Anwar, T., Beck, H., & Navathe, S. (1991). Conceptual Clustering in Database Systems. 2nd ASIS SIG/CR Classification Research Workshop, 11-32.
doi:10.7152/acro.v2i1.12544

Conceptual Clustering in Database Systems[1]

Tarek M. Anwar
Howard W. Beck
Database Research and Development Center
Computer and Information Sciences
Gainesville, Florida 32611
INTERNET: ta0@beach.cis.ufl.edu

Shamkant B. Navathe
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332

Classes are an integral part of all semantic data models. Despite this, class formation in these data models is ad hoc due to the varied treatment of classes and because the issue of grouping instances into classes is considered an art rather than a science. It is the view of this paper that class formation be based on category theory through the use of an *attribute-based purpose-directed conceptual clustering technique*. Several issues concerned with category theory, especially exception handling, are discussed. The emphasis in this approach is on reasoning at the instance level. Schema generation occurs as a result of conceptually clustering the underlying data instances and guiding this process by specifying a context in the form of a clustering seed. The use of this approach in the areas of schema integration, schema evolution and querying will be discussed. These facilities have been implemented on a database system based on the CANDIDE [3] semantic data model. CANDIDE is essentially an extended version of the term-subsumption languages known as the KL-ONE family of languages.

1. INTRODUCTION

A construct that is common to all semantic data models is that which represents a generic concept. This construct is referred to by many names such as a class [19], an entity type [12], a category [11], or a frame [8]. In this paper we will refer to this construct as a class, unless otherwise specified. A class models a generic concept as an abstraction of a set of instances of that generic concept. These instances are members, the extension, of the class. A class may also contain an internal description, the intension. This description identifies other concepts associated with the class through different types of relationships (such as generalization, specialization, aggregation, association) and conditions (semantic integrity constraints) that dictate this type of association. Through certain types of relationships (generalization and specialization) some data models allow classes to inherit properties from one or more other classes. In some data models class membership is determined by necessary and sufficient conditions as exemplified by the Entity-Category-Relationship data model [12] where predicate-defined classes called categories can be specified and in term-subsumption languages such as KL-ONE [8] or CANDIDE [3] where attribute restrictions may be used. If an instance satisfies these conditions it is automatically added to the class. In other words, the generic concept modeled by that class is represented as a predicate defined over instances. However, in most data models class membership is determined manually

1. This work was supported in part by the Florida High Technology and Industry Council and by a grant from Georgia Tech Research Corporation, Atlanta, Georgia

without the use of necessary and sufficient conditions. Essentially, a schema (the definition of the structure of the database) of a semantic data model consists of a set of data model constructs assembled together using some modified version of a semantic network [12,19].

Currently, the process of designing a schema is very difficult due to the unpredictability and evolution of the real world and the error-prone process of interviewing end-users to determine relevant classes, attributes and relationships [5]. Moreover, the expressive power of semantic data models gives designers greater leeway in modeling an application domain without any guidance as to the appropriateness of the constructs used. This is termed relativism [17,7]. Consequently, the quality of the classes generated, and hence the schema are dependent on the designer's intuition and experience. In addition to this, it is extremely difficult, if not impossible, to define 'natural kinds' using necessary and sufficient conditions as utilized in some data models. How can you define a 'chair' using necessary and sufficient conditions? Does a bean-bag chair satisfy these conditions? Because of these reasons a class description is usually too general and does not specify all the pertinent information of the corresponding generic concept. This in turn will force the user to tailor the data to the schema resulting in the possible loss of information [5].

In this paper we propose a novel approach to schema design by the use of an attribute-based purpose-directed conceptual clustering technique. This technique creates conceptually cohesive [23] classes that are based on category theory rather than ad hoc classes. Schema generation occurs as a result of conceptually clustering the underlying data instances of different databases and guiding this process by specifying a clustering seed. The clustering seed guides the conceptual clustering process by specifying the context and inductive (generalization) preference criteria. The context specifies criteria by which a set of relevant instances are extracted from the database. The inductive preference criteria guide the way these instances are clustered into classes. By modifying the clustering seed we are able to vary the schema generated to accommodate different user needs. In this technique we recognize the importance of data instances in class formation and reason at the instance level to create classes that more accurately and precisely reflect the data stored. By allowing instances to come from any application domain or mini-world including existing databases our methodology can be treated as a schema design or schema integration methodology. For the generic use of this methodology in different contexts we shall use the term view generation to refer to the activity of defining a schema over single or multiple databases.

In a nutshell, the goal of the present paper is to correct the shortcomings and limitations conventional schema design methodologies mentioned earlier as follows:

- Generating conceptually cohesive classes based on category theory rather than ad hoc class formations.
- Recognizing the importance of data instances in class formation and incorporating them in the process of generating more accurate class descriptions.
- Capturing essential semantics that must be considered for defining schemas to avoid ad hoc class formations due to simple measures of similarity.
- Generating the primary constructs of semantic data models. Our methodology is not geared toward one particular data model but can be easily tailored to any data model.
- Providing increased automation by reducing user interaction.

ISSN: 2324-9773

The remainder of the paper is organized as follows. Section 2 presents an overview of category theory relevant to our purpose. A survey of several prominent conceptual clustering techniques is given in section 3. Section 4 discusses several schema design applications in database systems to which we are applying conceptual clustering. Conclusions are presented in section 5.

2. A THEORY OF CATEGORIZATION

It has long been argued in [26] that human categorization is not an ad hoc process but rather a process that is characterized by certain principles of categorization. Categories are groups of objects which are considered similar in a certain context [26]. To categorize an instance means to consider it, for purposes of that categorization, not only similar to other instances in the same category but different from instances not in that category. Category boundaries try to reflect the natural discontinuities that occur in perceptual and functional features of our world. The cohesion between members of a category does not lie in the overall degree of feature overlap, but in the overlapping of relevant, context-dependent features [18,27,26,28]. Context affects both the level of abstraction at which an instance is considered and the relevance of its features. Consider batteries and reservoirs. They do not have very much in common, but in the context of 'energy storing devices', they are very similar. They are even considered analogous, as in 'batteries are reservoirs' [15]. By the same token, humans around the world share many common attributes; but, in the context of 'religious beliefs', are divided into many categories. Simply put, categories are viewed as complex conceptual clusters rather than simple predicates. Instances of categories exhibit three types of similarities: 1. structural (correspondence between object components), 2. semantic (correspondence in meaning) and 3. pragmatic (relevance to the context specified in the clustering seed) similarity [18].

2.1 Principles of Category Formation

Two general principles guide the formation of categories. The first, cognitive economy, deals with the functional aspect of a category system to provide maximum information with least cognitive effort. The second principle deals with the structure of the perceived world and asserts that objects of the world are perceived to possess high correlational structure rather than arbitrary or unpredictable features. That is, the world is not an unstructured total set of equiprobable co-occurring attributes. Thus, maximum information with least cognitive effort is achieved if categories map the perceived world structure as truthfully as possible. To reiterate, category boundaries try to reflect the natural discontinuities that occur in the perceptual and functional features of our world.

