經營科學
第15卷 第2號
1998年 11月

A Framework for Intelligent Data Interpretation System in Organizational Computing

Chul Yong Jung*

■ Abstract ■

One of organization's generic functions is the interpretation of events to carry out decision- making activities. In Intelligent Data Interpretation System(IDIS), interpreting is computationally modeled as classification of new data into categories having similar features. We define the Extensional Object Model(ExOM) as a formalism for IDIS. In ExOM, objects and categories are loosely coupled to provide flexibility for both object description and category definition in data gathering and interpretation process. Objects are classified inductively based on exemplars of categories as well as deductively based on category structures.

I. Introduction

In recent years organizational computing has received a great deal of attention from both computer and organization scientists because of the increasing strategic importance of information technology in an organization's success. Computer systems are expected to play more important roles in supporting the ongoing activities of organizations and therefore, expected to acquire more of the characteristics of organizations. A few examples are(group) decision support systems, executive information systems, computer-supported cooperative/collaborative work, and knowledge management systems. This paper proposes a framework for intelligent data interpretation systems(IDIS), based on the model of organizations as loosely-coupled interpretation systems.

From the database point of view, we integrate AI techniques with the object-oriented model to meet new requirements for database management systems in advanced applications such as IDIS. Thus we explore the possibilities of knowledge discovery or inductive learning, as well as the deductive capabilities, in database framework. Both the deductive and inductive approaches are integrated in our work.

* 상명대학교 경영학과

In our view, this integrated approach will become the basis for the next generation of organizational information systems, providing the capabilities envisioned in organization theories.

Ⅱ. Background

2.1 Organization as Data Interpretation System

The nature of organizational 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[7] proposed a model of organizations as interpretation systems, which scan the environment for information, interpret scanned information, execute actions, and learn from the feedback of actions. Scanning is defined as the process of monitoring the environment to collect data either through formal data collection systems or personal contacts. Interpretation gives meaning to data. Shared understanding and collective cognitive maps among members of organizations are developed through the process of organizational interpretation. Organizational learning is the process by which knowledge about action-outcome relationships between organization and environment is gained.

Also, Levitt and March[17] observed that behavior in an organization is based on routines, which are based on interpretations of the past experience and adapt to experience incrementally in response to feedback about outcomes. So, organizations are seen as learning by encoding inferences from history in routines that guide behavior.

2.2 Organization as Loosely Coupled System

The concept of organizations as loosely coupled system is widely adopted to explain the fact that organizations appear to be both determinate, closed systems searching for certainty and indeterminate, open systems expecting uncertainty[25]. The fact that elements in organizational systems are linked together and preserve some degree of determinacy is captured by the word *coupled*, while the fact that these elements are also independent with each other and preserve some degree of indeterminacy is captured by the word *loosely*.

The direct results of loose coupling are modularity and autonomy. Modularity, which is also widely emphasized in software design, can be attained by using loose coupling instead of tight coupling. Autonomy is the freedom to act on its own. March suggested that loose coupling creates autonomy for information gatherers[20].

2.3 Motivation and Requirements for IDIS

The motivation for intelligent data interpretation in organizations can be summarized as follows: information overload, complexity in interpretation, and volatile organizational memory on interpretation.

Why do organizations process information? Daft and Lengel suggested an answer to this question: organizations process information to reduce the uncertainty and equivocality they face from their technology, interdepartmental relations, and environment[6]. Galbraith explained the observed variations in organizational form based on the amount of information needed to reduce task related uncertainty and thereby

attain an acceptable level of performance[9]. Equivocality means ambiguity, the existence of multiple and conflicting interpretations about data[35].

The requirements for IDIS from the perspective of Daft and Lengel's two information contingencies, uncertainty and equivocality, are enumerated as follows:

- **Loose coupling between data and interpretation** To give discretion to both information gatherers and interpreters, loose coupling between scanning and interpretation is desired. It can provide the flexibility in both describing data and interpreting, while preserving the nature that interpretation may guide the collection of information.
- **Imprecise descriptions of data** In business environments, there are a lot of cases in which precise data are not available. Sometimes, even if they are available, it may cost too much to pursue information gathering. Therefore, we need to compare the benefit with the cost of getting precise data. When we can infer some needed data from existing data, or when we can do interpret data, to a certain extent, with imprecise descriptions, it may be better to go with imprecise data considering information gathering costs.
- **Multi knowledge bases for interpretations** To interpret an event in the business context, the system needs various types of knowledge in a unified framework. Managerial decision-making typifies a task in which interpretation depends on past experiences(cases) as well as general principles (rules). A manager familiar with only financial principles but ignorant of any past cases would be critically handicapped in the task of making, anticipating, and evaluating interpretations.
- **Hybrid reasoning** Because different types of knowledge are used in IDIS, hybrid reasoning is required for IDIS. Extensive work done by people in the deductive object-oriented database field may be viewed as an extension of object-oriented database (OODB) to include deductive reasoning as a problem solving tool. From our experience in SEMLOG [32], which is a multi-paradigm language for conceptual modeling, deductive reasoning in itself is not enough to interpret business events.
- **Organizational learning** IDIS needs a mechanism to enhance organizational learning through feedback for the results of interpretation from the environment. It develops common understanding among members in organizations and enhances knowledge sharing.
- **Knowledge discovery in database** The rapid growth of data and information bases has resulted in a greater need for knowledge-extracting from databases[21, 22, 36].

2.4 Framework for IDIS

We extend traditional object-oriented data base concepts to incorporate loose coupling between objects and categories. The interpretation of an object in IDIS is computationally modeled as the classification process of that object into categories(Figure 1). Objects are interpreted and classified into categories when they are entered into the database. For example, business forecasting is modeled as the process of interpreting(classifying) the economic indicators

(an object) to be an instance of a certain business cycle(a category). Therefore, objects (data) and categories(interpretation) are *loosely coupled*, which is compensated by the built-in classifier in IDIS.

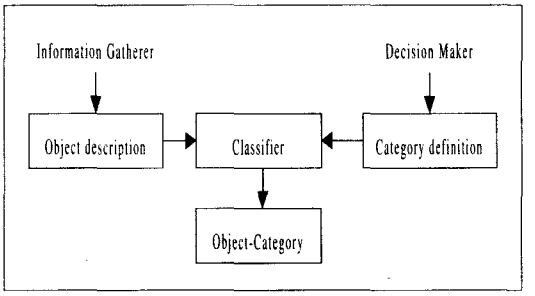

Figure 1. A Framework for IDIS: Classification as Data Interpretation

We propose a new data model, called Extensional Object Model(ExOM), as formalism for the base data model of IDIS, based on the category theory in cognitive science[27, 30, 31, 33]. It consists of objects, categories, classifiers (which map each object to categories), and a domain knowledge-base. The main benefit of this approach is its flexibility in describing objects and defining categories: new information on an object can be added over time without structural restriction, and the schema can be defined extensionally as well as intensionally and also can be easily modified to reorganize the database. This flexibility is mainly due to the loose coupling of objects and categories in the sense that object descriptions and category definitions can be done independently, while category membership is determined through reasoning by a classifier.

