Mining Distributed Databases for Attribute Definitions

Jan M. Żytkow and Zbigniew W. Raś

Comp. Science Dept. Univ. of North Carolina, Charlotte, N.C. 28223, USA
also Institute of Comp. Science, Polish Academy of Sciences, Warsaw, Poland

ABSTRACT

The paper focuses on discovery of knowledge needed to establish the shared meaning of attributes in a network of distributed autonomous databases. In this paper we concentrate on the role of equations as definitions of attribute values. We briefly describe various applications of such definitions, including predictions, knowledge verification, intelligent query answering and several others. We present an interface between a Distributed Autonomous Knowledge System *DAKS* and a discovery system 49er. To find knowledge useful in defining attributes missing in one database, the discovery mechanism of 49er can be applied to other databases. *DAKS* makes requests for definitions and then manages the discovered definitions, verifies their consistency and applies them in its query-answering mechanism. To put a system of equation-based attribute definitions on a firm theoretical foundation we introduce semantics which justifies empirical equations in their definitional role. This semantics augments the earlier developed semantics for rules used as attribute definitions.

Keywords: distributed databases, discovery of equations, distributed knowledge systems, operational semantics

1. INTRODUCTION

In many fields, such as medical, manufacturing, banking, military and educational, similar databases are kept at many sites. Each database stores information about local events and uses attributes suitable for a local task, but since the local situations are similar, the majority of attributes are compatible among databases. Yet minor exceptions can occur. An attribute may be missing in one database, while it occurs in many others. Different tests may be applied in different hospitals, but their results may be linked by simple dependencies. For instance, one hospital applies test a_1 to their diabetics patients while another applies test a_2. There may be a direct relationship between both tests, and if it is known, data collected at one hospital can be used at the other.

Missing attributes lead to problems. A doctor new at unit S may be familiar with test a, and issues a query that seeks patients who match a description that uses results of test a, only to realize that because the results of a are not available in S, the query cannot be answered. The same query would work in other databases but the doctor is interested in identifying patients candidates at the unit S.

Many computational mechanisms can be used to define values of an attribute. Ras, Zemankova and Maitan[7,14] (1989-1990) introduced a mechanism which seeks rules in the form "If Boolean-expression(x) then a(x)=v" and then applies them as partial definitions of a. A number of strategies can be used to find rules describing decision attributes in terms of classification attributes. We should mention here such systems like *LERS* (developed by J. Grzymala-Busse), *DQuest* (developed by W. Ziarko) or *AQ*15 (developed by R. Michalski).

In this paper we present a mechanism that expands attribute definitions from rules to equations. Other types of knowledge, however, for instance taxonomies, could be also used as definitions. System 49*er*, developed by Zytkow and Zembowicz,[16] allows to describe attributes as functions of other attributes. Such functional descriptions can hold for all data or in data subsets. When attributes are numerical, functional descriptions take on the form of equations.

Definitions that are used to compute attribute values of objects are often called operational definitions. They are common in science, where values of each attribute are computed in many ways, depending on different applications. Operational semantics has been introduced by Bridgman[1] and developed by Carnap[2] and many others, including semantics of coherent sets of operational definitions developed by Zytkow[17] and applied in robotic experiments.[18]

Further author information: e-mail: zytkow@uncc.edu or ras@uncc.edu

2. APPLICATIONS OF EMPIRICAL EQUATIONS

Empirical equations are an important class of regularities that can be discovered in databases. In this paper we concentrate on the role of equations as definitions of attribute values.

Functional dependencies in the form of equations are a convenient form of knowledge. They are very concise and apply in many ways. They can be used in making predictions, explanations and inference. Many equations can be combined to perform these functions.

2.1. Making predictions and computing missing values

$a = r(a_1, a_2, ..., a_m)$ can be directly used to predict values $a(x)$ of attribute a for object x by substituting the values of $a_1(x), a_2(x), ..., a_m(x)$ if all are available. Missing values of a can be computed that way as well as values of an attribute a which is not present in a given database.

2.2. Detecting wrong values

When we suspect that some values of a may be wrong, an equation imported from another database may be used to verify them. An equation acquired at the same database may be used, too, if the discovery mechanism is able to distinguish the wrong values as the outliers. For instance, patterns discovered in the clean data can be applied to discovery of wrong values in the raw data

2.3. Consistency verification

An equation that has been derived from data is an empirical generalizations. Although it has been verified in the database in which it was derived, and may be reliable, we cannot trust it unconditionally.