2.2 The Category System

A category system is a taxonomy by which categories are related together through class inclusion. Such a system has a vertical and horizontal dimension. The vertical dimension reflects the level of abstraction or 'inclusiveness' of a category, similar to that of a generalization hierarchy. As a category becomes more abstract, and therefore more inclusive (contains more instances), there would be very little that all the instances have in common. This is termed the family resemblance effect [26]. The em basic level of abstraction is the level which maximizes the similarity between category members in a certain context [26,13]. Consider the following category system: electronic equipment, computer and personal computer. When referring to a Macintosh, an inventory clerk may identify it as electronic equipment while an end-user will probably relate to its computing capabilities and identify it as a personal computer. The horizontal dimension is concerned with the segmentation of categories at the same level of abstraction. Most, if not all, categories do not have clear-cut boundaries. To achieve separateness and clarity, categories tend to become defined in terms of prototypes or prototypical instances that contain the features most representative of the instances inside the category and least representative of instances outside the category. Most data models ignore prototypes and treat all instances as equal members of the class. The notion of prototypes is closely related to that of default values which are typical attribute-value pairs of instances in a class.

2.3 Exception Handling: To Err Is Human

Exception handling has emerged as a particularly difficult problem, and one for which we have not yet developed a satisfactory solution. This is due in part to the extremely rigorous way we treat exceptions. Previous treatments of exceptions have, in our view, failed to appreciate the difficultly of the problem and defined exceptions in a relatively trivial fashion. In fact, exceptions play a central role in the dynamics and evolution of categories. The category system must modify itself to accommodate exceptions, but first it must be able to recognize exceptions. While category modification (which data modelers call schema evolution, see section 4.2) is itself not trivial, we argue that exception recognition is much more difficult.

Intuitively, exceptions occur whenever the classification system makes an error. For example, a disease diagnosis program fails to classify a disease correctly, a computer vision system mistakenly identifies a telephone pole as a tree, a bird mistakes a broom handle for a snake, a policeman mistakes a toy gun for a real weapon, or a language speaker misinterprets the use of a particular utterance. First notice that such examples are widely available. Such errors are not necessarily good or bad. To err is human, and computers ought to fail just as gracefully. Also, the error may never be detected, it may have no consequence. The correct disease may never be known, it may not matter to the robot whether something is a telephone pole or tree, and the bird may flee from the broom and live another day just the same. So what makes these misclassifications erroneous?

More formally, an instance is an exception if it belongs in a particular category, yet fails to meet any intensional description specifying conditions for membership in that category. Conversely, an instance is an exception if it does satisfy conditions for membership in a particular category, yet it does not belong in that category. In either case, the exception is miscategorized. The first kind of exception is an error of excluding an instance from a category to which it belongs, the second kind is an error of including an instance in a category to which it does not belong. Failure to diagnose a case of pneumonia as pneumonia is an example of the first kind of exception. Mistaking a broom handle for a snake is an example of the second kind of exception.

The problem with previous treatments of exception handling is that they presuppose that the categories to which an instance belongs is already known. This begs the question in categorization problems. Once the category is known, it is then a simple matter to test whether the intensional description is or is not satisfied. Borgida [5] uses this approach in a system which automatically learns constraints for database classes. Exceptions are given to the system, and constraints are modified to accommodate exceptions. This addresses the schema evolution problem, but does not address the problem of identifying exceptions since the classes to which the exceptions belong are told to the system. More recently, Bergadano et. al [6] have developed a two-tiered concept representation which includes a Base Concept Representation (BCR) which is much like a prototype or typical properties associated with a category, and an Inferential Concept Interpretation (ICI) which provide rules and conditions for deviations from the BCR. Some exceptions can be identified by deductive reasoning from the ICI. While this uses a more sophisticated inferencing approach to identify exceptions, it could not in general identify all exceptions, specifically those which cannot be deduced from the ICI. Also, the ICI is not learned but specified by an outside trainer. In general, we have not found any approaches which consider the more difficult problem of exception handling. Making an error in an initial categorization is natural and expected, but then the error must be detected and the category structure corrected to accommodate the exception. And all this must happen without the system being explicitly told about the error. Although learning by being told is a valid form of learning, it cannot be used as the sole means of exception handling.

To make matters worse, workers in the field of information semantics such as Dretske [9], Fodor [14], , and Godfrey-Smith [16], have identified a problem, which they call the disjunction problem, related to the exception handling problem. Briefly, this arises in the causal theory of meaning which states that symbols (i.e. names of categories) get their meaning because of a direct, causal physical connection to the things in the real world which they represent. For example, the concept 'cow' is causally connected to real Cows because of light hitting a Cow, bouncing off, hitting the retina, exciting neural activity, and so on, ultimately resulting in a tokening of the "cow" concept. The disjunction problem has to do with explaining, using this causal theory, how misrepresentation can occur. For example, a Horse may under certain circumstances be mistaken for a Cow, that is, Horse may cause a tokening of 'cow'. But if symbols get their meaning by causal chains, then apparently 'cow' doesn't just mean Cow, it means the disjunction, Cow or Horse, since Horse causes tokening of 'cow' just as Cows do. Thus, 'cow' is a misrepresentation of Cows, and we have a problem similar to the exception handling problem mentioned earlier. An early proposed solution included a form of learning by being told [9], in which a teacher (which may be a human or nature) provides training on what things are examples of the concept and what are not. A proposed teleological solution suggests that the perceptual apparatus must be functioning normally, under optimal conditions, for a symbol to have correct meaning, but that is a rather vague notion in which 'optimal' is difficult to specify. But generally, there has not yet been a plausible solution proposed to the disjunction problem. The connection between the disjunction problem and handling exceptions is that a causal mechanism for handling exceptions should solve the disjunction problem.

In our own work we have developed a notion of similarity based on conceptual measures. Central to our similarity definition is a function which we call INTERSECT. INTERSECT takes two instances as arguments and produces a new class. The new class tells what if anything the structure of the two instances have in common by doing a graph intersection on the semantic network representations of the two concepts. Thus, the INTERSECT of London and Paris is a new class, which might be something like "Capital Cities of European Countries." We have attempted to develop an exception condition using INTERSECT. The exception conditions suggests that a new instance, *I*, might be an exception to an existing class, *C*. It works by computing INTERSECT between the new instance, *I*, and every instance in the database (although this sounds computationally complex, the candidate set of instances which actually need to be checked is only a small percentage of the complete database). If in the process INTERSECT generates non-trivial results between *I* and instances of *C*, yet *I* fails to satisfy the intensional description for *CI*, and in addition whatever I has in common with other instances of *C* it does not share in common with non-members of *C* (I is uniquely similar to other members of *C*), then perhaps I is an exception to *C*. Unfortunately, "perhaps" is the strongest that this can be stated. There is no absolute test, according to this approach, which would determine strongly that *I* is an exception, for perhaps *I* is a member of *C*, and perhaps it is not (the disjunction problem).

It appears that deductive reasoning cannot be the answer to exception handling. Exceptions cannot be identified on the basis of past and present information consisting only of empirical observations and rules for determining category membership. Given only that information, then the categorization process will produce a particular classification. If that classification is erroneous, the errors cannot be detected using the same information used to generate the classification.

So a promising direction is to look at the influence of future information on the dynamics of category structure. Simply put, a particular categorization is used to make predictions about the future, and these predictions can be tested when additional information is obtained. If the predictions turn out to be false for particular instances, then these instances are identified as exceptions and the category structure can then be modified to correct the error. This is nothing less then the scientific method. But unfortunately, again this is not a trivial solution to the problem. For how is the interpretation of the new information to be trusted? Where in the chain of reasoning did the real error occur [10]. Furthermore, it leads to the need to maintain multiple hypotheses, and that can produce a combinatorial explosion even in small domains.