When the description of an object is given to the system, the subsumption reasoner classifies the object into structure-based categories if the feature description of the object satisfy the feature constraints implied by the category definition. If an exemplar-based category, which is represented by the prototypical objects(called exemplars) of a concept, is further defined within the structure-based category, the case- based reasoner evaluates the similarity of the object with the exemplar to determine the membership [26]. The rule-based reasoner is evoked when knowledge-based matching is needed for two seemingly different, but semantically equivalent, features. Queries are also processed as the same interpretation process. Classification is justified if all the structural conditions of a category are satisfied with the features of an object. Similarly, it can be justified by the similarity of the object to an exemplar to which the same interpretation is applied.

Ⅲ. Informal Overview of the ExOM

The ExOM introduces an extensional category representation scheme into database modeling. Rather than pre-defining the structures of objects in advance, users are allowed to describe an object flexibly and the system maintains a model of how to classify described objects according to the definition of categories. Therefore, the classification burden is taken from users to the system when it is ambiguous to be done. Because no unique fixed structure is predefined for objects, each object can be described more flexibly, reducing information loss during the knowledge acquisition process.

In the ExOM, categories are defined either intensionally by necessary and sufficient conditions in terms of features or extensionally by

known typical instances of the category: the former we call *structure-based* and the latter *exemplar-based*. In an extensional category representation, the definition of a category is implicit in its instances: no necessary and sufficient definition needs to be provided. A category concept is learned simply by storing exemplar objects in the category.

3.1 Example database

We illustrate the general features of the ExOM by the following example database from marketing information systems. The long-term success of any organization depends on its ability to interpret the needs of its customers and to deliver a product which meets those needs better than the competitor's. Therefore, the classification of consumers, products, or markets into categories on the basis of their characteristics, which is called *market segmentation*, is one of the most important processes in strategic market planning.

Let's take an automobile manufacturer, who produces several types of vehicles with various models. In case of cars, for example, a manufacturer produces luxury cars, family cars, sports cars, mid-sized cars and compact cars, all with a wide range of options. Also, we want to segment customers into categories according to their needs and potential purchasing power such as low-price seeker, brand switcher, prestige lover, and so on. Figure 2 shows the schema of this imaginary database. Note that these categories do not have clear-cut boundaries at the lower level, in which case conventional data models are not flexible enough to represent and to learn these category definitions. This is also a typical situation where one starts with an incomplete knowledge of a customer and incrementally explores the details. Possible attributes for use in the classification of customers include income, assets, education, occupation, self- confidence, number of stores visited, perceived quality differentiation among brands, and so on, but are open-ended.

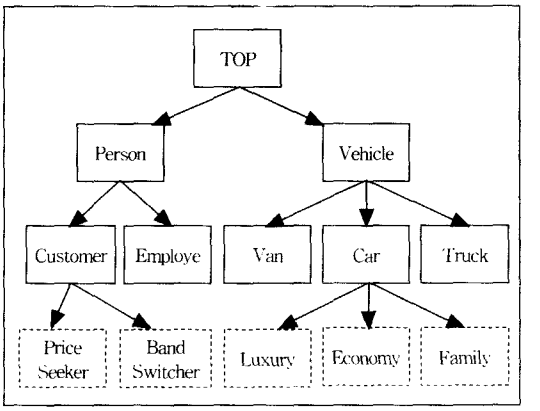

Figure 2. Schema of the Example Database

3.2 Schema Definition

In designing an ExOM database, we first introduce terms we are going to use to describe categories or objects. Attribute names and their feasible values(types) are asserted as follows:

```
ATTR customer_num: RANGE[0001..9999]
ATTR name: STRING
ATTR age: RANGE[10..100]
ATTR monthly_income: RANGE[0..20000]

high_income :=[monthly_income: GTE 4000]
moderate_income :=[monthly_income: RANGE[2000..3999]
status_car :=[owns: AT_LEAST(1) {BMW, Bentz, Rolls}]
stable_job :=[job: AT_LEAST(1) tenured_professor]
```

The type of an attribute is defined as a set of values allowed for that attribute. Also, a term

like high_income can be defined by a *feature*, which is an attribute/set-value pair(s).

In the ExOM, categories are defined, based on a category theory[30], either intensionally by necessary and sufficient feature conditions or extensionally by known typical instances: the former we call *structure-based* and the latter *exemplar-based*. For those structurally different concepts as well as definitional or mathematical concepts, it is relatively easy for the designer to define the operational criteria for examining the category membership of individual objects. These definable intensional category definitions will be rather stable over time. For example, the category customer may be definable as those objects among persons that have attributes such as customer_num, salary, assets, family_ size, owns, and job, which are necessary and sufficient conditions for category membership.

```
STRUCTURED CATEGORY customer
IN person
SATISFIES[customer_num, name, salary:monthly_income,
          assets: INT, family_size: RANGE[1..20],
          own: AtLeast(1) car, job: profession]
```

For other categories, the properties and membership conditions may not be easy to define. For example, the category prestige_customer is not easy to define, but often we need to classify customers into this category to better market types of cars they would prefer. In this case, the category is defined as a collection of objects, which are *similar* to a specific object known to belong to the category. For example, customer John is known to belong to the category prestige_customer from the past experience because he has bought luxury cars in the past. We call this model-like object *exemplar*. The general description of an exemplar- based category is induced by the system ex post based on the objects classified, but may not be true of all instances because of exceptional cases. Therefore, the definition for exemplar-based categories is coarse at the initial stage, which will be learned and refined over time as objects are classified or exemplars are added for more effective classification criteria. Let's suppose that John is already in our database.

```
EXEMPLAR CATEGORY prestige_customer
        IN customer
        SIMILARTO John
```

The category prestige_customer is defined as a subcategory of customer, that is, those objects similar to the exemplar John among customers.

3.3 Object Description

ExOM objects are described at three different levels-category, attribute, and value. That is, users first describe the category to which an object belongs, if it is known. Then, specific features are added by recognizing its attributes, the values of which are further explored if they are not completely known.

