경영정보학연구
제5권 2호
1995년 12월

외연적 객체모델의 정형화

정 철 용[1)]

A Formal Presentation of the Extensional Object Model

We present an overview of the Extensional Object Model (ExOM) and describe in detail the learning and classification components which integrate concepts from machine learning and object-oriented databases. The ExOM emphasizes flexibility in information acquisition, learning, and classification which are useful to support tasks such as diagnosis, planning, design, and database mining. As a vehicle to integrate machine learning and databases, the ExOM supports a broad range of learning and classification methods and integrates the learning and classification components with traditional database functions. To ensure the integrity of ExOM databases, a subsumption testing rule is developed that encompasses categories defined by type expressions as well as concept definitions generated by machine learning algorithms. A prototype of the learning and classification components of the ExOM is implemented in Smalltalk/V Windows.

1) 상명대학교 경영학과

I. Introduction

In recent years organizational computing has received a great deal of attention from both computer scientists and organizational researchers because of the increasing strategic importance of information technology in an organization's success. The nature of o rganizational tasks is to cope with conflicting, inconsistent, and partial information to generate sound, relevant, and reliable information to support decision making and action. Daft and Weick [DW84] proposed a model of organizations as interpretation systems to cope with the demands of organizational tasks. In their model, an organization scans the environment, interprets scanned data, executes actions, and learns from the feedback of actions. Scanning is the process of monitoring the environment to collect data either through formal data collection systems or personal contacts. Interpretation gives meaning to data by allowing members to develop shared understanding of their environment. Organizational learning is the process by which knowledge of action–outcome relationships between the organization and its environment is obtained.

From a computational perspective, work in machine learning, knowledge discovery, active databases, and data visualization addresses capabilities envisioned in Daft and Weick's model. The motivation for computer supported interpretation is information overload, complexity of interpretation, and volatile organizational memory. As database management systems and electronic data interchange have become popular, organizations receive enormous amounts of computer–readable data through inter and intra organizational computer systems. It has been remarked that computers have delivered a flood of data rather than a mountain of wisdom. Accurate and timely interpretation of this torrent of data is rather knowledge intensive and complex. There are many competing approaches to interpretation of which the knowledge is often dispersed throughout an organization. Organizational interpretations may be recorded in many forms including documents, procedure books, files, and informally through shared beliefs. Storing interpretations and the underlying data within the same database system can imprcve the retention of interpretations especially with regards to personnel turnover and time.

Traditional object–oriented data models have several limitations as a formalism for

computer supported interpretation systems. The traditional models are rather inflexible for information acquisition as they emphasize efficiency, uniformity, encapsulation, and reusability by assuming a mostly complete model of the world. They require a tight coupling between class and objects as the definition of a class must be pre-defined before its member objects can be stored. The instances of a class must be uniform except for the presence of null values and limited forms of exceptions. The traditional models also leave classification of objects as a burden of the user. Frequently the user decides class membership perhaps guided by integrity constraints. If the database system supports triggers and deductive reasoning, the designer can define rules to determine class membership. The role of learning to determine concept definitions and object clusterings, and the role of classification to interpret concept definitions is outside the scope of most active database systems.

We have designed the ExOM as an underlying formalism for computer supported interpretation systems. The ExOM is a departure from traditional object-oriented models because it emphasizes individual objects rather than classes. Thus, it may be considered an object-based rather than class-based approach [Wegn90]. The ExOM supports incomplete, imprecise, and incremental information acquisition through feasible values, strict and non-strict categories, and separation of object definition from classification. Besides flexibility in information acquisition, the ExOM also features both deductive and inductive reasoning. From our earlier experience with the deductive, object-oriented language, Semlog [SJW91], deductive reasoning is alone not sufficient to support interpretation of business events. Thus, a major goal of our research has been to support a broad range of inductive reasoning styles within the ExOM.

In this paper, we describe in detail the learning and classification components of the ExOM. By learning, we mean generating concept definitions and possibly determining object groupings or clusterings for a set of objects. This definition encompasses both unsupervised learning which determines object clusters and concept definitions as well as supervised learning which only generates concept definitions. By classification, we mean testing an object for membership in a given category by applying known concept definitions. Classification involves searching concept definitions in an efficient manner and matching a

concept definition to the values of an object.

The main contributions highlighted in this paper are support for a broad range of learning and classification approaches and the integration of learning and classification components with traditional database functions. The ExOM supports the learning dimensions of example acquisition(incremental or batch), category space(flat or hierarchical), and supervision(supervised or unsupervised) as well as teacher supplied concept definitions. The ExOM supports several kinds of classification functions and concept definitions including type expressions with subsumption reasoning, decision trees, weighted feature records with inexact matching, and simple rules. The learning and classification components are integrated into traditional database functions through operators that interpret concept definitions, specify a set of categories for learning and classification, determine a set of objects from which to apply a learning algorithm, conduct a learning experiment, explain membership of an object in a category, and explain the concept definition, classification function, and learning algorithm of a category. In addition, the ExOM provides standard interfaces for introducing new learning and classification functions as well as new kinds of concept definitions. Because consistency of ExOM databases is still important, a subsumption testing rule is given that can be used to augment a taxonomic reasoner. We describe a prototype implementation of the ExOM in which most features described in this paper are included.

II. Overview of the ExOM

1. Schema Definition

In designing an ExOM database, we first define terms used in objects and categories. Type names and associated constraints should be defined before use. The fundamental part of an ExOM schema is the representation of categories and their relationships. Objects are asserted and organized into categories. We use the term category instead of class to emphasize the broader role of classification and learning in the ExOM than traditional object-oriented models.

〈Figure 1〉 22shows the schema of an imaginary bank database. Structured categories are similar to classes in traditional object-oriented databases in that members of a structured category are homogeneous in their attributes. The numbers in paren-

theses following an attribute's type name are the minimum and maximum cardinality, respectively.

STRUCTURED CATEGORY Customer
Parents TOP
[CID::String(1,1), name::String(1,1), address::String(1,1), phone::String]
Key CID

STRUCTURED CATEGORY Household
Parents Customer
[marrital_status::Marriage_type, children::Set of Children,
vocation::{'Professor', 'Doctor', 'Others'}, position::String,
annual_income::Dollars, property::Dollars, deposits:Set of Deposit(1,n),
loans::Set of Loan(0,n), contribution::{'High', 'Medium', 'Low'},
relationship_duration::{'Long','Medium', 'Short'}]

STRUCTURED CATEGORY Account
Parent TOP
[acct_ID::String, product-name::Product(1,1), customer_ID::Customer(1,1),
opening_date::Date(1,1), balance::Dollars(1,1)]

STRUCTURED CATEGORY Credit_Applied
Parents Top
[application_no::Integer(1,1), application_date::Date(1,1),
applicant::Customer(1,1), product_name::Product(1,1),
linked_accounts::Account(0,1),credit_line::Dollars,collateral::Collateral_type,
collateral_value::Dollars]
Key application_no

Note that as a subcategory of Customer, Household inherits its attributes. Credit_Applied represents applications for credits or loans of several kinds. It includes loan or credit product name, accounts related to this application, the amount requested, and collateral information. Each application is evaluated for its approval or rejection and further categorized into Credit_Approved or Credit_Rejected. In the ExOM, the type expression of a structured category(i.e., the attributes and restrictions) serves as a both necessary and sufficient condition of membership.