It is a good practice to seek further verification, especially if the predicted values of a go beyond the range of values of a for which the equation has been derived. The verification may come from additional knowledge that can be used as alternative definitions. Ras[12,13] (1997-1998) used rules coming from various sites and verified their consistency. His system can use many strategies which find rules describing decision attributes in terms of classification attributes and then compare the resultant rules.

Equations that are generated at different sites can be used, too, to cross-check the consistency of knowledge and data coming from different databases. If the values of a computed by two independent equations are approximately equal, each of the equations receives further confirmation as a computational method for a.

2.4. Understanding the meaning of an attribute

All equations by which values of a can be defined expand the understanding of a. Attribute understanding is often initially inadequate when we receive a new dataset for the purpose of data mining. We may know the domain of values of a, but we do not understand a's detailed meaning, so that we do not know how to apply background knowledge and we cannot interpret the knowledge discovered about a. In such cases, an equation that links a poorly understood attribute a with attributes $a_1, ..., a_n$, the meaning of which is known, explains the meaning of a in terms of $a_1, ..., a_n$.

2.5. Query answering

Intelligent Query Answering System[15] (IQAS) for Distributed Autonomous Information Systems (DAIS) is concerned with identifying all objects satisfying a given description in one of the component systems of DAIS.

To give an example of an intelligent query answering strategy, assume that a query issued against a database DB includes an attribute a which is unknown in DB. The meaning of a can be determined in another database DB_1 that includes a as well as some other attributes $b_1, ..., b_n$ shared by DB and DB_1. A request for a procedure that defines a by some of $b_1, ..., b_n$ in DB_1 is initially treated as a request for a definition of a in terms of $b_1, ..., b_n$. Such a definition can take on the form of a system of rules, an equation, a taxonomy and so on. If discovered, the definition in each of these forms can be converted into a procedure that allows the computation of the value of a, given the values of $b_1, ..., b_n$. Such a procedure is then used to answer the original query in DB.

3. REQUEST FOR A DEFINITION

For the purpose of inducing equations from data we could adapt various discovery systems.[3,4,9,8,5,16] We have chosen the 49er system[16] because it is tuned to data available in databases. The system allows to describe one attribute as a function of other attributes and it seeks equations when attributes are numerical. It has demonstrated successful applications in many databases coming from various domains.

Special requirements must be satisfied by an equation so that it can be used as a definition of a given numerical attribute. One of the main problems with the search for equations is that the best fit can be always found for any dataset in any class of models (equation schemas). But is the best fit good enough? How good is good enough? Equations often provide rough estimates of patterns, but an estimate may not be sufficient for a definition. How good must be a fit of an equation to the data so that this equation can be used as a definition?

When we know the desired accuracy of fit, we know how to evaluate a candidate equation against data. In database applications there is a "natural" limit on the accuracy for those attributes, common in all sorts of databases, whose values are numerical and discrete. Consider an attribute whose values are integers, such as weight in pounds or age in years. The error (accuracy) of fit can be derived from the granularity of the domain. For any three adjacent values v_1, v_2, v_3 in the ascending order, the acceptable accuracy of determination of v_2 is $(v_3 - v_1)/4$. For instance, for the age in years, the accuracy is half year. That error rate is entirely satisfactory, but sometime even a worse fit is still acceptable for a definition.

When the required accuracy of fit ε is provided, for each candidate equation the probability can be estimated that a given dataset could have been generated from that equation compounded with residua generated from normal distribution $N(0, \varepsilon)$. A demanding probabilistic significance threshold such as $Q \geq 0.01$ is also needed.

In summary, a *quest* for a definition in the form of an equation includes[15]:

1. the attribute a for which a definition is sought in the form of an equation;
2. the accuracy of attribute a for each value in the domain of A;
3. a set of attributes $\{a_1, ..., a_n\}$ which can be used in the definition;

The discovery proces can be decomposed into three stages which have been describe elsewhere[15]: (1) functionality test, (2) finding preliminary equations, (3) refining the equations until they become acceptable. The resultant equation, if any, has the form $a = f(a_{i_1}, ..., a_{i_k})$, and it fits the data within the probability threshold $Q = 0.01$, which we use as a default value for definitions.

4. RECURSIVE SEARCH FOR EQUATIONS

Let us re-examine the intelligent query answering. When an attribute a is needed but unreachable in database S_1, a request for a definition of a is issued to other sites in the distributed database system. The request specifies the attributes $a_1, ..., a_n$ available in S_1.