All objects in the ExOM belong to at least one category, called the TOP category. Therefore, if an object is not given any category specification, it is simply interpreted to belong to the most general category, TOP. The description of an object usually begins with the category to which the object is already known to belong and then extra features are specified. In this way, features play the role of constraints. For example,

```
OBJECT John IN customer
  SATISFIES[customer_num:= 123, name:= John,
            age:=32, monthly_income:=5000]
```

describes an object John, who is known to be a customer with features such as name "John", 32 years old and income $5000 per month. Because the category customer is defined as a structure-based category, John is also assumed to have those attributes such as assets, family_size, own, and job, but with a "null" value. In ExOM, the null value of an attribute is the most general value, that is, the type of the attribute. Therefore, the above assertion is equivalent to the following:

```
OBJECT John IN customer
  SATISFIES[customer_num:=123, name:=John,
            age:=32, monthly_income:=5000,
            assets: INT, family_size: RANGE[1..20],
            own: AT_LEAST(1) car, job: profession]
```

Another level of specification in describing objects is value specification. Adding features to object description, we first perceive whether certain attributes exist or not, and then specify the values of those attributes. Sometimes, however, we cannot specify the exact values of those perceived attributes. In this case, the values are roughly specified by constraining feasible values for attributes. The ExOM provides three kinds of constrained value expressions-range, cardinality, and category object. Range expressions constrain an unknown numerical value to an interval. For example, the previous assertion of John may have looked initially like the following, saying that John's age is between 30 and 40 and his monthly income is greater than $4000.

```
OBJECT John IN customer
  SATISFIES[customer_num:=123, name:=John,
            age: RANGE[30 40],
            monthly_income: GT 4000]
```

Cardinality constraints are used to restrict the number of elements, or the number of elements satisfying some constraints in a set value. Let's suppose that the type of the attribute own is defined as a set of cars. Over time, we accumulate the information on John's car ownership:

```
TYPE own: Car

OBJECT John
  SATISFIES[own: EXACTLY(2), family_size:=4,
            preference:sporty_look]
OBJECT John
  SATISFIES[own: AT_LEAST(1)
            luxury_car<=[price: GT 35000]]
OBJECT John
  SATISFIES[own:={Lexus, Jaguar},
            job:=associate_prof, addr:="Austin, TX"]
```

We refine the value of own by constraining the feasible values as we obtain more information: John is initially known to own exactly two cars(the cardinality of own). Then at least one of them is known to be a luxury car with the price more than $35,000, where luxury_car is a subcategory of car, and finally the value of own is disclosed and asserted. Note that the object-category expression luxury_car<=[price: GT 35000] denotes a set of luxury_car whose price is greater than $35,000, which is consistent with the type of own because luxury_car is a subcategory of car.

3.4 Queries

In ExOM, queries are expressed by object-

category expressions, which denotes a set of objects satisfying cert_in conditions specified. That is, an arbitrary object-category expression can be viewed as a query requesting all the objects in the database which are members of the category virtually defined by this object-category expression. We call this category a *virtual* category. Therefore, a query processing is viewed as a classification process of domain objects into a virtual category, which is implicitly represented by a query. The usual syntax of an ExOM query is as follows:

```
RETRIEVE Attributes FROM Object-Categoryexpr
```

Queries can be divided into two kinds: *content-based* queries that retrieve the set of objects satisfying a user's qualification and *classification-based* queries that retrieve the category most closely matching a user's feature set. Classification queries are inferred from the stored knowledge by using either deductive or inductive reasoning.

3.4.1 Content-based Queries

Queries about existing categories and objects without involving classification process are called *content-based* queries. Simple queries are shown below:

```
RETRIEVE name FROM customer
        SATISFYING[age: RANGE[30..39],
                   monthly_income: GT 4000]
```

In the following queries, the type of the attribute own is the category car. The first query retrieves the names of customers, who own at least two cars either explicitly or implicitly. (e.g., [own:= {Lexus, Jaguar}],[own: AT_LEAST (3)]). The second one asks for the names of customers who own at least one car with a price greater than $50,000.

```
RETRIEVE name FROM customer
        SATISFYING[own: AT_LEAST(2)]

RETRIEVE name FROM customer
        SATISFYING[own: AT_LEAST(1) car
            SATISFYING[price: GT 50000]]
```

Object-category expressions allow us to represent a subset of objects, which satisfy specified feature conditions among the domain objects represented by categories. By extending this notion to the "nested" sub-object-category expressions, the ExOM allows us to ask for the objects that a sub-object-category expression denotes, without introducing inverse attributes. For example, the query requesting the price of luxury cars which are owned by prestige customers and are more than $50000 is formulated by using path expressions without introducing an inverse attribute owned_by in the car category:

```
RETRIEVE own.price
        FROM prestige_customer
        SATISFYING[own: luxury_car
                SATISFYING[price: GT 50000]]
```

3.4.2 Classification-based Queries

The ExOM allows the expression of queries based on inductive and deductive reasoning at the level of individuals, category concepts, and category membership. Retrieval is done by matching queried features either to the definitional features of structure-based categories or to the features of existing exemplars of exemplar-based categories. A query asking for

the names of individual customers similar to John can be written as follows:

```
RETRIEVE name FROM customer SIMILARTO John
```

In the previous section, a query asking for the names of all customers who introduced at least one prestige customers whose income is greater than $50,000 is written by using nested object-category expression. In the same fashion, a query to retrieve the names of customers who introduced at least one prestigious customer similar to John is expressed as follows:

```
TYPE introduced: person

RETRIEVE name FROM customer
  SATISFYING[introduced: AT LEAST(1)
    prestige_customer SATISFYING[income: GT 50000]]

RETRIEVE name FROM customer
  SATISFYING[introduced: AT_LEAST(1)
              prestige_customer SIMILARto John]
```

Note that this query will first evoke the similarity-based classifier to find the subset of prestige customers who are similar to John, and then select customers whose value of introduced have at least one element from that subset, involving both deductive and inductive reasoning.

We can ask for the categories of a feature expression or an object(when its feature description is already given), followed by the keyword CATEGORY. Let's assume we have accumulated the following information on a customer Lee:

```
OBJECT Lee IN customer
  SATISFIES[customer_number:124, name:Lee,
            annual_income:GTE 80000]
OBJECT Lee
  SATISFIES[annual_income:RANGE[60000..70000]]

OBJECT Lee
  SATISFIES[own:{Porsche} job:dean, age:40]
```

After gathering all information, we can ask the result of the classification and see the justification of the system for the result.

```
RETRIEVE Lee CATEGORY
```

In fact, the system will classify all the objects into the most specific category when it is updated with more information. If there were no prespecified category for Lee, the system will start from the category Top. First, it classifies Lee into the category customer because it is preclassified. Then it further classifies Lee into more specific categories, that is, subcategories of customer. Because the subcategories of customer are exemplar-based categories, the membership is determined by the similarity of an object to the exemplar of the category. So, for example, the features of Lee are matched with the features of John, who is the exemplar of prestige_customer.

When the description of an object is incomplete and the classification is ambiguous, the system will generate the most feasible classification by using knowledge-based matching techniques. Similarly, when the query is ambiguous or has no exact match(partial match), similarity-based retrieval generates a more satisfiable answer to users than a simple "nil" message. For example, "X is a C because it is like Y which is known to be a C." is a reasonable argument when the available infor-

mation is not enough to draw the definite conclusion that X is a C.

Ⅳ. The Extensional Object Model

The Extensional Object Model(ExOM) consists of objects, categories, classifiers, which map each object to categories, and a domain knowledge-base.