The ExOM also supports unstructured categories which provide more flexibility than structured categories. Unstructured categories have a concept definition in var-

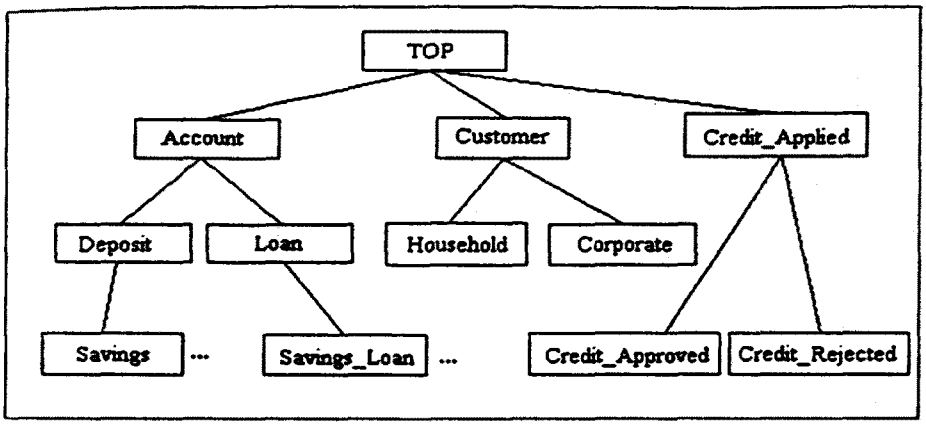

Figure 1. Example Bank DB Schema

ious forms such as weighted feature records, rules, and decision trees. In continuing with our bank database example, we define two unstructured categories including their methods of membership testing. These two categories serve as predictors of credit application likely to be accepted and rejected, respectively. The membership conditions of Credit_Accepted contain two weighted feature records where a weight is enclosed in parentheses following an attribute-value pair. The first specifies an application involving product, contribution to the bank, relationship duration, and credit line while the second involves product, vocation, and credit line. The weighted feature records were specified by the database designer. Membership testing is performed using 'WeightedFeatureTest' as specified.

```
UNSTRUCTURED CATEGORY Credit_
Accepted
  Parents Credit_Applied
    [product.name='savings_loan'(1.0),
    applicant.contribution='High'(1.0),
    applicant.relationship_duration=
      'Long'(0.8)
  credit_line<=$10,000(0.75)]
    [product.name='savings_loan'(1.0),
    applicant.vocation='Professor'(0.8)
    credit_line<=$5,000(0.75)]
  Classification By WeightedFeatureTest

UNSTRUCTURED CATEGORY Credit_
Rejected
  Parents Credit_Applied
    [product.name='savings_loan'(1.0),
    applicant.vocation='Others'(0.8),
    applicant.annual_income<=$20,000
      (0.9)]
```

Classification By Weighted Feature Test

In order to use a learning algorithm, categories are grouped into category sets. A category set is defined by specifying the member categories, learning algorithm, and classification function. For compatibility, the learning algorithm and classification function must use the same kind of concept definition. The definition of a category set depends on the kind of learning algorithm. If the learning algorithm is supervised, the member categories must be defined before the learning algorithm is executed. If the learning algorithm is unsupervised, the member categories are determined by the learning algorithm and not specified directly. The following example revises the concept definitions of Credit_Accepted and Credit_Rejected by learning through asserted instances in each category. The category set Application_Status includes the categories Credit_Accepted and Credit_Rejected.

```
CATEGORY SET Application_Status
  Root Credit_Applied
  Contains [Credit_Accepted, Credit_Rejected]
  Classification By WeightedFeatureTest
  Learning Through InstanceAlgorithm1
```

2. Information Acquisition

ExOM objects are described by their category memberships, attributes, and values. All ExOM objects belong to at least one category, called the TOP category. Therefore, if an object is not given any category specification, it is interpreted as being a member of TOP. Membership in more specific categories is subject to satisfaction of the associated feature type expressions and concept definitions. If a user indicates a more specific category of an object, the object is inserted as a member of that category if the concept definition can be satisfied. In the following example, an object Cust2 is asserted as a member of Household with values for the indicated attributes.

```
INSERT Cust2 In Household Satisfies
  [CID:= '581219-1234567', name:=
    'Hong Gil Dong', address:='Seoul',
  marrital_status:={'married', children:
    =child1, child2}, vocation:='Profes-
    sor']
```

If the precise value of an attribute is not known, a feasible value expression may be

used. The ExOM provides three kinds of feasible value expressions: range, enumeration, and cardinality. The meaning of a feasible value expression is that the true value is an element of the values denoted by the feasible value expression. Range expressions constrain the feasible values of an unknown numerical value. Enumeration expressions constrain the values of an unknown discrete value. Cardinality expressions set bounds on the number of elements of a unknown set value. In the following example, a customer is defined with a mixture of actual values denoted by the ':=' symbol and feasible values denoted by the ':~' symbol.

```
INSERT Cust3 In Customer Satisfies
  [CID: ='650105-7654321',  name: =
    'Kim Young', address: ='Seoul',
  annual_income: ~Range[ $ 10,000.. $ 20,
    000],  contribution: ~  {'Medium',
    'Low'}]
```

Upon the insertion of an object, the ExOM begin to find all categories the object belongs to as follows:

```
LET ctg_tree:CategoryTree IN
  IF object_category is not specified
  THEN object_category=TOP
  ctg_tree.parents=object_category
  ctg_tree.children=empty
  IF subcategories of object_category exist
  THEN
    For each subcategory of object_cate-
      gory
      Append(Classify(object,  subcate-
      gory), ctg_tree.children)
  RETURN ctg_tree
function Classify(object, category)=
  LET ctg_tree:CategoryTree IN
  IF Member(object, category) THEN
    ctg_tree.parents=category
    ctg_tree.children=empty
    IF  subcategories  of  category  exist
      THEN
      For each subcategory of category
        Append(Classify(object,  subcate-
        gory), ctg_tree.children)
    RETURN ctg_tree
  ELSE RETURN empty
```

X The result of this object classification process is a category tree, the uppermost parents of which is the category asserted by users. The category tree of an object is considered an interpretation for the object in the context of a given schema definitions.

3. Queries and Operators

The ExOM supports conventional queries as well as operators for learning and classification. Queries consist of a qualification with the usual relational operators and Boolean connectives as demonstrated in the following examples.

```
RETRIEVE name From Customer Satisfies
  [address='Seoul',  annual_income>=
    $100,000,
  contribution Satisfies {'High', 'Medium'}]
```

Queries can also reference feasible values. Terms involving feasible values can be evaluated as satisfying or possibly satisfying. The latter are evaluated as true if there exists an interpretation satisfying the term. Syntactically, possibly satisfying terms are marked with a question mark as shown in the following example.

```
RETRIEVE applicant.name From Credit_
Applied Satisfies
  [product_name='savings_loan' AND ap-
    plicant.property>? $1,000,000]
```

Other operators support membership testing of an object in a category and explanation about why(or why not) an object is a member of a category. As previously mentioned, when an object is inserted, category membership is tested by interpreting the object in the context of concept definitions. The user is notified of the category tree as the result of all possible classifications. If the user is more concerned about a specific category, he or she may ask for explanation about that classification.

```
INSERT credit_application_202 In Credit_
Applied Satisfies
  [application_no:=  202,  application_
    date:= 07/30/95,
  applicant:=Cust2,  product_name :=
    'savings_loan',
  linked_accounts:=Saving58375, credit_
    line:= $10,000]

EXPLAIN credit_application_202 In Credit_
Accepted
```

The result of this operation is an explanation tailored to the concept definition of Credit_Accepted. Since the concept definition is a list of weighted feature records, the explanation provides the degree of match about each attribute value as follows:

credit_application_202:Credit_Accepted

Exemplar attributes	Values	Importance	credit_application_202	Similarity
product_name	savings_loan	1.0	savings_loan	1.0
applicant.vocation	Professor	0.8	Professor	1.0
credit_line	$5,000	0.75	$10,000	0.2

→Similarity 0.76 is above the match threshold of .66

To learn inductively about a category, a user defines a data set, invokes a learning algorithm using the data set, and possibly tests the accuracy of the resulting concept definitions. In addition, a simulation experiment can be performed by repeating the following sequence:1) randomly split the data object set into a training and test set, 2) learn from the training set, and 3) test the learned concept definitions with the test set. For example, a data set is specified for the Example category set using a maximum of 200 randomly selected instances that satisfy the target query. The second operation(LEARN) invokes the learner with a randomly selected 60% of Example_data for learning and the remaining for evaluating the learned concept definition(a decision tree).