In this section we have attempted to show that exception handling is not a trivial problem. Especially, a rigorous notion of what an exception is needs to be accepted. A good solution to exception handling is currently lacking. This problem stands in the way of a good conceptual clustering algorithm. In spite of this, several promising conceptual clustering techniques have been developed. These will now be described.

3. CONCEPTUAL CLUSTERING: AN OVERVIEW

Conceptual clustering is a technique based on category theory that creates classes of objects that are conceptually cohesive [27]. In contrast, classes in conventional data analysis, based on numerical taxonomy and regression analysis [23], are formulated solely on the basis of a numerical measure of similarity between the objects. The similarity between objects is determined as the value of a mathematical function applied to each object's description. If this value was within a certain threshold, the object would be considered a member of that class. This type of similarity measure is context-free; it is not able to capture the properties of a class that are not derivable from the properties of the individual objects [23]. For instance, by using this type of similarity measure we could categorize objects, with the property "location," as being similar if they were in the vicinity of each other. But, it is doubtful our notion of vicinity will be the same in context of 'countries in the vicinity' and 'people in the vicinity'.

For any given set of facts there will be a potentially large number of hypotheses that could be generated that imply these facts. Background knowledge is used to constrain this process. Components of background knowledge include information about object properties, assumptions about the inductive assertion, a variety of inference rules (deductive and inductive), and specialized procedures that allow the system to generate logical consequences of given assertions and new descriptions [27]. The context determines not only the relevant attributes but also the level of abstraction . For example, doctors have many attributes and can be classified in many different ways. But, when trying to find a competent doctor, hair color or height are usually not relevant to your choice even though they are valid classifications. Relevancy of attributes cannot usually be directly determined; therefore, deductive rules and specialized routines are required. Deductive rules capture the interrelationships between attributes and therefore determine attribute relevancy. Specialized routines are required to transform the domain of the attribute from one representational space to another to facilitate comparisons and other operations. Michalski's view of conceptual clustering is as follows [22, 23, 27]. Given 1) a set of object descriptions (in our case these are attribute-value vectors), 2) a tentative inductive assertion, which may be null (the clustering seed), and 3) background knowledge to guide the clustering process, generate a class taxonomy that implies the object descriptions and satisfies the background knowledge. In a nutshell, {\em conceptual clustering can be viewed as a heuristic search through the space of possible symbolic descriptions generated by the application of various inference rules to object attributes and constrained by background knowledge [23, 27].

CLUSTER/2 is a conceptual clustering technique developed by Stepp [23]. It utilizes Michalski's INDUCE/2 algorithm to generate class descriptions from a set of instances. Although INDUCE/2 was intended to be used as a form of learning from example, it was applied in such a way that CLUSTER/2 would operate without a tutor providing examples of class members. CLUSTER/2 utilizes a set of rules incorporated in the Lexical Evaluation Function (LEF) to determine category quality. Classes that do not meet the quality control are rejected. CLUSTER/S [27], based on the work in CLUSTER/2, utilizes a goal-dependency network (GDN) to derive attributes related to the initial goal. For example, if the goal is "Survive," then the concept "Eat Food" is determined to be relevant through the GDN. Classes are formed that correspond to relevant attributes identified by the GDN.

COBWEB [13] is a conceptual clustering technique that utilizes a probabilistic approach. A statistical measure, the *class utility measure* to evaluate the quality of different classes (clustering patterns) over the same set of instances. This class utility measure is designed to maximize the ability to predict instance attributes from knowledge of class membership. By using such a measure COBWEB would have difficulty creating classes that are relevant to a specified goal because pragmatic similarity is not considered.

ACME [18] is a program that utilizes a connectionist approach to conceptual clustering. In this program constraints of a class are represented be means of a network of supporting and competing hypotheses regarding which elements of the instance to map. A cooperative algorithm for parallel constraint satisfaction identifies mapping hypotheses that collectively represent the overall mapping that best fits the interacting constraints. The network incorporated by ACME utilizes

ISSN: 2324-9773

numerical semantic weights, thus a class modeled by this network is determined by a numerical measure of similarity.

4. SCHEMA DESIGN APPLICATIONS IN DATABASE SYSTEMS

4.1 Schema Integration

Schema integration is the activity of integrating schemas of proposed or existing databases into a global, unified schema. The basic problems to be dealt with during schema integration stem from structural and semantical diversities of the schemas to be merged [2, 25]. Any conventional schema integration methodology can be considered as a mixture of these following activities:

1. an analysis of the schemas to establish the integration policy,
2. comparison of the schemas to determine possible correspondences,
3. conforming the schemas to facilitate merging,
4. merging and restructuring of the schemas.

The main drawback of current schema integration methodologies is that they are *data model dependent* and integrate at the *construct level.* All of the four steps outlined in Batini, Lenzirini and Navathe [2], entail some kind of analysis of schemas. Moreover, these schemas must be "normalized" by translation into a common model prior to integration. These methodologies reason at the class level to the exclusion of instances. Furthermore, the integration process is marred by extensive designer interaction [2, 7]. Among other causes, this could be attributed to the ad hoc and informal treatment of classes in semantic data models. Without formal treatment of classes the designer must use his intuition and experience to interpret the meaning of each class. Moreover, current integration tools depend on simple measures of similarity such as string and domain comparisons [2] which are incapable of capturing essential interrelationships between database objects vital to the integration process. As an example consider the attributes "birth-date"and "age." These attributes would not be considered similar by these measures of similarity even though one can be derived from the other (age = date - birth_date).

The application of conceptual clustering to schema integration creates conceptually cohesive classes that are based on category theory. Schema integration occurs as a result of conceptually clustering the underlying data instances of different databases and guiding this process by specifying a clustering seed. By reasoning at the instance level we are able to generate classes, and schema, that more accurately and precisely reflect the actual data stored. Fig. 1 illustrates the architecture of the conceptual clustering system (CCS) [1]. CCS consists of four main modules. These modules are the databases (to which it is connected), the background knowledge base, clustering seed, and the view generator. The following subsections will describe these four components in greater detail.

Anwar, T., Beck, H., & Navathe, S. (1991). Conceptual Clustering in Database Systems. 2nd ASIS SIG/CR Classification Research Workshop, 11-32.
doi:10.7152/acro.v2i1.12544