Definition:
An *ExOM* is defined as a tuple $\langle O, C, K, \Lambda \rangle$ where O is a set of domain objects, C is a set of categories in the domain of discourse, K is a domain knowledge-base, and Λ is a *classifier* from O to C.

4.1 Object Description

Objects are described in terms of feature values consisting of attributes and value expressions. The feature values are collected over time and accumulated via feature combination operators. In this section, we describe the basic building blocks, attributes and features, and then how features are accumulated over time.

4.1.1 Attribute

Attribute names are terms that are used to describe the properties of domain objects or category concepts. In the ExOM, the structure of an object is not predefined. Instead, the set of values allowed for each attribute, called the *type* of an attribute, is defined. Besides base types - *STRING*, *INT*, *REAL*, *BOOL*, and *OBJECT* - the language provides a structure type, called feature type.

A new attribute name with its type is declared as follows:

TYPE Age: INT
TYPE Color: {red, yellow, orange, blue, brown, white, black}
TYPE Preference:[color:Color, shape:{sports, sedan, convertible}]

The above statement says that the attribute age can have a value of integer. In the ExOM, this type definition of an attribute declares two things; one is the name introduced is a term known to the system and the other is the value(s) associated with this attribute should be from the type declared. In fact, the type of an attribute is the coarsest description for the property denoted by that attribute because it says nothing but all possible values the attribute can have.

4.1.2 Feature

A feature is an association of attribute names to *feasible* values. It is very similar to a *record*, which is an association of names(labels) to values. However, in case of features, every attribute is semantically associated with a set, the elements of which denote feasible values for that attribute. Formally, a feature is defined as follows:

Definition:
A *feature* f is defined as a multi-valued partial function from A into V: $\boldsymbol{A} \rightarrow 2^{V}$.

So a feature is defined as a point-to-set mapping from A to V except for feature type attributes. A feature described by attributes(a_1, ..., an) and feasible value sets $\{v_{i1}, ..., v_{im}\}$ for

each a_i is syntactically written as follows:[..., a_i : $\{v_{i1}, ..., v_{im}\}$, ...]

A *well-formed* feasible value, υ, of an attribute with the type is defined as follows:

$$\upsilon = \{v_1, ..., v_n\} \text{ where } \forall v_i \in \tau$$

When a feasible value is a singleton set, we may omit the angle parenthesis. Hereafter, we assume that every feasible value is well- formed.

The feasible values associated with an attribute name a in a feature f is $f(a)$; the notation $f.a$ will also be used. When a feature f is defined on the set of attributes $Att \in A$,

$f = \lambda x$. **if** $x \in Att$ **then** $f.x$ **else**(**if** $x \in A$ **then** $\perp_{unknown}$ **else** $\perp_{exception}$)

where $\perp_{unknown}$ means that no information on the property denoted by attribute name x is provided yet, while $\perp_{exception}$ means that the attribute name x is not yet introduced to the system.

We assume that the feasible values always include the actual true value of an attribute. Let υ_{true} be the true value of an attribute a. Then, the following relationship is held between and υ_{true} by definition:

υ = nil $\Rightarrow \upsilon_{true}$ = nil
$\upsilon = \{\upsilon\} \Rightarrow \upsilon_{true} = \upsilon$
$\upsilon = \{\upsilon_1, ..., \upsilon_n\} \rightarrow \upsilon_{true} \subseteq \{\upsilon_1, ..., \upsilon_n\}$

When $f(a)$ is nil, it means that a has no feasible value, which is interpreted as the attribute a is "not" applicable to the object concerned. When υ has a single value, the actual value of a is inferred to be that value because it is the only value which is feasible to a. Therefore, the semantics of a feasible value is interpreted as follows:

$[a: \{\upsilon_1, ..., \upsilon_n\}] \Rightarrow \upsilon_{true} = (\upsilon_1$ **or** ..., **or** $\upsilon_n)$

For instance, a feature [own: {car1, car2, car3}] is interpreted as υ_{true} = car1 **or** car2 **or** car3 and all the possible actual values can be enumerated as follows:

υ_{true} = car1
υ_{true} = car2
υ_{true} = car3
υ_{true} = car1 and car2
υ_{true} = car1 and car3
υ_{true} = car2 and car3
υ_{true} = car1 and car2 and car3

When the information on an attribute a is complete, we call the attribute a is *closed*, which is syntactically denoted by ":=". That is,

$[a := \{\upsilon_1, ..., \upsilon_n\}] \Rightarrow \upsilon_{true} = (\upsilon_1$ **and** ..., **and** $\upsilon_n)$

The domain of a feature is the set of attribute names it defines:

$$Dom(f) = \{a \mid a \in A \text{ and } f(a) \neq \perp\}$$

The binary relationship $\leq_f$ between f and f' ($f, f' \in F$), which compares the *informatedness* between two features, is defined as follows:

Definition:

$f \leq_f f' \Leftrightarrow Dom(f') \subseteq Dom(f)$
and $\forall a \in Dom(f')$
$f(a) = f'(a)$ **if** (a closed in f' or $f'(a) = nil$)
$f(a) \subseteq f'(a)$ **if** a is a Base type
$f(a) \leq_f f'(a)$ **if** a is a Feature type

In the above definition, f is more informated than f' because it is defined on more attributes and also shows narrow-downed feasible values for all common attributes.

4.1.3 Feature Refinement Operators

It is useful to formalize the notion of incremental refinement, that is, conjoining and generalization, to compute the accumulation of information about an object over time. The feature conjoint operator, denoted by $\otimes$, computes the common part of separately gathered information about an object. Let's suppose that f is an original feature and g is a refinement feature.

Definition:

$$f \otimes g = \lambda x \,.\, \begin{cases} f(x) & \text{if } x \in \mathrm{Dom}(f) - \mathrm{Dom}(g) \\ g(x) & \text{if } x \in \mathrm{Dom}(g) - \mathrm{Dom}(f) \\ f(x) \cap_{fv} g(x) & \text{if } x \in \mathrm{Dom}(g) \cap \mathrm{Dom}(f) \end{cases}$$

If $f(x) \cap g(x) = \varnothing$ this contradicts the initial assumption that a feasible value will always include the true value. This implies that information gathering efforts will not totally fail to get some useful information. $\otimes$ is reflexive and symmetric, but not transitive.

Two features can be combined to generalize the concepts represented by them. Generalization operator, denoted by $\oplus$, is defined as follows:

Definition:

$$f \oplus g = \lambda x \,.\, \begin{cases} f(x) \cup_{fv} g(x) & \text{if } x \in \mathrm{Dom}(g) \cap \mathrm{Dom}(f) \\ \perp & \text{otherwise} \end{cases}$$

4.1.4 Objects

Let O be the set of all domain objects of interest.

