structure entity discovered from the breast cancer database by DBI can be recorded by the set-elements relations that represent the IS-A hierarchies. The structural relation is represented by the MLL prefix,

$$[\forall Y\# / \text{benign} : \text{symptom}][\exists X\# / Y],$$

and the functionality description is represented by

$$\text{p-breast-cancer(Y X)}.$$

Even when structure changed by the additional data, the MLL formula is not generally changed. Because of the uncertainty and incompleteness of data in databases, the knowledge discovered from databases is only a hypothesis, which must be refined (evaluated/modified) in multiple learning phases. Also, because databases are not static but dynamic, new hypotheses are generated when data change in databases [27, 29]. Therefore, this kind of model representation is very important for knowledge discovery in databases. Here, learning is to select an expression with more information.

In the phase of knowledge generation, we see that the evaluation of the MLL information is mainly used for selecting quantifiers of a MLL formula with same structure from the state space of quantifiers. This is one of two aspects of evaluating the MLL information as stated in

Section 3.1. Another aspect is to evaluate a MLL formula with different structures. This involves knowledge refinement to be stated in the following Section 4.2.

4.2. Refinement

There are two main methods for refining the hierarchical model. First, a proper hierarchical model is selected by evaluating the MLL information. That is, re-construct an expression of the hierarchical model to a more informative one. Second, the hierarchical model is refined by cooperatively using domain knowledge and informative evaluation. That is, an expression is so refined as to satisfy special requirement. We consider that these methods are reasonable because the information of MLL may be either an increase or decrease along with the change of data in a database, and experts can easily represent their knowledge in a knowledge-base for refining the hierarchical models.

4.2.1. Refinement by evaluating the information of MLL

When new concept clusters are discovered from a database, the better hierarchical model can be selected by evaluating the MLL information. For example, when another concept cluster as shown in Fig. 10 is discovered by adding *group* 2 of data to the breast cancer database, we can calculate the amounts of their information based on the MLL prefix,

$$[\forall Y\# / \text{benign} : \text{symptom}][\exists X\# / Y] ,$$

for selecting the better one from two hierarchical models shown in Figs. 9 and 10 by using the algorithm stated in Section 3.2. Since the result of the calculation is

$$K_{g2} = 2.864 > K_{g1} = 2.764 ,$$

the hierarchical model shown in Fig. 10 is selected. That is, learning is to select a hierarchical model with more information.

4.2.2. Use of domain knowledge in refinement

Since experts can bring domain knowledge to bear while refinement, the hierarchical model also can be refined by using cooperatively domain knowledge and informative evaluation. For example, if the following domain knowledge,

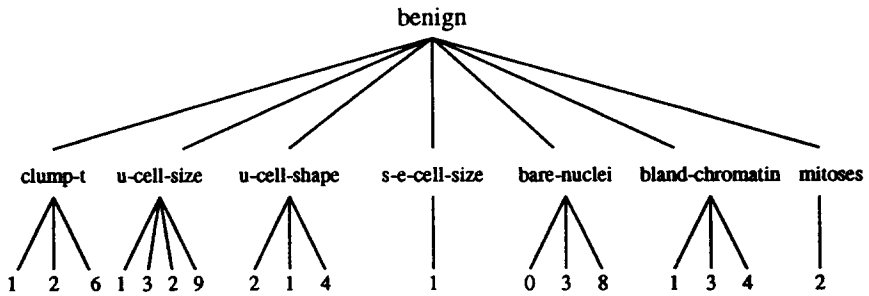

Fig. 10. A hierarchical model in which *group 2* of data was added.

!ins_e uniformity u-cell-size, u-cell-shape;

!ins_e cell u-cell-size, u-cell-shape, s-e-cell-size;

!ins_e nuclei bare-nuclei, normal-nucleoli, mitoses;

!ins_e other clump-t, bland-chromatin;

is used, then a more refined hierarchical model as shown in Fig. 11 can be acquired. That is, domain knowledge is used for conceptual abstraction (generalization). Here, the lowest leaves of the hierarchical model are only *observable values* that are collected in a database. The other values are called *abstract values*, and the second lowest leaves of the hierarchical model are called *the lowest level of abstract values*. However, *abstract values* can be "tangled" (i.e. a value can belong to more than one *abstract value* in a higher level as shown in Fig. 11).

The prefix of the MLL formula with the hierarchical model as shown in Fig. 11 can be represented into

$$[\forall Z\#/\text{benign:symptom}][\forall Y\#/Z][\exists X\#/Y].$$

Since the same quantifiers appeared in succession in the prefix of this formula, the amount of information of this formula with the hierarchical model shown in Fig. 11 is the same with the one shown in Fig. 10 (according to Theorem 2). Therefore, the hierarchical model shown in Fig. 11, in which *group* 2 of data was added and conceptual abstraction was done, is selected as a more refined one. Here, learning is to select the best hierarchical model by using cooperatively domain knowledge and informative evaluation.