DATASET Example-Data For Example_category_set

Instances:

RETRIEVE attribute_list FROM Example_category_set

Method:Random

Maximum Size:200

LEARN Example_category_set With Example_Data Training%:60

Ⅲ. Formal Definitions of the ExOM

We present the details of the learning and classification components of the ExOM which were informally described in the previous section. We begin with formal definitions underlying ExOM databases followed by definitions of classifiers and learners.

1. Categories

As in other object_oriented models, ExOM categories provide convenient partitioning of objects for storage and retrieval.

In addition, ExOM categories contain concept definitions generated by learning algorithms and used by classification functions. In this subsection, we formally define categories and related aspects beginning with ExOM databases.

Definition 1: An ExOM database is a tuple $\langle O,C,CL,L,M\rangle$ where

O is a set of domain objects,

C is a set of categories,

CL is a set of classifiers(Section 2),

L is a set of learners(Section 3), and

$M::2^c \rightarrow CL \times L$ assigns a classifier and learner pair to a subset of categories where 2^c is the powerset(set of all subsets) of category identifiers.

Objects are pairs consisting of an identifier and a feature value. A feature value is an association of labels to values. Formally, ExOM values are drawn from the recursive domain V:

Definition 2: $V = B_0 \cup B_1 \cup \cdots \cup B_n \cup O_{ID} \cup FV \cup S$ where

$B_0, B_1, \cdots, B_n$ are base types,

O_{ID} is the set of object identifiers,

FV is the set of feature values(finite mappings from the set of labels(A) to values)

$FV = A \rightarrow V = \{fv \in A \rightarrow \mid \{a \in A \mid fv(a) \in V\}$ is finite$\}$

S is the set of set values where a set $S_i = \{v_1, v_2, \cdots,\}$ with $v_1 \in V$

A type is a set of values drawn from V. The ExOM supports type expressions formed from the base types, range types defined over a base type with an interval specification, enumerated types defined over a base type with a value specification, set types defined with a minimum and maximum cardinality, feature type expressions defined as an association of labels to type expressions, and ExOM category names. The use of a category name implicitly defines a feature type because the denotation of a category includes a feature value. Other type constructors such as function types, lists, and labeled unions can be included but are not discussed here for brevity.

Formally, types are defined as expressions that denote a subset of V. Denotation of a type expression τ with environment η is defined by $D \parallel \tau \parallel^{\eta}$ where $D \in TE^* \times Env \rightarrow 2^V$ is the semantic function for type expressions(TE) and $\eta \in T_{ID} \rightarrow 2^V$ is an environment, a mapping of type identifiers(T_{ID}) to subsets of V.

Definition 3: Denotation of a type expression is as follows:

$D \| k_i \| = B_i$ where k_i is a type constant (a base type).

$D \| E(v_1, v_2, \cdots, v_n) \| \eta = \{v_1, v_2, \cdots, \mid \exists B_i(\{v_1, v_2, \cdots, v_n\} \subseteq B_i)\}$
where $E(v_1, v_2, \cdots, v_n)$ is an enumerated type with value set $\{v_1, v_2 2, \cdots, v_n\}$

$D \| RG(\lambda, v) \| \eta = \{r_{ij} \mid \exists B_i(r_{ij}, \lambda, v \in B_i \wedge (r_{ij} \geq \lambda) \wedge (r_{ij} \leq v) \wedge (\lambda \leq v))\}$
where $RG(\lambda, v)$ is a range type with minimum(λ) and maximum(v) values

$D \| [\cdots, a_i :: \theta_i, \cdots] \| \eta = \cap_i \{fv \in FV \mid fv(a_i) \in D \| \theta_i \| \eta\}$
where $[\cdots, a_i :: \theta_i, \cdots]$ is a feature type

$D \| ST(\lambda, v, \theta_i) \| \eta = S_i$ *such that* $S_i \subseteq D \| \theta_i \| \eta \wedge (\mid S_i \mid \geq \theta) \wedge (\mid S_i \mid \leq v)$
where $ST(\theta, v, \theta)$ is a set type with minimum(λ) and maximum(v) cardinalities

$D \| C_i \| \eta = \{\langle o_{id}, fv \rangle \mid o_{id} \in C_i.O \wedge fv \in D \| C_i.ft \| \eta\}$ where

C_i is a category name, $C_i.O \subseteq O$ is the set of object identifiers of C_i, and

$C_i.ft$ is the feature type expression of C_i.

Objects are organized into categories which define conditions for object membership and possibly an interpretation function to classify objects. Formally,

Definition 4: an ExOM category is a tuple $\langle C_{id}, P, cd_i, cl, l \rangle$ where

C_{idd} is a category identifier,

P is the set of parents categories,

$cd_i \in CD$ is a concept definition of kind j,

$cl \in CL$ is an optional classifier compatible with cd_j, and

$l \in L$ is an optional learner compatible with cd_j.

Most categories are defined with only the first three components. A concept definition is a structure that is specified by a designer(or sometimes generated by a learner) and interpreted by a classifier. Because our aim is to support a broad range of learners and classifiers, we do not limit the ExOM to a single kind of concept definition. However, since there are many possible kinds of structures, we limit our focus to those which are explainable and can be generated by a machine learning algorithm. Note that we do not consider a concept definition to be a function because we want the flexibility of having multiple membership functions for a given kind of concept definition. Currently, we deal with four of the most popular structures: feature types, weighted feature values, rules, and decision trees. Formally, the set of concept definitions CD is defined below.

Definition 5: $CD=FT+WFVset+RLset+DT$ where

$FT=A\rightarrow TE$ is the domain of feature types

$WFVset=2^{WFV}$ is the powerset of weighted feature values where

$WFV=(A\rightarrow V\times W)\times C_{ID}$ is the domain of weighted feature values where

W is a real number [0..1] indicating the importance of the feature.

$RLset=2^{RL}$ is the powerset of rules where

$rl_i\in RL\wedge rl_i=\langle lhs,rhs,st\rangle$ where

lhs(left hand side) is a set of triples $\langle label, relop, scal\text{-}expr\rangle$ where

relop is a relational operator and

scal-expr is a scalar value, set of scalar values, or pair of scalar values

rhs(right hand side) is a category identifier or pair ⟨label,value⟩, and st is a real number [0..1] indicating the strength of the rule.

DT is the set of decision trees with $dt_i\in DT\wedge dt_i=\langle node, \{\langle edge, dt_i\rangle\}\rangle$

where node is a label for non-leaf nodes and a category identifier or nil for leaf nodes, and edge is a scalar value or pair of scalar values (an interval specification).

We have limited the complexity of these structures due to bias in machine learning algorithms. Because of computational complexity, machine learning algorithms are limited in the expressiveness of the concepts they can generate. For example, rules are limited to conjunctions of simple comparisons because most machine learning algorithms cannot efficiently search a space of more complex rules. Another form of bias is the structure of the input. Most machine learning algorithms use only scalar values but some use feature values, set values, and fuzzy values. In Section 3 we describe limitations on the structure of inputs used by learners. In concept definitions, we do not limit input structure as we permit a label to be an attribute name or a dot expression of the form, $label_1.label_2\cdots label_n$, where labeli refers to an attribute of $label_{i-1}$. Thus, a concept definition may reference indirect attributes of an object.