When attribute a and a subset $\{a_{i_1}, ..., a_{i_k}\}$ of $\{a_1, ..., a_n\}$ are available in another database S_2, 49er's discovery mechanism is invoked to search S_2 for a computational procedure by which values of a can be computed from values of some of $a_{i_1}, ..., a_{i_k}$. If discovered, such a procedure is returned to S_1 and can be used in numerous ways.

Earlier[15] we consider a computational mechanism that searches for equations suitable in a role of definitions of a. But there are numerous situations when this mechanism must be expanded and applied recursively.

4.1. Non-overlapping attribute sets

First, there may be no database which contains a and any of $\{a_1, ..., a_n\}$. Suppose that database S_3 contains a and $\{b_1, ..., b_n\}$, and an equation $a = f(b_1)$ has been discovered in S_3. It cannot be used in S_1, because b_1 is unavailable there. But there may be still another database S_4 that includes b_1 and some of $\{a_1, ..., a_n\}$. An equation may be discovered that defines b_1 in terms of $\{a_1, ..., a_n\}$. That equation can be substituted into $a = f(b_1)$ leading finally to an equation that can be applied in S_1.

This can be illustrated with the following example of simple database schemas:

```
S1: a1, a2, a3
S3: a, b1, b2
S4: a1, b1
```

4.2. Search for a sufficient fit

Second, there may be a database S_2 that includes a and some of $\{a_1, ..., a_n\}$. But no equation that defines a through any of $\{a_1, ..., a_n\}$ has a sufficient fit to play the role of a definition. In this situation, the search for a definition can be expanded. Perhaps an equation is discovered that has a sufficient fit to play the role of a definition, but in addition to some of $\{a_1, ..., a_n\}$ it uses b_1, unavailable in the original database S_1. To remedy this situation, the search can be applied in other databases for a definition of b_1 in terms of attributes $\{a_1, ..., a_n\}$ available in S_1. As a caution, this search should not consider the same equation which has been discovered in S_2.

4.3. Reduct-driven discovery

A third situation, to a degree similar to the second, occurs within the rough sets approach.[10] A search for reducts[6] is often applied to determine some or all sets of attributes which are sufficient to distinguish between all records in a given database. Attributes in a reduct are then used to build concept definitions. It may happen that in every reduct there are attributes additional to the set $\{a_1, ..., a_n\}$ available in S_1. Again, an equation discovered in such a reduct cannot be directly applied in S_1, so that it must be augmented by other equations that define attributes not available in S_1.

4.4. Empirical contents in a set of definitions

But there is a more systematic reason why the search for equations should continue, even if it has been successful. Multiple equations give a chance for cross-verification as their predictions can be compared. Each consistent prediction provides extra justification in the system of definitions, while each inconsistency calls for further empirical analysis of data and definition improvements.

4.5. Recursive discovery algorithm

The following algorithm can be used to search recursively for an attribute definition:

```
Algorithm: find definitions of attribute A that are applicable in DB

for each database X
  if A is available in X then
     seek definition of A in X
     Keep all definitions of A in X on list def(A,X)
for each X and each definition DEF in def(A,X)
  if all attributes that define A are available in DEF,
     then add DEF to list of definitions of A
     else
       for each attribute B missing in DB
         find definitions of attribute B that are applicable in DB
```

5. SHARED SEMANTICS IN A DISTRIBUTED DB SYSTEM

Before we define the shared meaning of attributes in a Distributed Autonomous Knowledge System $DAKS$ we must introduce the notions of an information system and a distributed information system.

5.1. Information system

By an information system we mean a structure $S = (X, A, V)$, where X is a finite set of objects, A is a finite set of attributes, and $V = \bigcup\{V_a : a \in A\}$ is a set of their values. We assume that:

- V_a, V_b are disjoint for any $a, b \in A$ such that $a \neq b$,
- $a : X \longrightarrow V_a$ is a function for every $a \in A$.

Instead of a, we will often write $a_{[S]}$ to denote that a in an attribute in S.

5.2. Distributed information system

By a distributed information system[12] we mean a pair $DS = (\{S_i\}_{i \in I}, L)$ where:

- $S_i = (X_i, A_i, V_i)$ is an information system for any $i \in I$,
- L is a symmetric, binary relation on the set I,
- I is a set of sites.

