Discovery of Equations and the Shared Operational Semantics in Distributed Autonomous Databases

Zbigniew W. Raś and Jan M. Żytkow

Computer Science Dept. University of North Carolina, Charlotte, N.C. 28223
e-mail: ras@uncc.edu & zytkow@uncc.edu
also Institute of Computer Science, Polish Academy of Sciences

Abstract. 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. Such definitions can be used in many ways that we briefly describe. We present a discovery mechanism that specializes in finding equations that can be used as definitions. We introduce the notion of shared operational semantics. It consists of an equation-based system of partial definitions and it is used as a tool for knowledge exchange between independently built databases. This semantics augments the earlier developed semantics for rules used as attribute definitions. To put the shared operational semantics on a firm theoretical foundation we developed a formal interpretation which justifies empirical equations in their definitional role.

1 Shared semantics for distributed autonomous DB

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, an attribute may be missing in one database, while it occurs in many others. For instance, different military units may apply the same battery of personality tests, but some tests may be not used in one unit or another. Similar irregularities are common with medical data. Different tests may be applied in different hospitals.

Missing attributes lead to problems. A recruiter new at a given unit may query a local database S_1 to find candidates who match a desired description, only to realize that one component a_1 of that description is missing in S_1 so that the query cannot be answered. The same query would work in other databases but the recruiter is interested in identifying suitable candidates in S_1.

In this paper we introduce operational semantics that provides definitions of missing attributes. Definitions are discovered by an automated process. They are used for knowledge exchange between databases and jointly form an integrated semantics of our Distributed Autonomous Knowledge System.

The task of integrating established database systems is complicated not only by the differences between the sets of attributes but also by differences in structure and semantics of data, for instance, between the relational, hierarchical and network data models. We call such systems heterogeneous. The notion of an intermediate model, proposed by Wiederhold, is very useful in dealing with the heterogeneity problem, because it describes the database content at a relatively high abstract level, sufficient to guarantee homogeneous representation of all databases. In this paper we propose a discovery layer to be an intermediate model for networked databases. Our discovery layer contains rules and equations extracted from a database.

To eliminate the heterogeneity problem D. Maluf and G. Wiederhold [5] proposed to use an ontology algebra which provides the capability for interrogating many knowledge resources, which are largely semantically disjoint, but where *articulations* have been established that enable knowledge interoperability. The main difference between our approaches is that they do not use the intermediate model for communication, and they did not consider automated discovery systems as knowledge sources.

Navathe and Donahoo [6] proposed that the database designers develop a metadata description (an intermediate model) of their database schema. A collection of metadata descriptions can then be automatically processed by a schema builder to create a partially integrated global schema of a heterogeneous distributed database. In contrast, our intermediate model (a discovery layer) is built without any help from database designers. Its content is created through the automated knowledge extraction from databases.

1.1 Methods that can construct operational definition

Many computational mechanisms can be used to define values of an attribute. Ras et al. [4], [11] (1989-1990) introduced a mechanism which first seeks and then applies as definitions rules in the form "If Boolean-expression(x) then a(x)=w" which are partial definitions of attribute a. Recently, Prodromidis & Stolfo [8] mentioned attribute definitions as a useful task. In this paper we expand attribute definitions from rules to equations. We call them operational definitions because each is a mechanism by which the values of a defined attribute can be computed. Many are partial definitions, as they apply to subsets of records that match the "if" part of a definition.

1.2 Shared semantics in action: query answering

Many real-world situations fit the following generic scenario. A query q that uses attribute a is "unreachable" at database S_1 because a is missing in S_1. A request for a definition of a is issued to other sites in the distributed autonomous database systems. The request specifies attributes $a_1, ..., a_n$ available at 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, a discovery mechanism is invoked to search for knowledge at S_2. A computational mechanism can be discovered by which values of a can

be computed from values of some of $a_{i_1}, ..., a_{i_k}$. If discovered, such a mechanism is returned to site s_1 and used to compute the unknown values of a that occur in query q.

The same mechanism can apply if attribute a is available at site s_1, but some values of a are missing. In that case, the discovery mechanism can be applied at s_1, if the number of the available values of a is sufficiently large.