The ExOM has two kinds of categories, structured and unstructured, corresponding respectively, to closed and open views of the world. The feature type, required for structured categories as a concept definition, is interpreted according to a closed world view. In this interpretation, only objects that satisfy the feature type can be inserted into the category. Formally,

Definition 6: a structured category $\langle C_{id}, P, ft \rangle =$

$\{obj \mid \forall p_i \in P(obj \in p_i \wedge M_{ft}(obj.fv, ft))\}$

$obj.fv$ denotes the feature value of obj,

$M_{ft}: :FV \times FT \rightarrow [0..1]$ is a membership function for feature types

Unstructured categories support an open world interpretation of feature types to provide more flexibility but less uniformity. The concept definition for unstructured categories can be a feature type, weighted feature type, rules, or decision trees. Therefore, the interpretation of a concept definition is more flexible than for structured categories, and the concept definition is usually associated with a graded membership function(see Section 2). Formally,

Definition 7: Unstructured category $\langle C_{id}, P, cd_i, cl_j \rangle =$

$\{obj \mid \forall p_i \in P(obj \in p_i \wedge M_{dj}(obj.fv, cd_j))\}$ where

$M_{dj}: :FV \times CD \rightarrow Boolean \cup [0..1]$ is the membership function for concept definitions

Categories alone(either structured or unstructured) are not sufficient to integrate machine learning. Category sets are the glue between categories and machine learning. A category set defines a collection of categories sharing a learning algorithm, classifier, and concept definition. Formally,

Definition 8: a category set $\langle CS_{id}, C_p, C, cd_i, cl_j, l_j \rangle$ where

CS_{id} is a category set identifier,

C_p is a parents category, which is the domain of the category set,

C is a set of categories,

cd_i is a concept definition,

$cl_j \in CL$ is an optional classifier compatible with cd_i, and

$l_j \in L$ is learner compatible with cd_i.

In a category set, element categories are usually unstructured, and do not have additional attributes nor have any descendant categories except for those defined in the parent category. A category set corresponds to the horizontal dimension of categorization from studies of human cognition[SM81]. The horizontal dimension involves the segmentation of a concept into a collection of subconcepts that are not distinguished structurally but rather by differences of feature interpretation. Often, the set of subconcepts does not have clear cut boundaries. For example, the division

of customers into price seekers and prestige seekers does not involve any new features for the subconcepts but rather different interpretations of the features of customers. Because of this intended use, element categories in a category set inherit attributes from the parents category but may not add or change the inherited set.

2. Classifier

Classifiers use concept definitions to determine (the degree of) category membership. They also explain why a classification was a success or failure.

Definition 9: An ExOM classifier is a tuple $\langle CL_{id}, M_k, k, PM, EX_i \rangle$ where

CL_{id} is a classifier identifier,

M_k is a graded membership function of the form

$FV \times 2^{CD} \times TV \rightarrow 2^{CDi \times [0..1]}$ (individual category test) or

$FV \times 2^{Cid \times CD} \times TV \rightarrow 2^{Cid \times CDi \times [0..1]}$ (category set test)

where TV represents threshold values between 0 and 1.

k indicates the kind of concept definition,

PM is an optional set of parameters of the form $\langle label, value \rangle$ used by a membership or explanation function, and

$EX_i \in EX$ is an explanation function (see later discussion).

Graded Membership testing can be performed in two contexts depending on whether a concept definition is defined for a category or category set. In the former case, the concept definition of the category is tested resulting in a concept definition identifier and graded membership value pair(s), which are above a given threshold value. Only the category with the highest grade value can be returned. In the latter case, a set of concept definitions is tested resulting in the selected element category as well as a concept definition identifier and a graded value. Typically, graded membership functions are useful for object descriptions having feasible values or concept definitions having weights because they provide confidence information that can be used to select among alternative categories. A cutoff parameter is used to decide whether a graded membership value is large enough to justify membership. The Boolean membership function is defined as follows:

function Member (obj, CID) =

```
IF CID is a member of category set CS
LET CID_list=Empty
  For all CIDj in CS
    Append(M_g(obj.fv, CIDj.CDi, tv),
      CID_List) %{CIDj * CDi *
      GradeVal}
IsMember(CID, CID_List)
Else IsEmpty(M_g(obj.fv, CID.CDi, tv))
  %{CID*CDi*GradeVal}
```

Various kinds of membership functions can be defined in the ExOM and this flexibility enables a designer to select among alternative functions that best match the characteristics of a category of interest. For example, in classifying census forms about industrial and occupation categories, different membership functions were reported as showing higher accuracy [CMSW92]. In the current implementation, the ExOM supports classifiers, one for testing feature types of individual categories, one for testing decision trees for category sets, and two each for weighted feature values and rules. For weighted feature values, the ExOM supports a graded membership function.

Expert systems provide explanation because users must have confidence in recommended decisions. Confidence can be increased by understanding the chain of reasoning used to reach a decision as well as by an indication of how much certainty the system has in a decision. Weak explanations that result from either a misunderstood chain of reasoning or low certainty indicate a need for more information gathering by the user.

As a generator of inductive expert systems, the ExOM provides explanation about membership decisions. The basic kind of explanation answers the questions why or why not. In a rule-based expert system, the why question is answered by displaying the chain of rules used to reach a decision. In the ExOM, there is no chain of reasoning because of the nature of concept definitions. Rather, explanation functions in the ExOM return the concept definition (or part of the concept definition if disjunctive) responsible for the classification decision, a detailed explanation of the success or failure, and a certainty value if the classifier uses a graded membership function. Formally,

Definition 10: an explanation function EX_i is of the form

$FV \times 2^{CD_k} \rightarrow CD_k \times String \times [0..1]$ or

$FV \times 2^{CD_k} \rightarrow C_{ID} \times CD_K \times String \times [0..1]$

where

the output concept definition(CD_k) is

drawn from the input set(2^{CDk})

For disjunctive concept definitions, the manner in which the responsible element (CD_k) is selected depends on the kind of concept definition and membership function. If the concept definition is ordered such as a decision tree or ordered rules, the first failing or succeeding path/rule is selected. If the associated membership function is graded, the disjunct with the largest value is selected. Otherwise, the most satisfying disjunct is selected where the definition of most satisfying is the responsibility of the explanation function.

Other kinds of explanation functions are both possible and desirable. A simple variation is to return information on all categories tested for membership rather than only the most satisfying category. This is especially important if more than one category satisfies the concept definition or no categories satisfy it.

3. Learner

The ExOM supports two kinds of learning about categories. For the traditional mode of learning by being told, a database designer directly provides concept definitions using editors and browsers for each kind of concept definition. More interestingly and importantly, the ExOM supports a wide range of machine learning approaches. In this section, we describe how the ExOM supports machine learning approaches by defining the notions of a learner and learning operators.

Fundamental to every learning algorithm is the notion of a training set. A training set [AB92] is a collection of examples (attribute, value pairs) from which a learning algorithm makes inductions. The ExOM recognizes supervised training sets (Definition 11) in which every example is associated with a category label and unsupervised training sets without category labels. We require that supervised training sets be functions. In addition, we require that examples in training sets satisfy constraints of the categories in which they were drawn.

Definition 11: a supervised training set (stsi) is defined as

$$STS = FV \rightarrow C_{ID} = \{sts_i \in STS \mid (fv_j = fv_k) \Rightarrow (sts_i(fv_j) = sts_i(fv_k))\}$$

Definition 12: an unsupervised training set (ustsi) is defined as

$$USTS = 2^{FV}$$

Learning algorithms use training sets to determine concept definitions, cluster data, and discover interesting relationships. The ExOM recognizes three kinds of learning algorithms depending on the input (supervised or unsupervised training set) and output (concept definitions and clustering). A supervised concept learner (Definition 13) determines a collection of concept definitions for a supervised training set. A clustered concept learner (Definition 14) determines a grouping for an unsupervised training set and possibly concept definitions for the groupings. A discovery learner (Definition 15) determines interesting subsets of an unsupervised training set as well as concept definitions for the subsets. A clustered concept learner differs from a discovery learner in that the grouping produced by a clustered concept learner covers the training set while the grouping produced by a discovery learner need not (and typically does not) cover the training set. In addition, the discovery learner also generates concept definitions while the clustered learner need not generate concept definitions. Practically, discovery learners have very large training sets because their purpose is to find interesting patterns in large data sets. Definitions 13 through 15 summarize this discussion.