4.3. Management

Management of hierarchical models is an important issue when more hierarchical models belonging to a family are generated along with data change (i.e. to add, delete or update data) in a database. In HML, the set chains and the inheritance graphs of hierarchical models are used for this purpose. By means of them, the following jobs can be done:

- Hierarchical models belonging to a family, which denote concept clusters discovered from a database, are first stored in the set chains, and then are refined (evaluated/modified);

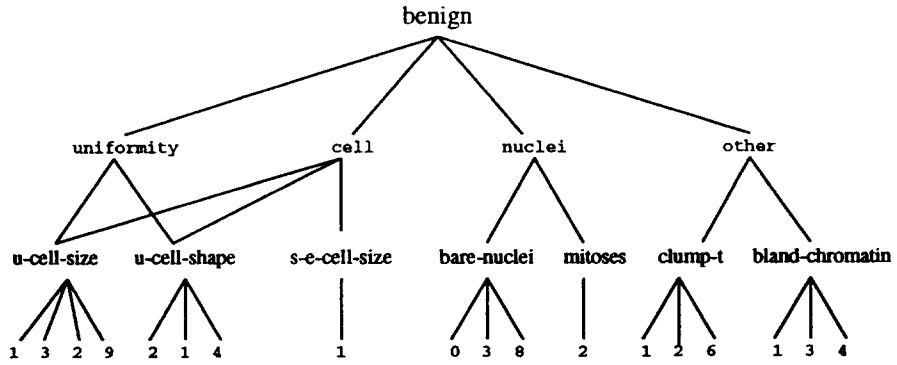

Fig. 11. A hierarchical model used domain knowledge.

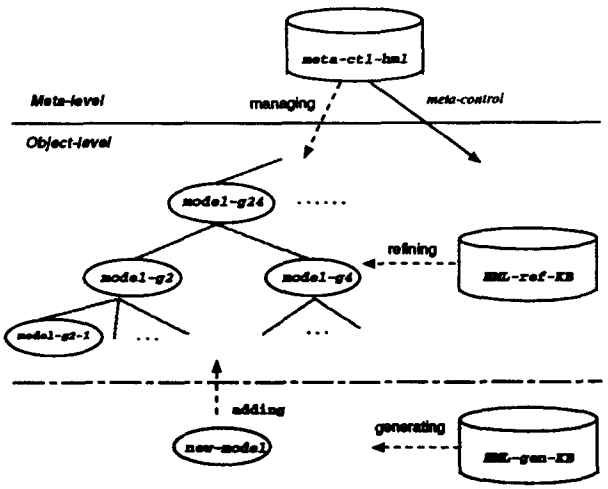

Fig. 12. An inheritance graph and operations.

- The time and history of hierarchical models are represented and managed. That is, the set chains for storing hierarchical models are dynamically generated as time goes on for recording the evolution process of hierarchical models;
- The inheritance graphs of hierarchical models are dynamically generated for describing the relationships among hierarchical models.

The set chains of hierarchical models are defined by the set-elements relation of KAUS. The set chains and the inheritance graphs are managed by a meta knowledge level as shown in Fig. 12. Fig. 12 also shows the structure of the inheritance graph and some operations for it in HML. That is, *model-g24*, *model-g2* and *model-g4*, etc. shown in Fig. 12 denote several hierarchical models with inheritance relationship; three knowledge-bases of HML, *meta-ctl-hml*, *hml-ref-kb* and *hml-gen-kg*, which are divided into two knowledge levels: *meta-level* and *object-level*, are, respectively, used for performing/controlling different operations.

5. Conclusion

We presented a *hierarchical model learning* (*HML*) approach for refining and managing concept clusters discovered from databases. It can be considered as a typical approach in which information theory is used as a criterion for learning knowledge. It is based on the model representation of Multi-Layer Logic (MLL). A prototype of HML has been implemented as one sub-system of our GLS discovery system. Main features of our approach can be summarized as follows:

- Hierarchical modeling for concept clusters. That is, concept clusters discovered from databases are represented as the MLL formulae with hierarchical models in a knowledge-base;
- Automatic selection of quantifiers in the prefix of the MLL formula. That is, quantifiers $\forall$

and ∃ for constituting the prefix of a MLL formula are selected from the state space of quantifiers;

- Automatic selection and refinement of hierarchical models. That is, the best hierarchical model is selected from a family of hierarchical models by evaluating the information of MLL;
- Domain knowledge can be cooperatively used with informative evaluation in refinement for acquiring the more refined hierarchical model;
- More hierarchical models generated along with data change in a database can be managed by using the set chains of hierarchical models and the inheritance graphs of hierarchical models. These set chains and the inheritance graphs are managed by a meta knowledge level.