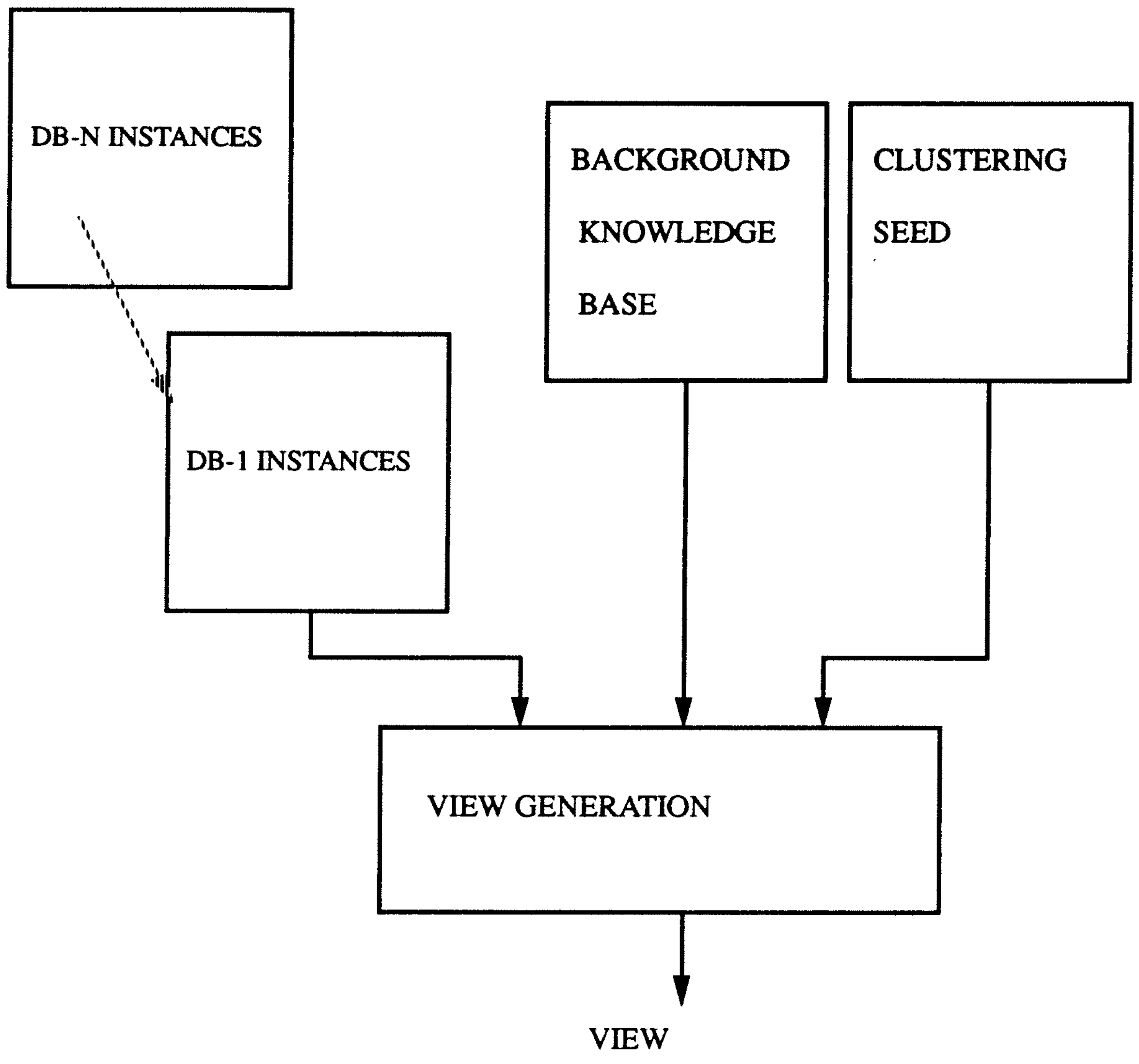

Fig. 1: Conceptual Clustering System Architecture

4.1.1 Databases

Several databases are connected to the CCS. Databases are accessed retrieval-only leaving updates to the discretion of the local database management system. Only actual instances are considered. These instances are assumed to be in attribute-value pairs of possibly variable sizes. We assume here that the data format of the instance descriptions are uniform. If not, several reformatting techniques could be specified that will generate a uniform representation of these instances.

ISSN: 2324-9773

4.1.2 Background Knowledge Base. The background knowledge of the system is stored in this module. This knowledge base is a set of facts that models specific interrelationships between attributes. These interrelationships allow the system to deduce relevant attributes with respect to the clustering seed specified by the user. The following is a list of the different types of interrelationships modeled by the system:

1. **Set-subset** relationship, denoted by the fact *subset* (attr_j, attr_k) specifies that the domain of attr_j is a subset of that of attr_k. That is,

   $$dom\ (attr_j) \subset dom\ (attr_k)$$

   For example, *subset*(*chairman*, *advises*) represents the fact that all those who chair a committee must also advise. In other words, the domain of 'chairman' is a subset of that 'advises'.

2. **Synonymity**, denoted by the fact *synonym*($attr_j$, $attr_k$) specifies that $attr_j$ is a synonym of $attr_k$. That is, it is considered equivalent. For example, *synonym*(*ss-num*, *ssn*) represents the fact that the attribute 'ss-num' is a synonym of, and therefore considered equivalent to, 'ssn'.

3.**Logical** implication, denoted by the fact *logical*($attr_j$, $attr_k$), specifies that $attr_j$ is logical consequence of $attr_k$. For example, *logical*(*name*, *ssn*), specifies that all who have a social security number must also have a name.

All the above constructs exhibit the transitivity property. That is,

$$subset(attr_1,attr_2) \wedge subset(attr_2,attr_3) \Rightarrow subset(attr_1,attr_3)$$
$$synonym(attr_1,attr_2) \wedge synonym(attr_2,attr_3) \Rightarrow synonym(attr_1,attr_3)$$
$$logical(attr_1,attr_2) \wedge logical(attr_2,attr_3) \Rightarrow logical(attr_1,attr_3)$$

While the *subset* and *logical* constructs are antisymmetric, the *synonym* construct is symmetric. That is,

$$synonym(attr_1, attr_2) \Leftrightarrow synonym\ (attr_2, attr_1).$$

Through the utilization of these properties the system is capable of inferring facts not explicitly specified.

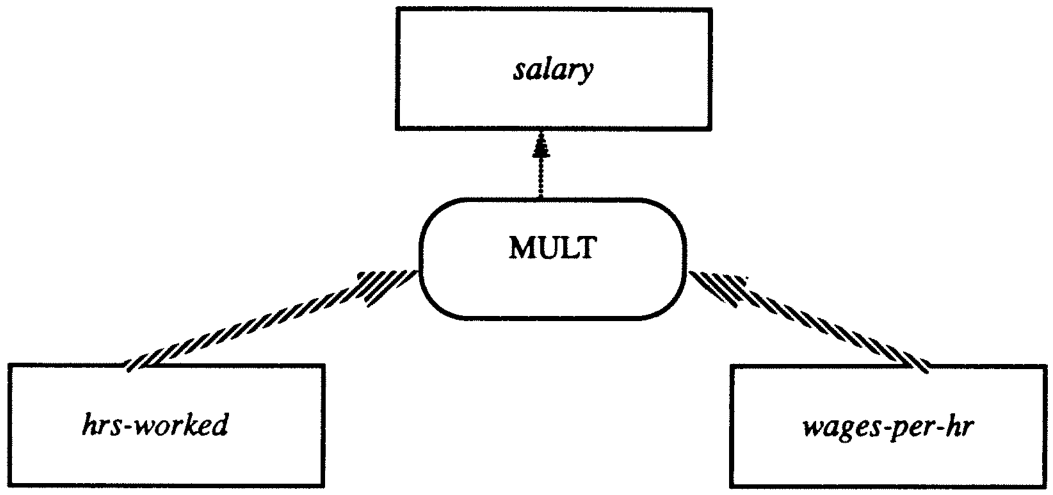

Fig.2. Dataflow Graph of *salary = hrs-worked* x *wages-per-hr.*

To model mathematical relationships between the attributes, a functional (acyclic) dataflow graph is used. These mathematical relationships permit the system to transform the domain of an attribute from one representational space to another. A dataflow graph that represents the following mathematical relationship

math(*salary* = *hrs-worked* x *wages-per-hr*)

is shown in Fig. 2 where attribute nodes are represented by boxes, actor nodes by octagons and arcs by dashed arrows.