Distributed information system $DS = (\{S_i\}_{i \in I}, L)$ is consistent if:

$$(\forall i)(\forall j)(\forall x \in X_i \cap X_j)(\forall a \in A_i \cap A_j)\ (a_{[S_i]}(x) = (a_{[S_j]})(x).$$

In the remainder of this paper we assume that $DS = (\{S_i\}_{i \in I}, L)$ is a distributed information system which is consistent. Also, we assume that $S_j = (X_j, A_j, V_j)$ and $V_j = \bigcup\{V_{ja} : a \in A_j\}$, for any $j \in I$.

We will use A to name the set of all attributes in DS, $A = \bigcup\{A_j : j \in I\}$.

5.3. Shared operational semantics

The shared semantics is defined for the set A of all attributes in all databases in DS. For each attribute a in A, the operational meaning of a is defined by:

1. the set of databases in which a is available directly: $S_i : a \in A_i$;
2. the set of databases in which a has been defined; and the set of definitions in each database. Definitions can be equations, boolean forms, etc.
3. the set of databases in which the definitions of a can be used, because the defining attributes are available there. An attribute a is a defined attribute in an information system S if:
   (a) a definition DEF of a has been discovered in one of databases in DS;
   (b) all other attributes in the definition DEF are present in S; in such cases they can be put together in a JOIN table and DEF can be directly applied.

5.4. Syntax of definitions

We will now define the syntax of definitions in the form of equations. Partial definitions are included, as they are often useful. They can also be automatically discovered by 49er.[16] In the next subsection we give an interpretation of partial definitions. We expand here the terminology introduced in.[15]

Functors are the building blocks from which equations and inequalities can be formed. Those in turn are the building blocks for partial definitions. Assume that x is a variable over X_i and $r_1, r_2, ..., r_k$ are functors. Also, we assume here that m_j is the number of arguments of the functor r_j, $j = 1, 2, .., k$. The number of arguments can be zero. A zero argument functor is treated as a constant.

By a set of *s(i)-atomic-terms* we mean a least set $T0_i$ such that:

- $\mathbf{0}, \mathbf{1} \in T0_i$,

for any symbolic attribute $a \in A_j$,

- $[a(x) = w] \in T0_i$ for any $a \in A_i$ and $w \in V_{ia}$,
- $\sim [a(x) = w] \in T0_i$ for any $a \in A_i$ and $w \in V_{ia}$,

for any numerical attributes $a, a_1, a_2, ..., a_{m_j}$ in A_i,

- $[a(x)\ \rho\ r_j(a_1, a_2, ..., a_{m_j})(x)] \in T0_i$, where $\rho \in \{=, \leq, \geq\}$

s(i)-atomic-terms of the form $[a(x) = w]$ and $[a(x) = r_j(a_1, a_2, ..., a_{m_j})(x)]$ are called equations

By a set of *s(i)-partial-definitions* (s(i)-p-defs in short) we mean a least set T_i such that:

- if $t(x) \in T0_i$ is an equation, then $t(x) \in T_i$,
- if $a \in A_i$ and $t(x)$ is a conjunction of *s(i)-atomic-terms* and $s(x)$ is an equation, then $[t(x) \longrightarrow s(x)] \in T_i$,
- if $t_1(x), t_2(x) \in T_i$, then $(t_1(x) \vee t_2(x)), (t_1(x) \wedge t_2(x)) \in T_i$.

For simplicity we often write t instead of $t(x)$.

The set of *s(I)-defs* is defined in a similar way to *s(i)-p-defs*: the set V_i is only replaced by $\bigcup\{V_j : j \in I\}$ and the set A_i is replaced by $\bigcup\{A_j : j \in I\}$. *s(I)-defs* represents all possible candidate definitions built from attributes that can come from different databases (information systems).

Standard interpretation M_i of *s(i)-p-defs* in a distributed information system $DS = (\{S_j\}_{j \in I}, L)$ is defined as follows:

- $M_i(\mathbf{0}) = \emptyset$, $M_i(\mathbf{1}) = X_i$
- $M_i(a(x) = w) = \{x \in X_i : a_{[S_i]}(x) = w\}$,
- $M_i(\sim (a(x) = w)) = \{x \in X_i : a_{[S_i]}(x) \neq w\}$,
- for any $\rho \in \{=, \leq, \geq\}$,
  $M_i(a(x)\ \rho\ r_j(a_1, a_2, ..., a_{m_j})(x)) = \{x \in X_i : a_{[S_i]}(x)\ \rho\ r_j(a_{1[S_i]}(x), a_{2[S_i]}(x), ..., a_{m_j[S_i]}(x))\}$,
- $M_i([t \longrightarrow s]) = \{x \in X_i : \text{if } [x \in M_i(t)] \text{ then } x \in M_i(s)]\}$,
- if t_1, t_2 are *s(i)-p-defs*, then
  $M_i(t_1 \vee t_2) = M_i(t_1) \cup M_i(t_2)$,
  $M_i(t_1 \wedge t_2) = M_i(t_1) \cap M_i(t_2)$,
  $M_i(t_1 = t_2) = (\text{if } M_i(t_1) = M_i(t_2) \text{ then } T \text{ else } F)$, where T stands for $True$ and F for $False$

Let us assume that $[t1 \longrightarrow (a_1(x) = w1)], [t2 \longrightarrow (a_2(x) = w2)]$ are *s(i)-p-defs*. We say that they are S_i-consistent, if either $a_1 \neq a_2$ or $M_i(t_1 \wedge t_2) = \emptyset$ or $w1 = w2$. Otherwise, these two *s(i)-p-defs* are called S_i-inconsistent.

Similar definitions apply when $w1$ and $w2$ in those partial definitions are replaced by $r_1(a_1, a_2, ..., a_{m_j})(x)$ and $r_2(a_1, a_2, ..., a_{m_j})(x)$.

6. DISCOVERY LAYER

In this section, we introduce the notion of a discovery layer, distributed autonomous knowledge system and *s(I)-queries* which can be processed locally at site i using operational definitions if needed. We introduce the concept of a dynamic operational semantics to reflect the dynamics of constantly changing discovery layers.

Notice that while in the previous sections partial definitions have been interpreted at the sites at which all relevant attributes have been present, we now consider definitions imported from site i to site k.

By a discovery layer D_{ki} we mean any $s(i)$-consistent set of partial definitions, of the two types specified below, which are satisfied, by means of the interpretation M_k, by most of the objects in S_k:

- $[t \longrightarrow [a(x) = r_m(a_1, a_2, ..., a_m)(x)]]$, where $a_1, a_2, ..., a_m \in A_i$ and $a \in A_k$ and t is a conjunction of atomic terms that contain attributes that occur both in A_i and in A_k
- $[t \longrightarrow (a(x) = w)]$, where $a \in A_k$ and t satisfies the same conditions as above.

Suppose that a number of partial definitions have been imported to site i from a set of sites K_i. All those definitions can be used at site i.

Thus, the discovery layer for site $i \in I$ is defined as a subset of the set $D_i = \bigcup\{D_{ki} : k \in K_i\}$ where K_i is a set of sites.

By Distributed Autonomous Knowledge System $(DAKS)$ we mean $DS = (\{(S_i, D_i)\}_{i \in I}, L)$ where $(\{S_i\}_{i \in I}, L)$ is a distributed information system and D_i is a discovery layer for a site $i \in I$.

Let us now consider queries that contain attributes which are locally unavailable. In[15] we introduced i-operational semantics N_i of *s(I)-queries* in $DS = (\{(S_i, D_i)\}_{i \in I}, L)$ where $S_i = (X_i, A_i, V_i)$ and $V_i = \bigcup\{V_{ia} : a \in A_i\}$.

Each pair (S_i, D_i) in DS was called a knowledge system for a site i.

The growth of the knowledge system can be driven by a systematic quests for partial definitions, but it can be also driven, more opportunistically, by queries which cannot be answered because of missing attributes.

Previously,[15] interpretation $N_i(t)$ of a query t assumes that all the rules and equations needed to resolve attributes listed in t which are not in A_i are already in D_i. In contrast, our Intelligent Query Answering System proposed here is based on a dynamic operational semantics which can be seen as an operational extension of the interpretation N_i. Assume now that the notation $t = t(b_1, b_2, ..., b_n)$ means that $b_1, b_2, ..., b_n$ are all the attributes listed in t which do not belong to A_i. If a query t is submitted to the Intelligent Query Answering System $IQAS$ of the site i, then for each attribute $b_j, j \in \{1, 2, ..., n\}$ our system $IQAS$ finds all $S_{k,j}$ such that $b_j \in A_{k,j}$ and $(A_i \cap A_{k,j}) \neq \emptyset$. In a system $S_{k,j}$, using $49er$, we may discover a number of partial definitions, such as $[t_j \longrightarrow b_j(x) = w_j])$. Among all these partial definitions our system $IQAS$ will choose a coverage by partial definitions with a minimal number of attributes $b_{subscript}$ listed in $t_j(b_{j1}, b_{j2}, ..., b_{jm})$ which are not members of A_i. This winning optimal coverage is stored in a discovery layer D_i of the knowledge system (S_i, D_i). If the list $(b_{j1}, b_{j2}, ..., b_{jm})$ in t_j is not empty, then for each b_{j*} in $b_{j1}, b_{j2}, ..., b_{jm}$ our $IQAS$ will search for a new optimal quest $[t_{j*} \longrightarrow b_{j*}(x) = w_{j*}])$. All these optimal coverages will be stored in a discovery layer D_i. This process will continue until all new optimal coverages needed to resolve t will involve only attributes in A_i.