Definition 13: a supervised concept learner (scli) is an element of

$$SCL = STS \rightarrow 2^{CD}$$

Definition 14: *a clustered concept learner (ccli) is an element of*

$$CCL = UTS \rightarrow 2^{UTS} \cup (2^{UTS} \times 2^{CD})$$

Definition 15: a discovery learner(dl_i) is an element of

$$DL = UTS \rightarrow (2^{UTS} \times 2^{CD})$$

For a specific learning algorithm, its kind is implicitly defined by a number of properties that must given before it can be used. The properties of a learning algorithm include the kind of concept definition it generates, timing (incremental or batch), category space (flat or hierarchical), supervision level (supervised or unsupervised), and attribute scale (nominal and/or numeric). If the learning approach is unsupervised, a few other properties are relevant including category space size (given and/or learned), disjointness of the category space (disjoint or overlapping), and the covering of the category space (covering or non-covering). Other attributes including attribute cardinality (scalar or set) and value precision (actual or fuzzy) are not currently supported in

the ExOM because few machine learning algorithms use set values and fuzzy values. Formally,

Definition 16: an ExOM learner is $\langle L_{id}, K, T, CS, SL, AT, SZ, DIS, COV \rangle$ where

L_{id} is a learner identifier,

K indicates the kind of concept definition,

T indicates the timing(incremental or batch),

CS indicates the category space (flat or hierarchical),

SL indicates the supervision level (supervised or unsupervised),

AT indicates attribute types accepted (nominal, numeric, or both),

SZ is an optional size constraint (given or learned),

DIS is an optional disjointness constraint (disjoint or overlapping), and

COV is an optional covering constraint (cover or non-cover).

These properties cover a wide range of machine learning algorithms. Table 1 depicts prominent learning algorithms and their property values. Abbreviations have been used for property values, e.g., B means Batch and I denotes Incremental for the Timing property. Note that supervised algorithms ($SL=S$) do not have values for the size (SZ), disjointness (DIS), and covering (COV) properties.

The ExOM uses these properties to apply a learning algorithm: when to learn, what to learn, and what constrains the learner. The timing property determines whether the learner depends on the order of the training instances. If the timing is incremental, the order is significant and learning occurs after each example is presented. Training instances are presented in the order in which a learning query produces them. If the timing is batch, the order is not significant and learning does not occur until the entire training set is assembled. All learners except for ones that only cluster produce concept definitions that are limited to one of the kinds of concept definitions supported by the ExOM. For unsupervised learners, the ExOM uses the size(SZ), category space(CS), covering (COV), and disjointness(DIS) properties to determine a learner's output. For example, case based reasoners produce a hierarchical clustering(CS=hierarchical, COV=cover) which can be disjoint or overlapping, and decide on the number of clusters (SZ=learned). Knowledge discovery algorithms[FPM91] determine labels for interesting categories but do not cluster objects

Table 1. Property Values of Selected Learning Algorithms

Learning Algorithm [Reference]	Learner Properties							
	K	T	CS	SL	AT	SZ	DIS	COV
ID3[Quin86]	DT	B	F	S	Nom*	–	–	–
ID5R[Utgo89]	DT	I	F	S	Nom*	–	–	–
IB3[AKA91]	WFV	I	F	S	All	–	–	–
Unimem[GLF89]	WFV	I	H	U	All	L	Non	Cov
Cobweb[GLF89]	WFV	I	H	U	Nom	L	Dis	Cov
CN2[CN89,CB91]	Rule	B	F	S	Nom	–	–	–
ITRULE[SG92]	Rule	B	F	U	Nom	L	Non	Non

* Preprocessor transforms numeric into nominal values.

(*COV*=non-cover). Some learners are constrained by the kind of attributes in the training set. If a learner is constrained by numeric or nominal attributes, a learner can only be applied with compatible attributes.

In the ExOM, learning is considered a part of the database design process rather than a way to dynamically configure a database. This philosophy is consistent with the view that machine learning is a form of data analysis, complimentary to other forms such as discriminant analysis, clustering, and analysis of variance. Even with incremental learning algorithms where the result of learning is available after each example, we feel that a separate learning phase is still the norm because of the nature of data analysis. Generally, the results of any data analysis process must 2be carefully considered before putting those results into practice. Because of this philosophy, the ExOM supports a learning phase separated from using the learned knowledge to classify new objects. In the learning phase, the ExOM supports executing a learning algorithm, browsing of the learned knowledge, capturing test results, and performing experiments.

The ExOM provides a number of operators to support the learning process. The learning process begins with the definition of a data set for the learning phase using the DATASET operator.

The user specifies the category set, relevant attributes in the data set, the sampling method(random or systematic), the sample size, and the data source(external

file, generated data set, or data base query using existing or new instances). The sampling method determines how the data set is constructed from qualified instances of the data source. We assume that there are a menu of possible random and systematic sampling methods from which to choose. If the learner is unsupervised, the data source is usually a query selecting instances from the root of the category set. If the learner is supervised, each instance in the data set must be labeled with its category. If the constituent categories in the category set are already populated, output labeling can be accomplished by a query that selects from the constituent categories. Otherwise, the user can interactively provide the category labels or an external data set can be used. As a result of the DATASET operator, the specifications about the data set are stored for later use by the LEARN operator. The LEARN operator imports the data set, invokes the specified learner, and saves the result of the learning. The user specifies a data set identifier, an execution time (time point or interval), and split percentage. If the learner is unsupervised, the data set is split into a training set that is used by the learner and a test set that is used to test the learned concept definitions. The split percentage indicates the percent of the data set used in the training set. The split percentage is 100% for unsupervised learners. The EXPERIMENT operator repeatedly executes the LEARN sequence for a specified number of trials and accumulates the results. It is used only for supervised learners.

4. Consistency of ExOM Databases

In strongly-typed object-oriented databases, taxonomic reasoning is used as a learning and classification method. Taxonomic reasoning provides automatic classification of a new concept in a hierarchy of concepts and ensures the consistency and minimality of the conceptual schema. To ensure efficient subsumption testing (the key algorithm underlying taxonomic reasoning), the expressive power of class description languages must be severely limited. Unfortunately, ExOM concept definitions extend beyond the expressive power of class description languages with efficient taxonomic reasoning. Thus, efficient taxonomic reasoning cannot be guaranteed for ExOM concept definitions leading either to potential consistency problems without taxonomic reasoning or unacceptable performance of taxonomic reasoning.

In this section, we present properties of ExOM databases and demonstrate how to ensure the consistency of ExOM databases with limited use of taxonomic reasoning. We first demonstrate efficient taxonomic reasoning for ExOM category definitions. We then present several propositions that define the relationship between ExOM concept definitions and feature type expressions. Principally, we define conditions under which a category definition logically implies another category definition when both are defined as a combination of a feature type expression and concept definition. This result leads to practical strategieg to ensure the consistency of ExOM databases which are presented in the last subsection.

(1) Taxonomic Reasoning for ExOM Categories

We demonstrate that ExOM category definitions can be mapped to equivalent expressions in the language FL* [BS92]. FL* is an extension of the frame description language of FL- [BL84] which was the first frame description language in which a sound, complete, and tractable subsumption testing algorithm was found. FL* extends FL- with constructs to define minimum and maximum cardinality, disjointness between primitive categories, and attributes with value sets. The translation from category definitions into FL* uses the following rules.

1. AttrName::Type becomes (ALL AttrName Type) (NR AttrName 01)
2. AttrName::ST(min,max,Type) becomes (ALL AttrName Type) (NR AttrName min max)
3. ($A1::TE1,\cdots,A_n::TE_n$) becomes AND ($A1::TE1,\cdots, A_n::TE_n$)
4. $E(V_1,\cdots,V_n)$ becomes Dm where Dm= $(V_1,\cdots,V_n)$
5. PARENTS $C_1,\cdots,C_n$ becomes AND($C_1, \cdots,C_n$)

In the first two rules, the translation can be into a role or attribute depending on the kind of type. If the type is derived from a basic type (range or enumerated), the translation is to an FL* attribute definition (NRa); otherwise, it is to an FL* role (NR). The fourth rule translates the enumerated type into a unique domain name and then generates a value domain declaration. Since FL* does not support arbitrary enumerated types, we restrict enumerated types in the ExOM to disjoint value sets of a base type. The fifth rule translates the part of an ExOM category definition that is not part of a feature type

expression. The ExOM does not support primitive concepts nor disjointness between primitive concepts as in FL* although these features can be added without difficulty. The ExOM emphasizes learning and classification of concept definitions that cannot be tractably analyzed for taxonomic reasoning.