4.1.3 Clustering Seed. The user is able to specify the type of view desired by defining a clustering seed. This clustering seed guides the conceptual clustering process by specifying the relevant attributes of the domain of discourse and the inductive (generalization) preference criteria. The clustering seed has three clauses as shown below.

```
SELECT teaches OR salary
FROM Database-1, Database-2
BY                UNION(teaches), CLOSE-INT(salary)
```

This seed specifies the relevancy of the attributes 'teaches' or 'salary', that only instances from Database-1 and Database-2 will be considered, and to generalize over the domain of 'teaches' by obtaining the union of all its values and to obtain a range of all the values of 'salary'.

The SELECT clause of the clustering seed is an expression of the relevant attributes of the user's domain of discourse. Several operators can be used in this expression to restrict the domain of attributes. These operators are {>, <, ≤, ≥, ≠, =}. Boolean operators AND, OR and NOT are used

ISSN: 2324-9773

to construct more complex expressions. The FROM clause is a list of databases from which the view will be generated. Only the instances of these databases will be considered. The BY clause specifies the type of generalizations to be applied to each attribute for the generation of class descriptions and attribute restrictions. The clustering algorithm will abide by these types of generalizations. The types of generalization applied to an attribute depends on its domain. If the domain of the attribute is of type integer or real we can obtain the union of all the values (UNION), the range of all values by 'closing the interval' (CLOSE-INT), or perform statistical analysis of the values such as the mean (AVG), maximum (MAX) and minimum (MIN). By not specifying a type of generalization, the attribute restriction is generalized to the whole domain.

4.1.4 . View Generator. The view generator module will generate a view of the data in the chosen data model (CANDIDE in the proposed implementation) according to the clustering seed, background knowledge and the database instances. The steps entailed in this process are:

- Determination of the Relevant Instance (RI) Set and Relevant Attribute (RA) Set

From the SELECT clause of the clustering seed we are able to determine the attributes selected by the user. An *analogy function* is employed to determine the relevant instance set and the relevant attribute set. This function considers every instance in the databases (specified in the FROM clause) and adds that instance to the RI set if:

1. any of the attributes of the instance directly match the user specified attributes or its synonyms, provided the attribute value satisfies the expression in the SELECT clause. This is considered as a direct match.

2. any of the instance attributes can be derived from one of the user specified attributes by:
   - applying different transformations (logical/arithmetic) modeled in the background knowledge base by the logical and math constructs, respectively,
   - inferring a set-subset relationship between them through the subset construct, and its value satisfies the expression in the SELECT clause.

   This is considered an indirect match.

If an instance is determined relevant by one of the user specified attributes,that user specified attribute will be added to the relevant attribute (RA) set. At the end of this step we would have determined all the relevant instances and relevant attributes.

- Generation of the Class Taxonomy and Derived Attribute (DA) Set

At this stage we have determined the relevant instance (RI) set and the relevant attribute (RA) set. Now the initial class taxonomy can be constructed. This process consists of the following steps:

ISSN: 2324-9773

1. Determine the largest common subexpression of attributes among the instances of the RI set. This is a prototypical representation of the instances of the RI set. A class with these attributes is created. Add the attributes of the subexpression, if any, to the *derived attribute* (*DA*) *set*. Note, this subexpression might be null because as the number of instances grow they will have very little in common (family resemblance). All of the instances of the RI set are members of this class. This class is a subclass of THING.

2. For each class generated, C, determine the largest, non-overlapping subexpressions of attributes, not members of the DA set, among instances of that class. In choosing an attribute subexpression we try to maximize

   - the number of instances 'covered' by each subexpression and

   - the length of the subexpression.

   These subexpressions should cover all the instances of the class unless all the attributes of an instance are already members of the DA set. A candidate class is created for each subexpression. Instances with these subexpressions are members of the corresponding candidate classes. A candidate class is added to the class taxonomy if an identical class, with the same instances, does not already exist. The resultant classes generated are subclasses of the original class, C, and instances could participate in more than one class. All the attributes of the subexpressions are added to the DA set.

3. Repeat step 2 until all the attributes in the RA set are accounted for in the DA set. At the end of this process, the DA set and the class taxonomy will be completely specified.

4.2. Schema Evolution and Exception Handling. Once an exception has been identified (See section 2.3), the database schema must evolve to accommodate the exception. This means that intensional descriptions must be added or modified to include the exception. The taxonomy must be altered by creating new classes and restructuring superclass, subclass, and parent relationships to be consistent with the modified intensional descriptions. Schema evolution is a highly dynamic, incremental process which occurs whenever a new instance or class is added to the database.

An overview of the schema evolution process is as follows. Figure 3.a shows a new instance, I, which raises the exception condition. Instance I has just been added to the database, and its relationship to other instances in the database computed through INTERSECT (See section 2.3). This has resulted in the creation of one new class, C', which indicates that I has some uniquely similar features to other instances (I_2) of an existing class C. However, I does not satisfy the intensional description for C, and thus is not immediately recognized as an instance of C. The exception condition is raised to indicate that I might be an exception to C. If a decision is made to include I in C (this decision cannot currently be automated but must be guided externally), then the database schema must evolve to the state shown in Figure 3.b. To achieve the schema in Figure 3.b,

ISSN: 2324-9773

a new class *C″* is created which has an intensional description that covers both *I* and all the instances of *C*. *C″* necessarily subsumes *C* and *C″*.