Definition:

An object $obj \in O$, at a given time t, is defined as a tuple $obj^t ::= (o_{id}, f^t)$

where o_{id} is an element of *OBJECT* and $f^t \in F$ is the feature description of the object at a time t.

So the state of an object is time-variant, while its identity is time-invariant. The existence of objects in the real world cannot be captured solely by the values of expressions because of the inherent incompleteness(approximation) of our object representation. At some points, different objects may be described as having the same values. Therefore, the object-identity represents the existence of objects in the real world, while f^t represents the current information on the object concerned.

For each modeling object *obj*, we assume there exists a "true" partial function F_{obj} defined on the attributes $A = \{a_1, ..., a_n\}$ describing all the information on *obj*. Therefore, in F_{obj}, all the(feasible) values of attributes denote actual values. F_{obj} gives the complete knowledge of *obj*. However, we have only a partial information on F_{obj} at a given time, that is, f_{obj}, giving only approximate description of *obj*. Therefore,

$$\forall\, f_{obj}{}^t \; (F_{obj} \leq_f f_{obj}{}^t)$$

holds for all $obj \in O$. At a given time t, F_{obj} is approximated by $f_{obj}{}^t = [a_i: v_i, ..., a_j: v_j]$. Over time, *obj* will be described in more detail either by exploring unknown attributes or by narrowing down the feasible, but not yet fully specified values of recognized attributes. So, the information gathering process in the ExOM corresponds to a transition from $f_{obj}{}^t$ to $f_{obj}{}^{t+1}$ and eventually to F_{obj}:

$$f_{obj}{}^{t+1} ::= f_{obj}{}^t \otimes \Delta f_{obj}$$

where Δf_{obj} denotes additional information on *obj* between time t and $t+1$.

4.1.5 Object Expression

In the ExOM, objects are described in terms of object expressions. These expressions are evaluated by the system to update object memory.

Definition: An *object expression* is syntactically defined as

$$Obj_{expr} ::= (Ctg) \Leftarrow f_{expr}$$

where f_{expr} =[$attribute_{name}$: $value_{expr}$, ...].

Intuitively, an object expression is a statement about an object, which asserts that it is a member of specified categories and then specifies properties by describing the observed attribute value pairs(that is, the feature value).

4.2 Category Definition

The basis of any ExOM schema is the representation of various category structures and the relations between categories in the domain. Objects in the memory are organized into categories according to a schema, the process of which we call *classification*.

We distinguish two dimensions of classification: a *vertical* dimension and a *horizontal* dimension[27]. The vertical dimension is based on the inclusion relationships among categories where inclusion is determined by structural differences. This dimension reflects the level of abstraction; the larger the inclusiveness of a category, the higher the level of abstraction. In the ExOM, *is_a* relationship between categories, which is also in the core of object-oriented data models, represents this dimension of classification. The horizontal dimension, which is the focus of the ExOM, concerns the segmentation of categories at the same level of abstraction. For example, the segmentation of a category CAR into three categories, Luxury Car, Economy Car, and Family Car may not involve any significant structural addition in terms of feature description, but differentiates concepts in terms of feature interpretation mostly with no clear cut boundaries.

The ExOM has two kinds of categories: structure-based and exemplar-based. The concept of a structure-based category is represented by its features, which are necessary and sufficient conditions for category membership, and that of exemplar-based categories is represented extensionally by *exemplars*, which play the role of the representatives for similar objects. Let's denote structure-based categories by C_s and exemplar-based by C_e.

Definition: A set of *categories*, C, is defined as a disjoint union of two subsets, C_s and C_e,

$$C_s ::= (C_{id}, f_{Def})$$
$$C_e ::= (C_{id}, Exemplar_{id})$$

where $f_{Def} \in F$ is the(abstract) feature definition of a structure-based category.

4.2.1 Structure-Based Category

The ExOM uses the same syntax of object expression to define a structure based category, that is,

STRUCTURE CATEGORY Ctg_{new} :=$(Ctg) \Leftarrow f_{expr}$

The semantics of structure-based category definition is given as follows:

$Eval[Ctg \Leftarrow f_{expr}]=\{obj \mid obj \in Ctg \wedge obj.f \leq_f f_{expr}\}$

4.2.2 Exemplar-Based Category

To define an exemplar-based category, an exemplar should be asserted.

EXEMPLAR CATEGORY $Ctg_{new} := Ctg \Rightarrow obj_{id}$

Where $\Rightarrow$ means *similar_to*. It says that Ctg_{new} denotes a subset of objects in Ctg, which are similar to obj_{id}. The semantics of exemplar-based category definition is given as follows:

$$Eval[Ctg \Rightarrow obj_{id}] = \{obj \mid obj \in Ctg \wedge obj.f \ similar\_to \ obj_{id}.f)\}$$

Exemplars are a subset of objects in the domain, each of which is considered prototypical of a certain category.

4.3 Knowledge-Base

In the ExOM, the domain knowledge is represented by a set of Horn clauses. A *Horn clause* is an implication in which a conjunction of zero or more conditions implies a conclusion. The conclusion of a clause is separated from its conditions by "←" and the conditions are separated from each other by a comma. For example, the following clause:

$$A \leftarrow B_1, \ldots, B_n$$

says that A is concluded if every B_i (i = 1 ... n) proves to be true. The B_i's are the conditions of the clause, and A the conclusion. A clause may have no condition.

The conclusion and conditions of a clause can be any ExOM relation.

Definition: An ExOM relation is defined as a n-ary relation with a qualifier:

$$R^n ::= P[q](term_1, \ldots, term_n)$$

where P is a relation name, *term* is either an object ID, a feature, or a variable and q means a qualifier indicating the inherent strength of belief on each relation.

Qualifiers are usually used to determine the extent of *knowledge-based matching*. If there is no qualifier specified in a relation, then the relation is interpreted to be certain by default. The examples of relations are as follows:

```
imply[often]([own: Jaguar],[preference:
        sporty_look]).
equivalent(X, Y) <- has_function(X,_),
        has_function(Y,_).
```

4.4 Classifier

Classification is to identify a partially-known object as a member of known categories in the domain of discourse. We mean by "known" that a category is intended by data interpreters for some purposes. This is the difference between the ExOM classification and the general inductive classification where the goal is to find a set of classification rules with a given positive or negative test examples. Therefore, in ExOM, only relevant features for classification are provided when a category is defined.

According to the representation scheme of categories, the ExOM calls the appropriate classifier to classify an object into categories. In our ExOM, Λ has two components, a subsumption-based classifier (λ_s) for C_s, and an exemplar-based classifier (λ_e) for C_e because a classifier is dependent on the representation of categories. ExOM calls the appropriate classifier according to the representation of categories, and classifies given objects into categories.