Furthermore, we would like to emphasize that the *hierarchical model learning* (*HML*) approach is only used as a learning phase in multiple learning phases of our GLS discovery system [30], the results of the refinement by HML are not the final ones in the discovery process, and can be further used in the next learning phase for acquiring more high-level knowledge. How to use cooperatively hierarchical model learning with case based reasoning and decision analysis is a further research subject. Its objective is just to acquire more high-level knowledge from the discovered concept clusters which are represented as the MLL formulae with hierarchical models. This involves a new sub-system of GLS that is being developed by us [34].

Acknowledgements

The authors would like to thank Professor Hori and Professor Takasu for useful discussions in OHlab seminar and SIG-KDD meeting. Special thanks is due to Mr. Yamauchi for his enormous efforts in implementing KAUS and his help in use of KAUS. Further, we would like to thank the referees of the DKE journal for their valuable comments on the first submitted version of this paper.

This paper is a revised version of a paper presented at the 5th IEEE International Conference on Tools with Artificial Intelligence (TAI '93), Boston, USA (November 1993). We received financial support for the successful international conference from International Information Science Foundation of Japan.

References

[1] J. Cendrowska, PRISM: An algorithm for inducing modular rules, *Int. J. Man-Machine Studies* 27 (1987) 349–370.

[2] P. Clark and T. Niblett, The CN2 induction algorithm, *Machine Learning* (1989) 261–283.

[3] J. Han, Y. Cai and N. Cercone, Data-driven discovery of quantitative rules in relational databases, *IEEE Trans. Knowl. Data Engrg.* 5(1) (1993) 29–40.

[4] W. Klosgen, Problems for knowledge discovery in databases and their treatment in the statistics interpreter explora, *Int. J. Intell. Systems* 7(7) (1992) 649–673.

[5] C.L. Liu, *Elements of Discrete Mathematics* (McGraw-Hill, New York, 1977).