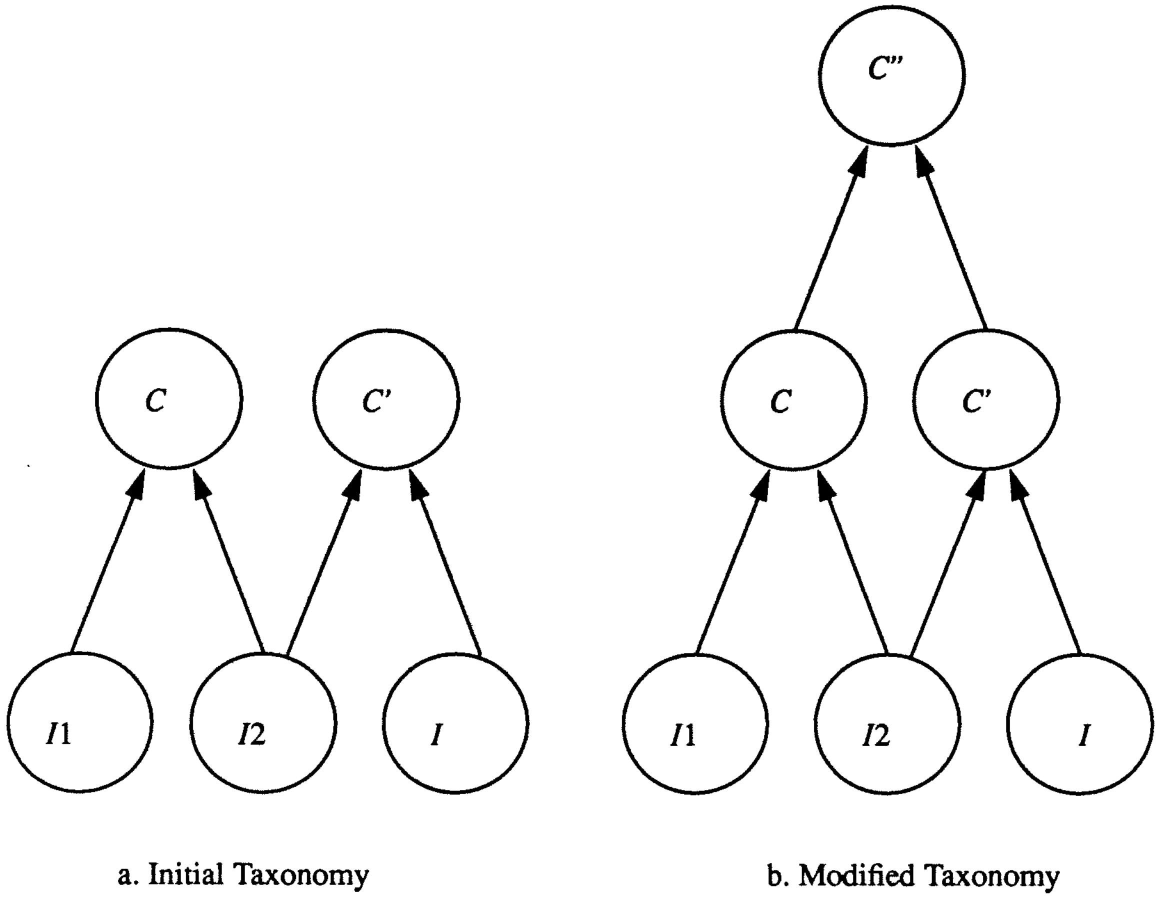

Fig. 3: The Schema Evolution Process

Notice that instance *I* acts as a clustering seed by identifying the context in which a new class is generated. Moreover, the concept associated with *C* has essentially shifted or evolved to *C″*. *C″*accommodates the exceptional instance *I*. Note also that the intensional description associated with *C″* is necessarily more general than that for *C*. This is a result of the family resemblance effect, in that the most abstract intensional descriptions become smaller and smaller as the class evolves to include more instances. Eventually the intensional description may disappear entirely because there is nothing that all the instances in the class have in common. However, there will be a rich collection of more specific subclasses. For example, the old class *C* is retained, and is a perfectly valid model of the subset of instances of *C″* which it is capable of describing. What happens in schema evolution as a category evolves to greater complexity is that an initial, small number of classes containing rich intensional descriptions are supplemented with additional classes at various levels of abstraction. The category as a whole covers more instances, yet the classes at top of the category, being most abstract, have very simple or non-existent intensional descriptions. However, the category as a whole consists of a large collection of specialized sub-classes, each modeling a portion of the instances in the class. The number and diversity of instances covered by the class increases.

4.3. Inexact Querying

Conventional queries manipulate databases to retrieve sets of instances that completely satisfy the conditions specified in a query. Recently, more attention has been given to inexact queries in which instances are retrieved that partially match a query. Such a facility improves user-system interaction by providing more informative answers to queries that do not match any instance in the database. Rather than returning a null answer to such a query the system retrieves instances that are somehow 'similar' to the conditions specified in the query. Moreover, users by necessity or intentionally could formulate their queries in such a way in which an inexact match is preferred. For example, it might be advantageous to find the earliest airline flight that most closely matches a request submitted be a traveler, even though no available flights would completely match this request. Without this capability the user will be forced to emulate such a request by repeatedly submitting that query with alternative values. If these alternative values are not known even this approach becomes infeasible. Thus, the need for such a facility is greater for naive users of a database.

A facility for inexact querying has been implemented which uses conceptual clustering techniques. In contrast to numeric [20, 24] or fuzzy set [29] approaches which ultimately rely on distance metrics and thresholds to process such queries, conceptual clustering retrieves instances which are structurally (correspondence between object components), semantically (correspondences in meaning), and pragmatically (relevancy to the goal, query object) similar to the query even though they may not match the query exactly [18]. The query processor has a deductive and inductive component. The deductive component finds exact matches in the traditional sense. In CANDIDE, a query object is specified in the same notation as other database class objects. Thus, the data definition language (DDL) and the data manipulation language (DML) are identical and provide uniform treatment for all database objects. Query processing occurs by classifying the query object to determine its correct place in the class taxonomy. Classification is performed automatically through subsumption. A class C_1 subsumes C_2 if C_2 satisfies the necessary and sufficient conditions specified in the class description of C_1. Essentially, the classifier finds the most specific classes that subsume the new class and the most general classes subsumed by the new class. A realization function is used to determine whether an instance meets an existing class description. Only those instances of the immediate superclasses are considered. Consequently, query processing is based on deductive inferencing about structured objects rather than procedural specification of operations. Inferencing techniques are defined formally by the subsumption function. Query specification is entirely declarative; the user need not specify *how* (in terms of algebraic operations) or from *where* (logical access paths and exact attribute names) the query is to be executed.

The inductive component identifies ways in which inexact matches may be considered similar. The inductive component generates new class descriptions from the original query. This is done by generalizing the constraints specified in the query in different *directions* and to different *levels*. The clustering algorithm thus automatically generates a class taxonomy and groups instances into classes. The imprecise query algorithm is triggered by a null answer in response to a query. That is, if no instances in the database satisfy the conditions stated in the query the system will try to find instances that partially satisfy the query.