4.4.1 Structure-Based Classification

An object *obj* is classified into a structure-based category Ctg_s if the feature of *obj* is "less" than the feature definition of Ctg_s. That is,

$$obj.f \leq_f Ctg.f_{Def} \textit{ implies } obj \in Ctg$$

A category Ctg_i *subsumes* a category Ctg_j if and only if every object in Ctg_j is also a member of Ctg_i where $Ctg_i, \in Ctg_j$ C_s. Therefore, ExOM may classify some partially-known objects into categories correctly when all the feasible values of partially known attributes are subsumed by the category feature definition. Because the category membership of an object is determined by the less than relationship between its feature and the category's feature definition, the subsumption relationship between two categories can be computed on the basis of whether the feature definition for Ctg_j logically implies the feature definition for Ctg_i, that is,

$$Ctg_j.f_{Def} \leq_f Ctg_i.\ f_{Def}$$

4.4.2 Exemplar-Based Classification

The exemplar-based classifier λ_e is a system for inductive classification with learning capability. λ_e is based on the exemplar-based categorization theory[27, 30] and implemented by using AI's case-based reasoning technique. The domain expert's model for the classification is implicitly stored in the memory through exemplars, reasoned by the system through a knowledge-based similarity matching process. This λ_e is learned and refined over time through the explanations for the failed classification.

When an exemplar-based category is created, because it is defined within the scope of a structure-based category, only those objects in that scope are examined for their membership. An object *obj* is classified into an exemplar-based category Ctg_e if *obj* is determined to be similar to the exemplar of Ctg_e in terms of its features. Figure 3 shows the algorithm for the ExOM exemplar-based classification. The threshold value for the similarity δ is assumed to be given.

For a given object *obj*
Select the most prototypical exemplar e in Ctg_e
  e := e
  Compute the similarity measure $S := S(e, obj)$
  For all other exemplars e_i in Ctg_e
    Compute Expected Gains $Exp(e, e_i)$ in similarity
    and Select e_i such that $\forall i\ Exp(e_i) \geq Exp(e_i)$
  If $Exp(e_i) \geq 0$ Then Compute $S := S(e_i, obj)$; e := e_i
If $S \geq \delta$ Then Classify *obj* into Ctg_e with the similarity S to the exemplar e

Figure 3. The ExOM Exemplar-based Classification

Following Tversky[33]'s featural approach to similarity and adapting its contrast model to the exemplar-based category theory, we define the simple ExOM similarity function, which measures the similarity of object *obj* to exemplar *e*, as follows:

$$S(e, obj) ::= (\sum_{ftr \subset fd} \theta_{ftr} + \sum_{ftr \subset fi} (\theta_{ftr} * \mu) - \sum_{ftr \subset fu} \theta_{ftr}) / \sum_{ftr \subset fe} \theta_{ftr}$$

where *fe* is the relevant features of exemplar *e*,
fd is the relevant features directly matched by *obj*,
fi is the relevant features indirectly matched by *obj*,
fu is unmatched relevant features,
θ_{ftr} denotes the relative importance of each relevant feature, and

μ denotes a certainty factor for indirect matching.

The exemplar e is a *model* of the category. So, it becomes the target object for a new object *obj* to be matched with in terms of features to be classified into the category. Not all of exemplar's features are, however, relevant to category classification. In fact, the system must have the knowledge to distinguish the relevant features among many given features. So, only exemplar's relevant features are considered important for classification and there is no effect of (unmatched) features distinct to an object *obj* on similarity measure.

In addition, the degree of relevance, called the *importance*, of each relevant feature may be varied in classification. The importance of exemplar's relevant features indicates the *diagnosticity*[31] of each feature in determining category membership. In the ExOM this knowledge is provided when an object is asserted as an exemplar of a category. Each unit feature of an exemplar has an importance value between 0 and 1, and if it is greater than 0, *ftr* is considered as a *relevant* feature and if equal to 0, as an irrelevant feature, respectively.

Note that the number of relevant features of an exemplar does not have an effect on the similarity measure because we normalize the total matching by dividing it by the total importance of exemplar's(relevant) features. Therefore, from the above similarity function,

$$-1 \leq S(e, o) \leq 1$$

where 1 is a perfect match and -1 a perfect mismatch. If the threshold value(τ) of $S(e,o)$ is 0.5, this means three fourth of exemplar's features should be matched to be classified into the category, if the importance of all relevant features is assumed to be same. μ is a measure of the certainty in matching, having the value between 0 for no match and 1 for a direct match. For each unit feature of a target exemplar, μ is computed through matching process.

Because a unit feature is a pair of attribute name and feasible value expression, rather than one term, we can think of three possible cases for indirect matching: semantic matching, partial matching, and knowledge-based matching.

(1) Semantic Matching

Semantic matching can be done when two feasible value expressions for a same attribute can be evaluated into the matchable values. That is, let[att: ν_{rf}_expr] be a relevant feature to be matched, and[att: ν_{of}_expr] be an object's feature. Semantic matching is to evaluate these feasible value expressions before trying to check the satisfiability, where the evaluation function of ExOM expressions takes a feasible value expression and the current environment (ExOM memory) as arguments and gives an ExOM value(refer to Figure 4).

For example, let's suppose that the category car has five members such as Bentz, Jaguar, Cadillac, Accord, and Escort, ν_{rf} = car <= [price:GTE 25000], and ν_{of} = car <= [price:GTE 20000, shape:sports]. Then ν_{rf} is evaluated into {Bentz, Jaguar, Cadillac} and ν_{of} is {Jaguar}. Because feasible values in category definition are interpreted as allowable values for category membership, the ExOM feature matching is a matter of constraint satisfaction. Because ν_{of} is evaluated to semantically satisfy ν_{rf} in a given current environment, they are considered as matched together.

```
Let v_i := Eval(v_i_expr, Env) and v_j := Eval(v_j_expr, Env) in Satisfiability(v_i , v_j )
where Eval(v_expr, Env) ::=
        case v_expr of
                Boolean b . return(v_expr)
                Integer i . return(v_expr)
                String s . return(v_expr)
                Object_id id . return(v_expr)
                Feasible_ Value ({..., expr, ...}) . return({..., Eval(expr, Env), ... })
                Feature [a1: e1, ..., an: en] .
                        vi := Eval(ei, Env) for i = 1, ..., n
                        return([a1: v1, ..., an: vn}])
                Field (expr.ai) .
                        v := Eval(expr, Env)
                        If v = Feature [..., ai: vi, ...] Then return(vi)
                        Else return(ERROR)
                Object_Category_Expression (Ctg ⇐ f_expr).
                        Let fg := Eval(f_expr, Env)
                        return({obj | obj ∈ Ctg ∧ obj.f ≤i fg})
                Object_Category_Expression (Ctg ⇒ f_expr) .
                        Let fg := Eval(f_expr, Env)
                        return({obj | obj ∈ Ctg ∧ Similarity(obj.f, fg) ≥ δ)
```

Figure 4. Semantic Matching and Evaluation Function

(2) Partial matching

When two feasible value expressions for the same attribute can be evaluated into only partially matched values, we need to determine the degree of satisfiability between two features. For example, let's suppose that the category car has the same five members as the above example, ν_{rf} = car <=[price:GTE 25000], and ν_{of} = car <=[made_in:America]. Then ν_{rf} is evaluated into {Bentz, Jaguar, Cadillac} and ν_{of} is {Cadillac, Escort}. As a result, ν_{rf} is partially matched with ν_{of} in a given current environment. The constraint feature may be "possibly" satisfied or may be not. This possible satisfaction reflects the fact that the value of the object's attribute does not represent exact values, but feasible values at the time of matching. We call the degree of partial matching *satisfiability* viewing the features of the exemplar as constraints.