Because ExOM category definitions can be translated into FL* concepts in polynomial time using the above rules, the algorithms and results in [BS92] apply. In particular, the polynomial time algorithm SUBS and CLASSIFY in [BS92] can be used to perform subsumption testing and automatic classification of a new category definition, respectively. The ability to use these algorithms is a straightforward result which we utilize in the following subsection. In addition, we rely on the same definition of subsumption and the derived Theorem in [BS92] that are restated in Definition 17 for convenience.

Definition 17: P SUBSUMES C iff $E \| C \| \subseteq E \| P \|$ where

E is the extension function for FL*.

Theorem: P SUBSUMES C iff SUBS(P, C)

(2) Relationship of Concept Definitions to Feature Type Expressions

In the ExOM, concept definitions augment traditional class definitions. To ensure consistency of an ExOM schema, the concept definitions must be analyzed by the SUBS and CLASSIFY algorithms. In this subsection, we define a well behaved learning process and demonstrate resulting properties of learned concept definitions. These properties underlie the consistency of ExOM databasesw.

A well behaved learning process uses background knowledge and a training set to generate concept definitions for a collection of categories. Background knowledge typically consists of a collection of feature type expressions in concept definitions from parent categories. Therefore, the background knowledge is the conjunction of the feature type expressions including inherited feature type concept definitions. Recall that a supervised training set is a collection of objects where each object consists of a set of values for the predictor attributes and a category identifier representing one of the categories in which to learn. The training set is produced by a data generator where each object generated adheres to its associated background knowledge. The training set is used by a supervised concept learning algorithm to

generate minimally consistent concept definitions. Formally, a data generator and minimally consistent concept learner are defined below.

Definition 18: $DG=2^{BG}\rightarrow STS$ where $bg_i\in BG=obj.fv\wedge fte_1\wedge\cdots\wedge fte_n$

fte_k represents the feature type expression of a parents category k

Therefore, $\forall dg\in DG, \forall bg\in BG, \forall sts_i\in dg(bg)$ $(Satisfies(sts_i, bg_i))$ where

i represents the category of training example set sts_i.

Definition 19: A supervised concept learner $scl_i\in STS\rightarrow 2^{CD}$ generates minimally consistent concept definitions iff

$\forall sts\in STS,\ \forall cd_j\in scl_i(sts)\ (\exists sts_k\in sts(Satisfies(cd_j, sts_k)))$.

Definitions 18 and 19 restrict the data generation process and the output of supervised concept learners. Definition 18 is a simplification of a data generation process. An actual data generation process usually involves either an historical or hypothetical data set. In the former case, the objects of the training set are taken directly from an existing category in a database. In the latter case, hypothetical objects are defined usually in consultation with a group of experts. We assume that adequate care is exercised so that each object in the training set satisfies its associated background knowledge in the form of a feature expression possibly augmented with inherited concept definitions from parent categories. Definition 19 establishes that concept definitions generated by a learner need not fully explain the training set. However, most examples will satisfy their associated concept definition.

Two things need to be noted. First, in a well-behaved learning process, a supervised concept learner generates concept definitions that are consistent with the background knowledge (feature type expression and inherited concept definitions) of the category. By Definition 19 each concept definition is satisfied by at least one object in the training set and by Definition 18, each example object in a training set satisfies its background knowledge. Thus, the background knowledge is consistent with the generated concept definition because there is at least 1 training object that satisfies both.

Second, in a well-behaved learning process, a supervised concept learner generates concept definitions that do not necessarily imply the background knowledge of their category. A learned concept definition

may only include a proper subset of the attributes from a feature type expression. Because supervised concept learners attempt to generalize their training set. One major part of generalization is to remove attributes that do not significantly contribute towards good classification performance. Further, supervised concept learners prefer simple concept definitions (few terms) to more complex ones, other things being equal. Therefore, a concept definition containing a proper subset of the attributes cannot possibly imply a feature type expression with conditions on all the attributes.

Definitions 18 and 19 along with these simple relationships are not sufficient to extend the subsumption test in Definition 17. To establish a subsumption test, there must be a rule to guarantee consistency between the learned concept definition of a category and the feature type of a parent category. Otherwise, there is nothing to prevent conflicts because the feature type of structured categories is usually defined by the database designer while the concept definition of unstructured categories is generated by a learning algorithm. To prevent conflicts, we assume that an attribute value cannot be over-generalized if it appears in the feature type of parent categories. In practice, this restriction should not be a problem because the concept definition typically imposes tight restrictions on any relevant attribute. The Attribute Redefinition Rule is formalized in Definition 20 below.

Definition 20: Attribute Redefinition Rule:

$\forall a_i \in Dom(cd_c)$ $(a_i \in Dom(ft_p) \rightarrow ft_p(a_i) \geq ft_c(a_i))$ where

C is a child of category P,

Dom is a domain function,

cdC is the concept definition of category C, *and*

$ft_X(a_i)$ is the value of attribute ai in the feature type expression of category X

It is concluded that a parent category subsumes a child category if concept definitions of a child category are generated by restricted concept learners, satisfying the Attribute Redefinition Rule. Formally,

SUBS(ft_P, ft_C) iff

1. $ft_C = scl_i(dg_i(ft_P))$ where $scl_i \in$ SCL and $dg_i \in$ DG,
2. satisfies the Attribute Redefinition Rule with respect to ft_P.

For maintaining a consistent ExOM da-

tabase, we isolate the non-provable part of the schema (unstructured category) from the provable part (structured category). The taxonomic reasoner verifies the provable part of the schema.

When adding a new unstructured category, the attribute redefinition rule and subsumption testing is performed to test consistency with its parent categories. If a learner is invoked for a new category, a background knowledge check is enforced for every data set used where the background knowledge is the composition of feature type expressions inherited from parents concept definitions. Every subsumption relationship given by a designer is verified.

Ⅳ. Prototype Implementation

The architecture of the ExOM contains three major components: query, classifier and learner managers. The Query Manager is the interface to the objects, categories, and types of an ExOM database. Users can retrieve objects by their content, define categories and types, and indirectly invoke Classifier Manager when asking about category membership. The Classifier Manager supports the definition and retrieval of classifiers as well as testing membership of an object in a given category and explaining its decision. The Learner Manager acquires objects, invokes learning algorithms, and creates concept definitions. In the case of unsupervised learning, it also creates categories and inserts objects into those categories. The knowledge base contains definitions of categories, types, concept definitions, learners, and classifiersw.

A prototype with most of the features described in Sections 2 and 3 has been implemented in Smalltalk/V Windows. In the prototype, the Query Manager has been divided into an object manager for manipulating and querying objects, a type manager for defining named type expressions, and a category manager for defining categories and teacher supplied concept definitions. The Classifier Manager has been divided into managers for concept definitions and associated membership and explanation functions (classifiers). The Learner Manager has been divided into a learner manager for registering learners, a category set manager for grouping categories with a learner and classifier, a data set manager for creating data sets from queries, external files, or a data generation program, and an experiment manager for executing the learn, test, and experiment