Clearly, if for each non-local attribute we collect rules and equations from many sites of $DAKS$ and then resolve any inconsistencies among them, the resulting rules and equations in the local discovery layer have more chance to be locally true.

REFERENCES

1. Bridgman, P.W. 1927. *The Logic of Modern Physics.* The Macmillan Company.
2. Carnap, R. 1936. Testability and Meaning, *Philosophy of Science, 3.*
3. Dzeroski, S. & Todorovski, L. 1993. Discovering Dynamics, *Proc. of 10th International Conference on Machine Learning*, 97-103.
4. Falkenhainer, B.C. & Michalski, R.S. 1986. Integrating quantitative and qualitative discovery: The ABACUS system. *Machine Learning, 1*, 367-401.
5. Kokar, M.M. 1986. Determining Arguments of Invariant Functional Descriptions, *Machine Learning, 1*, 403-422.
6. Kryszkiewicz, M., Rybinski, H., *Reducing information systems with uncertain attributes*, ISMIS'96, LNCS/LNAI, Springer, Vol. 1079, 1996, 285-294
7. Maitan, J., Ras, Z., Zemankova, M., "Query handling and learning in a distributed intelligent system", in *Methodologies for Intelligent Systems, IV*, (Ed. Z.W. Ras), North Holland, 1989, 118-127
8. Moulet, M. 1992. A symbolic algorithm for computing coefficients' accuracy in regression, in: Sleeman D. & Edwards P. eds. *Proc. of Ninth Intern. Conference on Machine Learning.*
9. Nordhausen, B. & Langley, P. 1993. An Integrated Framework for Empirical Discovery, *Machine Learning, 12*, 17-47.
10. Pawlak, Z., *Rough Classification*, International Journal of Man-Machine Studies, Vol. 20, 1984, 469-483
11. Ras, Z.W., "Dictionaries in a distributed knowledge-based system", in *Proceedings of Concurrent Engineering Conf.*, Pittsburgh, August 29-31, 1994, Concurrent Technologies Corp., 383-390
12. Ras, Z., "Resolving queries through cooperation in multi-agent systems", in *Rough Sets and Data Mining* (Eds. T.Y. Lin, N. Cercone), Kluwer Academic Publishers, 1997, 239-258
13. Ras, Z., Joshi, S., "Query approximate answering system for an incomplete DKBS", in *Fundamenta Informaticae Journal*, IOS Press, Vol. 30, No. 3/4, 1997, 313-324
14. Ras, Z., Zemankova, M, "Intelligent query processing in distributed information systems", in *Intelligent Systems: State of the Art and Future Directions*, Z.W. Ras, M. Zemankova (Eds), Ellis Horwood Series in Artificial Intelligence, London, England, November, 1990, 357-370
15. Ras, Z., Zytkow, J.,"Discovery of Equations to Augment the Shared Operational Semantics in Distributed Autonomous BD System", in *PAKDD'99 Proceedings*, LNCS/LNAI, Springer-Verlag, will appear
16. Żytkow, J. & Zembowicz, R., "Database Exploration in Search of Regularities", in *Journal of Intelligent Information Systems*, No. 2, 1993, 39-81.
17. Żytkow, J. An interpretation of a concept in science by a set of operational procedures, in: *Polish Essays in the Philosophy of the Natural Sciences*, Krajewski W. ed. Boston Studies in the Philosophy of Science, Vol.68, Reidel 1982, p.169–185.
18. Żytkow, J.M., Zhu, J., and Zembowicz R. Operational Definition Refinement: a Discovery Process, *Proceedings of the Tenth National Conference on Artificial Intelligence*, The AAAI Press, 1992, p.76–81.