The measure of satisfiability depends on the user's strategy for the classification of an object. One extreme case is to consider a partial matching as fully satisfiable, resulting in loosening feature constraints, while the other is as unsatisfiable because of a possible mismatch, resulting in tightening constraints. Between two extremes, we may set the satisfiability between 0 and 1.

In Figure 5, users may define a function to determine the satisfiability of partial matching between two feasible values. For example, Morrissey[23] proposed two methods for estimating uncertainty that can be used to rank imprecise objects to be retrieved in response to a query[1]: self-information and entropy.

1) These methods assume that all values are equally likely.

```
Given: Two feasible values of the same type, v_rf and v_of
Output: satisfiability of v_of for v_rf
    Satisfiability(μ_sf , v_rf) ::=
        If v_of ∩ v_rf = ∅
            Then μ_sf := 0
        Else (If v_of ⊆ v_rf Then μ_sf := 1
            Else μ_sf := user defined function)
```

Figure 5. Satisfiability for Partial Matching

The self-information approach computes the amount of extra information required by the system to prove that *obj* satisfy a given query clause q, by taking the difference between the information available and the minimum information required to be certain that the attribute value satisfies q. Similarly, the entropy approach computes the uncertainty for *obj* to satisfy a query q, by taking the difference between the entropy associated with the minimum information needed and that associated with the information available. The entropy is defined as the average of self-information:

$$Entropy = -\sum_v P(v) \log P(v)$$

where $P(v)$ is the probability of a particular attribute value v occurring. Even though both methods are lack of rigorous large scale testing, they seem to provide a good example of user-defined functions based on information theories.

(3) Knowledge-based matching

The ExOM incorporates domain knowledge into the similarity matching process using *knowledge-based matching*[26] in Figure 6. It matches dissimilar features by finding a path of

```
Input: f_e: relevant features of a target exemplar, f_o: object features
Output: μ_ftr for each feature in f_e

For each a ∈ (Dom(f_e) ∩ Dom(f_o))
        Let vo := Eval(f_o,a), ve := Eval(f_e,a)
        If vo ∩ ve = ∅
                Then μ_ftr_a := 0
        Else
                If vo ⊆ ve Then μ_ftr_a := 1
                Else μ_ftr_a := satisfiability(vo, ve)
Let f_ou be unmatched features of object o
For each a ∈ Dom(f_e) − Dom(f_o)
        Starting from the feature of highest importance ftr_a
        Let ve := Eval(f_e,a)
                Let SearchCost := SearchCost(θ_ftr_a)
                Evoke Knowledge-based matching for [a: ve]
                If find a path [a1: v1] [a2: v2], ..., [a: vn] using rules in KB
                  such that a1 ∈ Dom(f_ou), Eval(f_o,a1) ∩ v1 ≠ ∅, vn ⊆ ve
                Then
                        Let ls_i be the link strength of the ith link.
                                ls_i := ls(q_i, R_i)
                        Let link be the strength of the whole link
                                link := link(l_1, ..., l_n-1)
                        Let vo := Eval(f_o,a1)
                        If vo ⊆ v1
                                Then μ_ftr_a := link
                        Else Compute satisfiability of ( vo, v1)
                                μ_ftr_a := μ (link, satisfiability)
                Else μ_ftr_a := 0
```

Figure 6. Knowledge-based Similarity Matching

relations connecting them, which shows how the features are equivalent for the classification. That is, the implicit shared properties between an exemplar and an object are determined from different explicit feature representations.

It starts from the most important feature among unmatched features. The depth of search is dependent on the importance of the feature concerned. It is a backward-chaining search beginning from the target(exemplar's feature) until it finds a path linking to a source(object's feature) within a given search cost. The link strength of the path ftr_a, ..., ftr_i ..., ftr_k is determined jointly by the qualifiers on each relation and the nature of relation names. If that path is completely certain, the match is considered equivalent to a direct match, that is μ is set to 1.

For example, the new object may have a feature[lease:{Bentz}] with the following equivalent relation in the ExOM knowledge-base:

equivalent_to[sometimes]
([lease: {Cadillac}],[own: {Cadillac}])

Then the relevant feature[own:{Bentz, Jaguar, Cadillac}] can be matched with[lease:{Cadillac}] through the link such that*[own:{Bentz, Jaguar, Cadillac}] is satisfied by[own:{Cadillac}], which is sometimes equivalent_to[lease:{Cadillac}]*, but with some uncertainty represented by the qualifier sometimes and the relation name equivalent_to. That is, the overall link strength in knowledge- based matching is heuristically determined by both the certainty of each relation and the inherent linking strength of each relation name in the path. For example, the qualifier "always" is more certain than "sometimes" and the relation name "equivalent_to" is stronger than "imply" in terms of linking.

In summary, ExOM tries to match each relevant feature of an exemplar with a corresponding object feature by using semantic, partial, and knowledge-based matching

V. Related Works

This paper has presented a framework for intelligent data interpretation systems. An extensional approach to object-oriented data modeling is proposed as a base model of IDIS in weak-theory domains, under which structure-based and exemplar-based concept representation schemes are viewed as mutually supporting for domain modeling. Exemplars can compensate for difficult-to-generalize feature definitions, while structure-based category definitions can improve the efficiency of classification.

The proposed framework for IDIS has three modules category definition for domain modeling, object description for information acquisition, and reasoning for classification, which provide a framework for discussing related works. First, we discuss different views of representing category concepts in cognitive science. Then we discuss object-oriented data models and non-traditional data models. Finally case-based reasoning is reviewed shortly.

5.1 Cognitive Science

There are three major views of how people represent category concepts: Classical, Probabilistic, and Exemplar views[1, 30]. The classical view holds that a concept is defined as a

collection of features which are singly necessary and jointly sufficient for classifying an object as an instance of the concept. In the probabilistic view, the features entering into the summary description of a concept need to be only probabilistically related to concept membership. So one matches features of test item and target concept until one accumulates a threshold amount of probabilistic evidence. The exemplar view argues that a concept is represented not by a summary representation, but by separate descriptions of some of its exemplars. So, a concept is defined extensionally and past typical instances of a concept are retained for use in classification.