[6] O.L. Mangasarian and W.H. Wolberg, Cancer diagnosis via linear programming, *SIAM News* 23(5) (1990) 1–18.
[7] C.J. Matheus, P.K. Chan and G. Piatetsky-Shapiro, Systems for knowledge discovery in databases, *IEEE Trans. Knowl. Data Engrg.* 5(6) (1993) 904–913.
[8] R.S. Michalski, I. Mozetic, J. Hong and N. Lavrac, The multipurpose incremental learning system AQ15 and its testing application to three medical domains, *Proc. 5th National Conference on Artificial Intelligence* (1986) 1041–1045.
[9] R.S. Michalski, Mining for knowledge in databases: The INLEN architecture, initial implementation and first results, *J. Intelligent Information Systems* 1(1) (1992) 85–113.
[10] R.S. Michalski, J.G. Carbonell and T.M. Mitchell, *Machine Learning – An Artificial Intelligence Approach*, Vols. 1–3 (Morgan Kaufmann Publishers, 1983, 1986, 1990).
[11] M. Nunez, The use of background knowledge in decision tree induction, *Machine Learning* 6 (1991) 231–250.
[12] S. Ohsuga, A consideration to knowledge representation – an information theoretic view, *Bulletin of Informatics and Cybernetics* 21 (Nos. 1–2) (1984) 121–135.
[13] S. Ohsuga, Toward intelligent CAD systems, *Computer Aided Design* 21(5) (1989) 315–337.
[14] S. Ohsuga, *Data Bases and Knowledge Bases* (Ohm Ltd., Japan, 1989).
[15] S. Ohsuga, Framework of knowledge based systems – multiple meta-level architecture for representing problems and problem solving processes, *Knowledge Based Systems* 3(4) (1990) 204–214.
[16] S. Ohsuga, How can knowledge based systems solve large scale problems?: Model based decomposition and problem solving, *Knowledge Based Systems* 6(1) (1993) 38–62.
[17] S. Ohsuga and H. Yamauchi, Multi-layer logic – A predicate logic including data structure as knowledge representation language, *New Generation Computing* 3(4) (1985) 403–439.
[18] G. Piatetsky-Shapiro and C.J. Matheus, Knowledge discovery workbench for exploring business databases, *Inter. J. Intell. Systems* 7(7) (1992) 675–686.
[19] G. Piatetsky-Shapiro and W.J. Frawley, eds., *Knowledge Discovery in Databases* (AAAI Press and The MIT Press, 1991).
[20] J.R. Quinlan, Induction of decision trees, *Machine Learning* 1(1986) 81–106.
[21] J.R. Quinlan, Generating production rules from examples, *Proc. 10th Int. Joint Conf. on Artificial Intelligence* (1986) 304–307.
[22] J.W. Shavlik and T.G. Dietterich, eds., *Readings in Machine Learning* (Morgan Kaufmann Publishers, 1990).
[23] P. Smyth and R.M. Goodman, An information theoretic approach to rule induction from databases, *IEEE Trans. Knowl. Data Engrg.* 4(4) (1992) 301–316.
[24] S. Watanabe, *Knowing and Guessing – A Quantitative Study of Inference and Information* (John Wiley and Sons Inc., 1969).
[25] H. Yamauchi and S. Ohsuga, KAUS as a tool for model building and evaluation, *Proc. 5th Int. Workshop on Expert Systems and Their Applications* (1985).
[26] H. Yamauchi and S. Ohsuga, Loose coupling of KAUS with existing RDBMs, *Data & Knowledge Engrg.* 5(4) (1990) 227–251.
[27] N. Zhong and S. Ohsuga, GLS – A methodology for discovering knowledge from databases, *Proc. 13th Int. CODATA Conference entitled "New Data Challenges in Our Information Age"* (1992) A20–A30.
[28] N. Zhong and S. Ohsuga, HML – An approach for refining/managing knowledge discovered from databases, *Proc. 5th IEEE International Conference on Tools with Artificial Intelligence (TAI '93)* (IEEE Computer Society Press, 1993) 418–426.
[29] N. Zhong and S. Ohsuga, Discovering concept clusters by decomposing databases, *Data & Knowledge Engrg.* 12(2) (1994) 223–244.
[30] N. Zhong and S. Ohsuga, The GLS discovery system: Its goal, architecture and current results, in: Z.W. Ras and M. Zemankova, eds., *Methodologies for Intelligent Systems, Proc. 8th Int. Symp., ISMIS '94, Lecture Notes in Artificial Intelligence* 869 (Springer-Verlag, 1994) 233–244.
[31] N. Zhong and S. Ohsuga, Managing/refining structural characteristics discovered from databases, *Proc. 28th Hawaii Int. Conf. on Sys. Sciences (HICSS-28)*, edited in the *minitrack on information sharing and knowledge discovery in large scientific databases* 3, (IEEE Computer Society Press, 1995).

[32] N. Zhong and S. Ohsuga, KOSI – An integrated discovery system for discovering functional relations from databases, *J. Intelligent Information Systems* 5(1) (1995) 25–50.
[33] N. Zhong and S. Ohsuga, Toward a multi-strategy and cooperative discovery system, *Proc. First Int. Conf. on Knowledge Discovery and Data Mining (KDD-95)* (AAAI Press, 1995) 337–342.
[34] N. Zhong and S. Ohsuga, From Conceptual Hierarchical Models to More High-Level Knowledge (draft).
[35] J.M. Zytkow and R. Zembowicz, Database exploration in search of regularities, *J. of Intell. Infor. Systems*, 2(1) (1993) 39–81.

Ning Zhong is currently an assistant professor of the Department of Computer Science and Systems Engineering at Yamaguchi University, Japan. He is also a cooperative research fellow in Research Center for Advanced Science Technology (RCAST) at the University of Tokyo. He graduated at the Beijing Polytechnic University in 1982 and has been a lecturer in the Dept. of Computer Science, Beijing Polytechnic University. He received the Ph.D. degree in the Interdisciplinary Course on Advanced Science and Technology from the University of Tokyo. His research interests include knowledge discovery in databases, machine learning and intelligent information systems. Dr. Zhong is a member of Japanese Society for Artificial Intelligence, the Association for Foundations of Science, Language and Cognition (AFOS), IEEE Computer Society.

Setsuo Ohsuga is currently a professor of the Department of Information and Computer Science at Waseda University, Japan. He has been professor and director of Research Center for Advanced Science and Technology (RCAST) at the University of Tokyo. He has also been president of the Japanese Society for Artificial Intelligence. He graduated at the University of Tokyo in 1957. From 1957 to 1961 he worked in Fuji Precision Machinery (the present Nissan Motors). In 1961 he moved to the University of Tokyo and received Ph.D. in 1966. He became associate professor in 1967 and professor in 1981. His research interests are artificial intelligence, knowledge information processing, databases and CAD. He has received awards for his researches twice from the Academic Society in Japan. He is a member of the editorial boards of 9 scientific journals.