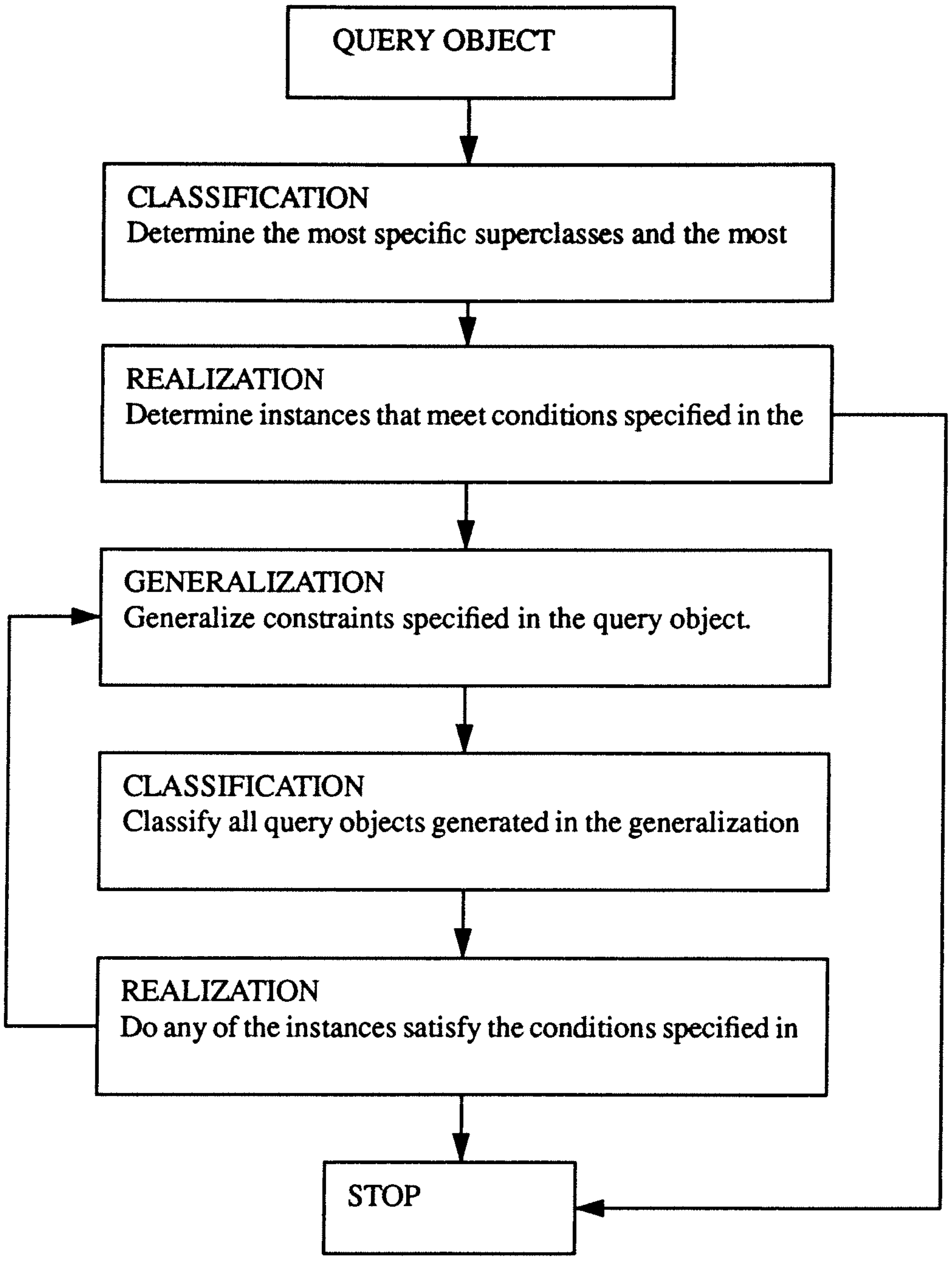
QUERY OBJECT
CLASSIFICATION
Determine the most specific superclasses and the most
REALIZATION
Determine instances that meet conditions specified in the
GENERALIZATION
Generalize constraints specified in the query object.
CLASSIFICATION
Classify all query objects generated in the generalization
REALIZATION
Do any of the instances satisfy the conditions specified in
STOP

Fig. 4: A schematic illustration of the imprecise querying algorithm.

The algorithm, as schematically described in Fig. 4, consists of the following steps:

1. The original query is generalized by relaxing the attribute restrictions stated in the query. This is done by selectively relaxing the value set constraints and the cardinality constraints stated in attribute restrictions. Several new query objects can be generated.

2. These generalized query objects are then classified in the taxonomy and the realization function is used to determine whether instances satisfy these query objects.

3. If no instances satisfy these generalized queries, the process is repeated for each generalized query until either all conditions are dropped (last step of generalization) or instances satisfy any of the queries.

Essentially, this technique follows up the failed query with several more general queries. This process is continued until a match is found.

The result of the conceptual clustering process is not only a set of instances but also a schema that describes those data instances. This is an *intensional answer* to a query [21]. The resultant schema can be regarded as an 'abstracted answer' to the query. Conceptually, this abstracted answer is much more informative, in the sense that it is easier to comprehend by the user, than to abstract an intension from a set of instances. A distinct feature of this approach is that the intensional answer is specified in the data definition language (in the form of a schema) as opposed to predicates in other techniques. Therefore, we are able to utilize the full expressive power of the semantic data model to describe these instances.

5. CONCLUSION AND FUTURE WORK

Data modeling and database management are heavily influenced by theories of categorization at various levels of sophistication. Ultimately designing and using databases ought to be viewed as a categorization process. In order for database classes to model accurately the structure of real categories, these processes ought to be based on the best available theories of category formation.

We have developed a working theory of category formation and applied it to the a number of database management problems including schema design, schema evolution, schema integration, and query processing. Current research effort is directed at the problem of exception handling, which is blocking greater automation of database functions.

We have implemented the CANDIDE data model, and implemented the object manipulation facilities including the classifier, and the INTERSECT function. Originally this was done under UNIX on a Sun workstation. The latest implementation is being done under MS-DOS using Microsoft Windows and C++. This is being done to provide a more practical delivery environment.

We are currently testing the database on some sample domains to evaluate the performance and quality of generated categories. A query processor has also been implemented, including inexact querying.

REFERENCES

[1] T. Anwar, S. Navathe, H. Beck. A Sementically Adaptive Modeling Interface for Schema Generation Over Multiple Databases. Technical Report TR-90-16, Database Systems Research and Development Center, University of Florida, Gainesville, FL, 1990.

[2] C. Batini, M. Lenzirini and S.B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. In *ACM Computing Surveys*, Vol. 18, No. 4, pp. 323-363, Dec. 1986.

[3] H.W. Beck, S. Gala and S.B. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *Proceedings of the IEEE Fifth International Conference on Data Engineering*, Los Angeles, CA, 1989.

[4] H.W. Beck, T.M. Anwar and S.B. Navathe. Towards Database Schema Generation by Conceptual Clustering. Technical Report TR-91-05, Database Systems Research and Development Center, University of Florida, Gainesville, FL, 1990

[5] A. Borgida, T. Mitchell and K.E. Williamson. Learning Improved Integrity Constraints and Schemas From Exceptions in Data and Knowledge Bases. *In Knowledge Base Management Systems*, Micheal L. Brodie and John Mylopoulos (eds.), Springer-Verlag, NY, 1986.

[6] F. Bergadano, S. Matwin, R.S. Michalski, and J. Zhang. Learning Two-tiered Descriptions of Flexible Concepts: The POSEIDON System, *Machine Learning Journal*, 5/1991.

[7] M. Bouzeghoub, G. Gardarin and E. Metais. Database Design Tools: An Expert System Approach. In *Proceedings of the 11th International Conference on Very Large Data Bases*, Stockholm, 1985.

[8] R.J. Brachman and J.G. Schmolze. An overview of KL-ONE knowledge representation system. In *Cognitive Science*, 9:171-216, 1985.

[9] F. Dretske. *Knowledge and the Flow of Information*. MIT Press, Cambridge, MA., 1981.

[10] K. P. Eiselt. Toward a unified theory of lexical error recovery. In *Proceedings of Cognitive Science Society*, Annual Meeting, 1991.

[11] R. Elmasri, J. Weeldreyer and A. Hevner. The Category Concept: An extension to the Entity-Relationship Model. In *International Journal on Data and Knowledge Engineering*, 1:1, May 1985.

[12] R. Elmasri, and S.B. Navathe. *FUNDAMENTALS OF DATABASE SYSTEMS*, Benjamin/ Cummings Publishing Company, 1989.

[13] D. Fisher. Knowledge acquisition via incremental conceptual clustering. In *Machine Learning*, 2:139-172, 1987.

[14] J. Fodor. *Psychosemantics*. Bradford Books, Cambridge MA., 1987.

[15] D. Gentner. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science*, 7:155-170, 1983.

[16] P. Godfrey-Smith. *Misinformation*. Canadian Journal of Philosophy, 19(4):533-550, 1988.