The concept representation in the ExOM is partly based on the exemplar view, in which a concept is represented not by a summary representation, but by separate descriptions of some of its exemplars[27, 30, 33]. The term exemplar can refer to either an abstraction of a category concept or to an individual instance of a concept. These exemplar representations in ExOM are initially acquired from expert DB designers in the form of an individual instance, but are revised by the system to correct mis-classified objects.

Rosch conceived of category systems as having both a vertical and horizontal dimension[27]. The vertical dimension concerns the level of inclusiveness of the category and the horizontal dimension concerns the segmentation of categories at the same level of abstraction. The ExOM pays more attention to the categorization of continuous object space, which does not have clear-cut boundaries, and for which necessary and sufficient criteria for membership does not exist.

5.2 Object-Oriented Languages

The object-oriented data models[15] are based on the classical view of a concept representation, in which a concept is defined as a collection of features which are singly necessary and jointly sufficient for classifying an object as an instance of the concept[30]. Classes are organized according to the *is_a* relationships between classes(the vertical dimension), in which the horizontal dimension of category systems is mostly ignored.

In classless languages[34], prototypical objects are used to implement shared information or behavior between objects through *delegation* instead of inheritance[19]. Both LaLonde[18] and Sciore[28] showed that the exemplar-based approach can provide a more flexible and powerful notion of inheritance by separating class hierarchies and instance hierarchies. The difference is that in the ExOM, exemplars are mainly used as models to recognize and classify objects into categories.

5.3 Non-traditional Data Models

In VAGUE[24], a vague query establishes a target qualification and is concerned with data that are *close* to this target. The adequacy of the geometric approach of the data metrics to similarity is, however, questioned on both theoretical and empirical grounds[27, 33].

In CANDIDE[2], data objects, query objects, and view objects are treated in a uniform way and classification functions are used to process database queries. CLASSIC[4] is a structural data model for objects, which permits partial and incremental descriptions of individuals under an

open world assumption. A concept is described by the conjunction of more general concepts with restriction-based constructors. Both of them, however, assume that the underlying data model reflects a complete model of the domain and all classification is done through the subsumption-based operations. In weak theory domains, the data model should provide a mechanism to reflect an incomplete model of the domain.

AGENDA[14] is called an item/category database to handle free-textual data and allows for organization to evolve as the database grows. The user classifies an item explicitly or the system classifies deductively using rules. The semantics of an item is totally determined by the classification.

In hypertext, classes(stacks) are semi-structured and the structure of objects(nodes) are allowed to be flexible. However, the need for "virtual" structures for dealing with changing information and the practical difficulties of classification in the real world have been recognized. Also, the incorporation of query-based access and intelligent knowledge into the system to derive new information from existing information are issues to be handled in these systems.

5.4 Case-based Reasoning

Case-based reasoning infers the solution(or classification) of a new case from that of similar cases previously presented to the system. It has been applied to a wide range of fields as law, medical diagnosis, question-answering, recipe planning and dispute resolution[5, 11, 16, 26]. Especially, the work of Porter[26] has had a great impact on the ExOM's knowledge-based matching.

Ⅵ. Conclusion

This paper proposes a computer-supported loosely coupled data interpretation system viewing organizations as interpretation systems. To implement IDIS computationally, we combine an extended object-based data model, called Extensional Object Model, and heterogeneous reasonings-subsumption-based and case-based-as complementary processes for classification and learning in a unified framework. Our next step is to provide the ExOM with a module to discover any regularity in its memory such as dependencies between attributes, features, or categories, and to generalize into classification rules. IDIS can be applied to decision support systems or office information systems for organizational computing with learning. Especially, semi-structured database systems such as document management systems and multi-d atabase systems may be a good application domain for the ExOM.

The contributions of this paper are summarized as follows:

- Proposal of the architecture of a computer supported data *interpretation* system to handle information overload, to discover knowledge from database, and to enhance organizational learning.
- Loose coupling between objects and categories to achieve the flexibility in describing objects and defining categories to support incomplete information and incremental knowledge acquisition;

- Integration of intensional and extensional concept representation;
- Provision of both inductive and deductive reasoning mechanism in the database environment for both classification and query processing through knowledge base;

Some of the possible directions in which a IDIS based on the ExOM can be extended are outlined below.

- How to efficiently handle the computational complexity resulted from the loose coupling of objects and categories,
- How to assure consensual validation for organizational interpretation, incorporating the coordination process into the model as a schema integration process,
- How to explore incremental knowledge discovery in our ExOM framework.

REFERENCES

[1] Bareiss, R. and Porter, B., "A Survey of Psychological Models of Concept Representation," AI TR87-50, University of Texas, 1987.

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

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

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

[5] Branting, Karl, *Integrating Rules and Precedents for Classification and Explanation: Automating Legal Analysis*, Ph.D. Dissertation, The University of Texas at Austin, 1990.

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

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

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

[9] Galbraith, J., *Designing Complex Organization*, Addison-Wesley, Reading, Massachusetts, 1973.

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

[11] Hammond, K. J., *Case-based Planning: Viewing Planning as a Memory Task*, Perspectives in Artificial Intelligence, Vol. 1, Academic Press, Inc., 1989.

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

[13] 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.

[14] Kaplan, S.J., et al., "AGENDA: A Personal Information Manager," *Communications of the ACM*, Vol.33, No.7, July 1990.

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

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

[17] Levitt and March, "Organizational Learning," *Annual Review of Sociology*, 1988.

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

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

[20] March, James G., "Ambiguity and Accounting: The Elusive Link between Information and Decision Making," *Account. Organization Society*, 1987.

[21] Mannila, H., "Methods and Problems in Data Mining," *Proc. Intl. Conf. on Database Theory*, Springer-Verlag, 1997.

[22] Matheus, Chan, and Piatetsky-Shapiro, "Systems for Knowledge Discovery in Databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol.5, Dec., 1993.

[23] Morrissey, J.M., "Imprecise Information and Uncertainty in Information Systems," *ACM Transactions on Information Systems*, Vol. 8, No.2, April 1990.

[24] Motro, A., "VAGUE: A User Interface to Relational Databases that Permits Vague Queries," *ACM Transactions on Office Information Systems*, Vol.6, No.3, July 1988.

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

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

[27] Rosch, E., "Principles of Categorization," *Cognition and Categorization*, Rosch and Lloyd(ed.), Hillsdale, N.J.: Erlbaum, 1978.

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

[29] 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), pp.301-316.

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

[31] Smith, E., "Categorization," *An Invitation to Cognitive Science: Thinking*, Osherson and Smith(ed.) Vol.3, The MIT Press, 1990.

[32] 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.

[33] Tversky, A., "Features of Similarity," *Psychological Review*, Vol.84, No.4, July 1977.

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

[35] Weick, Karl E., *The Social Psychology of Organizing*, Addison-Wesley, Reading, Massachusetts, 1979.

[36] Yoon and Kerschberg, "A Framework for Knowledge Discovery and Evolution in Databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol.5, Dec., 1993.