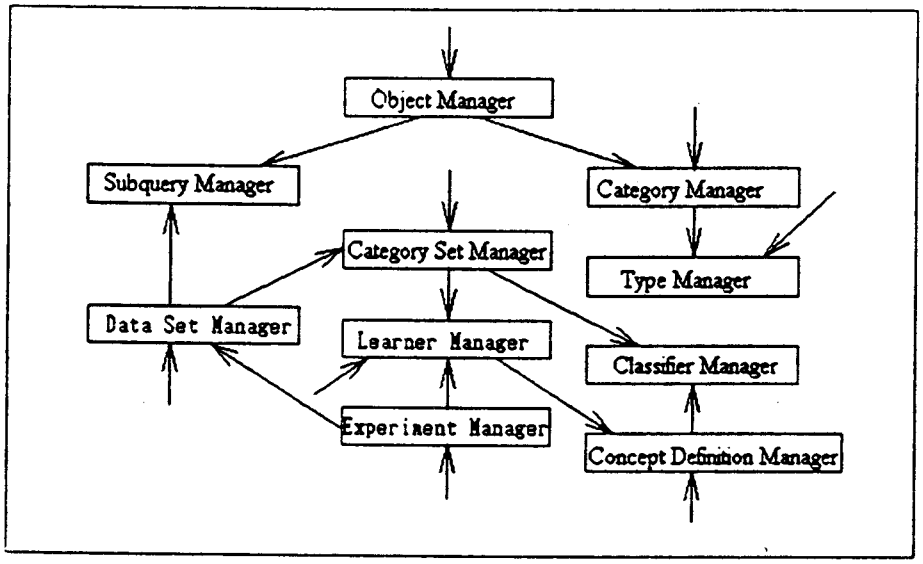

Figure. Flow among ExOM Managers

operators. All three concept definitions (weighted feature records, weighted rules, and decision trees) have been implemented. For the first two, there is an associated classifier for single categories (teacher supplied) as well as for category sets (learner generated). Several learning algorithms have been implemented: IB3 [AKA91], CN2 [CN89, CB91],and several versions of ID3 [Quin86]. The ID3 variations were implemented in C, and an interface to the ExOM is provided in the experiment manager. In addition, a data set generator as described in [Biss91] has been implemented in C, and an interface is provided in the data set manager. A number of features have been omitted due to time restrictions including feasible values, the possibility operator, limited query capabilities (only conjunctive semi-joins), support for unstructured categories, and unsupervised learning algorithms. The prototype is rather slow because it was built without an underlying database engine. We are in the process of converting the object and category manager to a suitable object-oriented database engine to improve performance.

The ExOM prototype is invoked by interactively navigating among a collection of window interfaces (one for each manager). Figure 2 depicts the flow among interface managers. For flexibility in defining a knowledge base, links be-

tween managers are supported in the form of popup menus. For example when defining a category set, the user can easily branch to the learner and classifier managers. The arrows without a source mean that the manager can be invoked directly from the ExOM main menu. The subquery manager (not mentioned above) handles category navigation in queries.

The prototype implementation comprises about 45 new classes, extensions to 10 existing classes, 200 methods, and 15,000 lines of code including comments. As with many window-based programs, much of the code is interface management.

V. Related Works

The ExOM has been inspired by work in organization theories and knowledge discovery. Work about information processing in organizations ([DW84] and DL86]) and organizations as loosely coupled systems [OW90] has been a general motivation of our research. Work in knowledge discovery has been a more specific influence on our work. Knowledge discovery has been defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data." [FPM91]. This definition extends notions of discovery in machine learning, databases, statistics, and visualization. The ExOM is a tool for discovering knowledge in an object-oriented database using a broad range of learning and classification approaches.

The notion of a category in the ExOM has been influenced by work in cognitive psychology and machine learning. Researchers in cognitive psychology have recognized three views of how people represent concepts [SM81]: classical, probabilistic, and exemplar. In the classical view, a concept is a collection of features which are necessary and sufficient for classification. In the probabilistic view, a concept definition has a notion of uncertainty for classification. In the exemplar view, a concept is represented by prototypical objects. The ExOM supports these three kinds of concept definitions to the extent they have been computationally developed in the machine learning literature. In particular, the instance based learning algorithm[AKA91], implemented in the ExOM prototype, combines the probabilistic and exemplar views of categories. In the design of the ExOM, we specifically considered case - based reasoning ([Kolo91] and [GLF89]), decision tree induction [Quin86], apprenticeship learning

[PBH90], and rule induction algorithms [CN89] as well as instance based learning [AKA91].

From a data model perspective, the ExOM is an extension of object-based models. Object-based models such as [LTP86], [Lieb86], and [Scio89] provide a more flexible notion of inheritance than class-based data models such as [MCB90] and [Kim90]. The object-based models use delegation instead of inheritance and separate object hierarchies from class hierarchies. The ExOM combines both notions (class-based and object-based) through structured and unstructured categories. The focus in this paper is the integration of learning and classification with categories rather than flexible inheritance. Inheritance and other aspects of the ExOM are reported in [Jung92].

The ExOM extends the reasoning capabilities of deductive databases [HW92] and utilizes learning for more than monitoring database designs. Through our experience with SEMLOG [SJW91], we found that deductive reasoning alone cannot support knowledge discovery in organizations. Likewise, we feel that approaches emphasizing subsumption reasoning in query processing ([BBMR89] and [BGN89]) and deduction in classification [KK90] are too narrow. The use of learning and classification in the ExOM is not limited to database design as described in [ISW92], [BW85], and [LM89]. Rather, the ExOM is a general purpose tool in which the user can determine the application of the learning and classification components including to monitor a database design. The ExOM is most closely related to the INLEN system [KMK91] that combines inductive and deductive reasoning with a relational database system. INLEN features some interesting knowledge generation operators including discriminant description finding, consistency and completeness checking, and relevance checks of attributes and examples. These operators would be useful additions to the ExOM. However, the ExOM is oriented towards a broader range of learning and classification methods, integration with an object-oriented database, and explanations of classifications and learning.

VI. Conclusion

We presented an overview of the Extensional Object Model (ExOM) and described in detail its support for classification and learning. Support for machine learning is divided into two components: classifiers

and learners. Classifiers test membership of objects in categories and explain their reasoning. Learners derive new knowledge in the form of concept definitions, categories, and object membership in categories. Because of the diversity of approaches in machine learning, the philosophy of the ExOM is to provide a broad range of classification and learning methods rather than a single canonical method. We demonstrated support in the ExOM for several kinds of concept definitions as well as the parameterization of learners and classifiers enabling new classifiers and learners to be introduced. The learning and classification components are integrated through grouping of categories for learning and classification, operators that support learning as a phase of database design as well as classifying and explaining object membership. To ensure the consistency of ExOM databases, properties relating concept definitions and feature type expressions were presented which led to several strategies for ensuring consistency. We briefly depicted a prototype implementation in Smalltalk/V that supports most of the features described in the paper.

We envision a number of short and long term extensions to the ExOM to improve its capability as a decision making tool. An important short-term extension is an expanded experiment operator. The current experiment operator supports iteration of training and testing with classification accuracy as the performance measure. To satisfy more detailed decision making situations, the experiment operator should support properties of attributes and categories (e.g., costs and noise levels), various performance measures (e.g., information theoretic measures and costs), data set factors (e.g., noise generation and skewness), and experimental factors (e.g., learning algorithms and cost variance levels). These extensions would support the experiments described in recent research [DM93, MM92] that evaluate propositions about system value rather than accuracy. Further extensions of the learn, evaluate, and experiment operators are necessary for unsupervised learning. However, these extensions are dependent on the development of standard evaluation approaches in the machine learning community.

Sensitivity analysis capabilities are long-term extensions to the ExOM that are necessary to make machine learning a widely-used analysis tool. In order to better understand a concept definition, a decision maker may want to know the effect on the output when an attribute changes

its value such that one or more terms in a concept definition change state (True to False or vice-versa). If the concept definition is deterministic, an explanation function can be implemented as a form of Boolean sensitivity analysis based on the notion of a Boolean derivative [Blan90]. If the concept definition uses weights, the Boolean derivative does not directly apply although the derivative of fuzzy valued functions might [Blan90]. A second kind of sensitivity analysis is output oriented. A decision maker may want to know how to change the values of the attributes to achieve a desired change of outcome. For example, if the output is the prediction of a category representing a firm's bond rating, the decision maker may want to know how to improve the bond rating from a poor category to a better category. Further, if there are costs associated with changing the states of attributes, the decision maker may want to know the least cost way to change categories.

With these extensions and further usage, the ExOM can become an important tool to explore the integration of machine learning and object-oriented databases. We believe that this integration can increase the value of object-oriented databases and lead to improved decision making through wider usage of machine learning.