[17] M. Hammer and D. McLeod. Database Description with SDM: A Semantic Database Model. In *ACM Transactions on Database Systems*, Vol. 6, No. 3, Sept. 1981.

[18] K. Holyoak and P. Thagard. Analogical Mapping by Constraint Satisfaction. In *Cognitive Science*, 13:295-355, 1989.

[19] R. Hull and R. King. Semantic database modeling: Survey, applications and research issues. In *ACM Computing Surveys*, 19:3, Sept. 1987.

[20] T. Ichikawa and M. Hirakawa. ARES: a relational database with the capability of performing flexible interpretation of queries. In *IEE Transactions on Software Engineering*, SE-12(5):624-634, May 1986.

[21] T. Imielinski. Intelligent Query Answering In Rule Based Systems. In *The Journal of Logic Programming*, 1987:4:229-257.

[22] R.S. Michalski. A Theory and Methodology of Inductive Learning. In *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.), Tioga, Palo Alto, CA, 1983.

[23] R.S. Michalski and R.E. Stepp. Learning From Observation: Conceptual Clustering. In *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.), Tioga, Palo Alto, CA, 1983.

[24] A. Motro. VAGUE: a user interface to relational databases that permits vague queries. In *ACM Transactions on Office Information Systems*, 6(3):187-214, July 1988.

[25] S.B. Navathe, R. Elmasri and J. Larson. Integrating User Views in Database Design. In *IEEE Computer*, 19:1, Jan. 1986.

[26] E. Rosch. Principles of Categorization. In *Cognition and Categorization*, E. Rosch and B. Lloyd, (eds.), Lawrence Erlbaum Associates, 1978.

ISSN: 2324-9773

[27] R.E. Stepp and R.S. Michalski. Conceptual Clustering: Inventing goal- oriented classifications of structured objects. In *Machine Learning: An Artificial Intelligence Approach*, Vol. 2, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.), Tioga, Palo Alto, CA, 1986.

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

[29] M. Zemankova and A. Kandel. Implementing imprecision in information systems. In *Information Sciences*, 37(1,2,3):107-141, Dec. 1985.

ISSN: 2324-9773

Anwar, T., Beck, H., & Navathe, S. (1991). Conceptual Clustering in Database Systems. 2nd ASIS SIG/CR Classification Research Workshop, 11-32.
doi:10.7152/acro.v2i1.12544

The Semantic Classification in the Temporal Theory of Psychoanalysis

Antal F. Borbely, M.D.
675 West End Ave. Apt. 2A
New York, NY 10025

INTRODUCTION

Many aspects of classification have been researched but the laws underlying changing classifications need to be focussed on. Psychoanalysis is a science dealing with the changing classifications within the individual. Aided by its theory, evolving over the last hundred years, psychoanalysis undertakes to provide actual scientific guidance in a person's changing his or her classifications or priorities in life. Such changes regularly occur in a psychoanalysis where, in the analytic setting, an analyst and an analysand form a unique relationship, usually lasting several years, for the purpose of analyzing the analysand's life. The analysand discovers his or her unique epistemologic categories as they developed based on a unique personal history. These epistemologic categories reflect both biographic achievements as well as constraints and, usually for the first time, become accessible for both understanding and change. While understanding and change can and does, to some extent, occur outside psychoanalysis, the psychoanalytic setting provides a unique opportunity to exclusively and systematically focus on such understanding and change.

The way the analysand classifies his or her life in terms of motivations, biographic events, plans, symptoms, fears and wishes undergoes a profound and non - predictable change in the course of an analysis. One's priorities change, developmental arrest in certain domains is overcome so that psychological development can resume, one's concepts change with each insight a little and, after a cumulative process, nothing quite means what it once meant.

Hundreds of thousands of persons have by now undergone an analysis and the literature concerning psychoanalysis may number more than one million books and journals. Few other sciences have as profoundly influenced human culture in the last hundred years. While psychoanalysis suffered for a long time from a self misunderstanding of being mainly a biological science because of the great importance it accorded to sexual and aggressive drives, it deserves to be viewed as an information science par excellence. This paper attempts to justify this claim and thus to contribute towards bridging psychoanalysis and information science. Specifically, it will use the temporal theory (Borbely, 1987) to touch on classification research connected problems, for example concept acquisition, the basis for semantic classes, knowledge representation schemes, reasoning with classification schemes, compatibility between classification schemes, information retrieval, parallel processing, distributive processing, and hybrid reasoning (Frisch, 1991).

THE PSYCHOANALYTIC SETTING:

The analysand is lying in the supine position on the couch while the analyst sits behind him or her, out of the analysand's sight. The analysand free associates, that is, verbalizes whatever comes to mind without censoring it, while the analyst listens and interprets the analysand's productions. By following the "basic rule" to free associate the usual dialogue between two subjects is suspended.

ISSN: 2324-9773

One reason for this arrangement is to allow for important communications, often unknown to the analysand, to emerge. This becomes possible by suspending the usual classifications for dialogue, for logical communication, for communication constrained by consideration for the other person's feelings or decorum.

The analyst's interpretations allow for growth promoting insights on the part of the analysand to occur based on the fact that the analyst, having undergone an analysis him or herself, does not shy away from what causes anxiety to the analysand. The analyst can therefore lift the analysand's communications to a higher, more mature classification level. Unconscious childhood conflicts involving important childhood figures surface and are reexperienced in the relationship with the analyst, to whom feelings and role assignments are transferred. The latter endeavors to remain neutral (value tolerant) and abstinent (not gratifying his or her own needs), so as to be able to show the analysand the "transference" in a convincing, insightful way. What appeared to the analysand to be an experience in the present, turns out to be a dramatization of past constellations which now, for the first time, can be understood and resolved in terms of adult categories of thinking and experiencing (Reclassification!).

Such understanding is not a mere intellectual one, as it is occurring within an experiential field containing many of the emotions of the original childhood setting. This allows for a resolution of old conflicts and a reintegration of its elements into the mainstream of psychological development leading not only to symptom removal but, more importantly, to a general creativity enhancement.

SOME ELEMENTS OF GENERAL PSYCHOANALYTIC THEORY:

Psychoanalytic theory has passed through several stages in these hundred years and continues to develop. While this development can be described from different points of view (different classification schemes can serve for the purpose of a historic overview) I chose as the most fruitful classification scheme for the history of psychoanalytic theory the one deriving from the concepts "defense" and "defended against."

One of the most basic clinical findings by psychoanalysis was that the human mind is conflicted and that often the real causes for the conflicts are, at first, unknown both to analyst as well as analysand. At first S. Freud (1900) thought that all defense was synonymous with repression. By "repression" he meant the making unconscious of what once was conscious in order to avoid anxiety. Later, Freud described other defense mechanisms: projection, introjection, undoing, rationalization, displacement, denial, isolation (of thought from affect) and others. All defenses served the purpose of warding off anxiety by psychologically severing the realistic connections between elements of one's life constellation, thus sacrificing the realistic classification of reality events for the (temporary) peace of mind. The "defended against" was usually seen as unacceptable drive derivatives or demands and prohibitions of the conscience ("superego"), which had to be covered by amnesia or through other defensive manipulation disguised and thus made "innocuous." A brief clinical vignette may serve as an example:

A man explains that he would like to stay in analysis forever as he likes to learn about himself. Upon analysis this explanation turns out to contain the following additional meanings: the

ISSN: 2324-9773