참 고 문 헌

[AB92] Anthony, M. and Biggs, N. *Computational Learning Theory*, Cambridge University Press, 1992.

[AKA91] Aha, D., Kibler, D. and Albert, M. "Instance-Based Learning Algorithms," *Machine Learning*, 6, 1991, 37-66.

[BBMR89] Borgida, A., Brachman, R., McGuinness, D., and Resnick, L. "CLASSIC: A Structural Data Model for Objects", in *Proc. ACM SIGMOD Conference*, May 1989, Portland.

[BGN89] Beck, H., Gala, S. and Navathe, S. "Classification as a Query Processing Technique in the CANDIDE Semantic Data Model," in Proc. *IEEE Data Engineering Conference*, February 1989, Los Angeles, CA.

[Biss91] Bisson, H. "Evaluation of Learning Systems: An Artificial Data-Based Approach," in *Proc. European Working Session on Learning*, Y. Kodratoff (ed.), Springer-Verlag, Portugal, March, 1991, pp. 463-481.

[Blan90] Blanning, B. "The Sensitivity Proper-

ties of Hierarchical Logic-Based Models," *Decision Support Systems* 6, 1990, 89–97.

[BS92] Bergamaschi, S. and Sartori, C. "On Taxonomic Reasoning in Conceptual Design," *ACM Transactions on Database Systems* 17, 3 (September 1992), 385–422.

[BW87] Borgida, A. and Williamson, K. "Accommodating Exceptions in Databases and Refining the Schema by Learning from Them," in *Proc. of the 11th International VLDB Conference*, August 1985, Stockholm, pp. 72–81.

[CB91] Clark, P. and Boswell, R. "Rule Induction with CN2: Some Recent Improvements," in *Proc. Machine Learning–European Working Session on Learning*, Porto, Portugal, Springer–Verlag, March 1991, pp. 151–163.

[CN89] Clark, P. and Niblett, T. "The CN2 Algorithm," *Machine Learning* 6, 4, 1989, 261–283.

[CMSW92] Creecy, R., Masand, B., Smith, S., and Waltz, D. "Trading MIPS and Memory for Knowledge Engineering," *Communications of the* ACM 35, 8, (August 1992), 48–64.

[CW90] S. Ceri and Widom, J. "Deriving Production Rules for Constraint Maintenance," in *Proc. of the Sixteenth International Conf. on Very Large Data Bases*, Brisbane, Australia, August 1990, pp. 566–577.

[DW84] Daft, R. and Weick, C. "Towards a Model of Organizations as Interpretation Systems," *Academy of Management Review* 9, 2, 1984.

[DL86] Daft, R. and Lengel, R. "Organizational Information Requirements, Media Richness and Structural Design," *Management Science* 32, 5, 1986.

[DM93] "Inductive Expert System Design: Maximizing System Value," *Information Systems Research* (to appear), August 1993.

[FPM91] Frawley, W., Piatetsky–Shapiro, G., and Matheus, C. "Knowledge Discovery in Databases: An Overview," in *Proc. First International Conference on Knowledge Discovery and Databases*, October 1991, New York.

[GLF89] Gennari, J., Langley, P., and Fisher, D. "Models of Incremental Concept Formation," *Artificial Intelligence*, 40, 1989, 11–61.

[HW92] Hansen, E. and Widom, J. "Rule Processing in Active Database Systems," *in Advances in Database and Artificial Intelligence*, JAI Press, Greenwich, Connecticut, 1992.

[ISW92] Ioannidis, Y., Saulys, T., D. and Witsitt, A. "Conceptual Learning in Database Design," *ACM Transactions on Information Systems* 10, 3, (July 1992), 265–294.

[JMW92] Jung, C., Mannino, M., and Whinston, A. "An Extensional Model of Objects," in *Proc. of the 3rd International Conference on Knowledge Systems for Manufacturing and Engineering*, March 1992, Lyon, France.

[Jung92] Jung, C. A *Framework for Computer-Supported Interpretation Systems*, Ph.D. Dissertation, The University of Texas at Austin, Department of Management Science and Information Systems, May 1992.

[Kim90] Kim, W. "Object-Oriented Databases: Definition and Research Directions," *IEEE Transactions on Knowledge and Data Engineering* 2, 3, September 1990.

[KMK91] Kaufman, K., Michalski, R., and Kerschberg, L. "Mining for Knowledge in Databases: Goals and General Description of the INLEN System," in *Proc. First International Conference on Knowledge Discovery and Databases*, October 1991, New York, pp. 449-462.

[Kolo91] Kolodner, J."Improving Human Decision Making through Case-Based Reasoning," *AI Magazine*, American Association of Artificial Intelligence, Summer 1991, 52-67.

[LM89] Li, Q. and McCleod, D. "Object Flavor Evolution Through Learning in an Object-Oriented Database System," in *Proc. of the 2nd International Conference on Expert Database Systems*, L. Kerschberg (ed.), 1989, Benjamin Cummings, Menlo Park, CA, pp. 469-495.

[Lieb86] Lieberman, H. "Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems," in *Proc. OOPSLA Conference*, October 1986.

[LTP86] Lalonde, W., Thomas, D., and Pugh, D. "An Exemplar Based Smalltalk," in *Proc. OOPSLA Conference*, October 1986.

[MCB90] Mannino, M., Choi, I., and Batory, D. "The Object-Oriented Functional Data Language," *IEEE Transactions on Software Engineering* 16, 11, November 1990, 1258-1272.

[MG90] McCann, J. and Gallagher, J. *Expert Systems for Scanner Data Environments*, International Series in Quantitative Marketing, Kluwer Academic Publishers, 1990.

[MM92] Mookerjee, V. and Mannino, M. "Redesigning Case Based Reasoning Algorithms to Reduce Information Acquisition Costs," Technical Report, Department of Management Science, University of Washington, September 1992, submitted for publication.

[PBH90] Porter, B., Bareiss, R., and Holte, R. "Concept Learning and Heuristic Classification in Weak-Theory Domains," *Artificial Intelligence Journal*, 45, 1990.

[OW90] Orton, J. and Weick, K. "Loosely Coupled Systems: A Reconceptualization," *Academy of Management Review* 15, 2, 1990.

[Quin86] Quinlan, J. "Induction of Decision Trees," *Machine Learning*, Vol. 1, 1986, 81-106.

[Scio89] Sciore, E. "Object Specialization," *ACM Transactions on Information Systems* 7, 2, 1989.

[SG92] Smyth, P. and Goodman, R. "An Information Theoretic Approach to Rule Induction from Databases," *IEEE Transactions on Knowledge and Data Engineering* 4, 4 (August 1992), 301-316.

[SM81] Smith, E. and Medin, D. *Categories and Concepts*, Cambridge University Press, 1981.

[SJW91] Stansifer, R, Jung, C., and Whinston, A. "SEMLOG: A Multiparadigm Language for Conceptual Modeling," in *Recent Developments in Decision Support Systems*, NATO ASI Series F: Computer and System Sciences, C. Holsapple and A. Whinston (eds.), Spring-Verlag, 1991.

[Utgo89] Utgoff, P. "Incremental Induction of Decision Trees," *Machine Learning* 4, 1989, Kluwer Academic Publishers, 161-186.

[Wegn90] Wegner, P. "Concepts and Paradigms of Object-Oriented Programming," OOPS Messenger 1, 1 (August 1990), 7-87.

◇ 저자소개 ◇

저자 정철용은 현재 상명대학교 경영학과 조교수로 재직중이다. 그는 서울 대학교 경제학과에서 학사학위, 미국 University of Washington에서 경영학 석사학위, 그리고 University of Texas at Austin에서 경영정보학 박사학위 를 받았다. University of Texas at Austin에서 강사로 재직하였으며, 한국 금융연구원 부연구위원을 역임하였다.
주요 관심분야는 지능형 데이터베이스, 객체지향적 시스템 분석 및 설계, 전자교역 등이며 주로 금융정보시스템 분야로 응용하고 있다.