2 Other applications

Functional dependencies in the form of equations are a succinct, convenient form of knowledge. They can be used in making predictions, explanations and inference. $a = r_m(a_1, a_2, ..., a_m)$ can be directly used to predict values $a(x)$ of a for object x by substituting the values of $a_1(x), a_2(x), ..., a_m(x)$ if all are available. If some are not directly available, they may be predicted by other equations.

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 clean data can be applied to discovery of wrong values in the raw data.

Equations that are used to compute missing values are empirical generalizations. Although they may be reliable, we cannot trust them unconditionally, and it is a good practice to seek their further verification, especially if they are applied to the expanded range of values of a. The verification may come from additional knowledge that can be used as alternative definitions. Ras [9], [10] (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. It has been used in conjunction with such systems like $LERS$ (developed by J. Grzymala-Busse) or $AQ15$ (developed by R. Michalski).

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.

All equations by which values of a can be computed 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 cannot 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 $a_1, ..., a_n$.

3 Request (Quest) for a definition

For the purpose of inducing equations from data we could adapt various discovery systems [7][2]. We have chosen the 49er system (Zytkow and Zembowicz, 1993) because it applies to data available in databases and because it searches for equations that apply to subsets of data, in addition to equations that apply to all data. 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 are needed for an equation that 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 (equations). But is the best fit good enough? How good is good enough? Equations often provide rough estimates of patterns, but those estimates may be not good for definitions. How good must be a fit of an equation so that this equation can be used as a definition?

When we know the desired accuracy of fit, we know how to evaluate equations against data. In database applications there is a "natural" limit on the accuracy for those common attributes 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 a year. That error rate is entirely satisfactory, but sometime even a worse fit is still acceptable from a definition.

Consider the situations when the required accuracy of fit ε_i is provided for all data $(x_i, y_1, \varepsilon_i),\ i = 1, ..., n$. For each candidate equation the probability can be estimated that $(x_i, y_1, \varepsilon_i),\ i = 1, ..., n$ could have been generated by $f(x_i) + r_i)$, where r_i is generated from normal distribution $N(0, \varepsilon)$. A demanding probability threshold such as $Q \geq 0.01$ is also needed.

In summary, the quest for a definition in the form of an equation includes:

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

The resultant equations, if any, have the form $a(x) = f(a_{i_1}, ..., a_{i_k})$, and they fit the data within a demanding probability threshold $Q = 0.01$, which is the default value for definitions.

3.1 Functionality Test

Plenty of time can be saved if equations are not sought in data which do not satisfy the mathematical definition of functional relationship.

Definition: Given a set D of value pairs (v_i, w_i), $i = 1, \ldots, N$ of two attributes a and b, and the range V_a of a; ***b* is a function of *a* in V_a** iff for each v_0 in V_a, there is exactly one value w_0 of b, such that (v_0, w_0) is in D.

The following algorithm approximates this definition. It determines whether it is worthwhile to search for an equation that fits the data.

Algorithm: Test approximate functional relationship between a and b

given the contingency table of actual record counts and set V_a of values of a
  AV $\leftarrow$ average number of records per cell
  for each value in V_a
    find all groups of cells with adjacent values of b and counts $>$ AV
    if # of groups $> \alpha$ **then return** NO–FUNC
  if average # of groups $> \beta$ **then return** NO–FUNC **else return** FUNCTION

This algorithm is controlled by two modifiable parameters, α and β, which measure local (α) and global (β) uniqueness of b; that is, the number of values of b for the same value of a. The default values used by 49er is $\alpha = 2$ for data from databases, and $\beta \approx 1.5$. For $\alpha = 3$ the functionality test fails when for a value in V_a there are more than 3 adjacent groups of cells with above average density of points. This higher value 3 of α solves the problem of rare outliers, allowing up to 2 outliers if they happen rarely. However, many outliers or frequent multiple values in y should fail the test, therefore the value of β is much smaller and close to 1. Note that both parameters set to 1 corresponds to the strict mathematical definition of functionality given above. Presence of error, noise, and other data imperfections force values of α and β to be larger than 1. The noise handling by varying the number of cells in the table is treated in detail by Żytkow & Zembowicz (1993).

The same mechanism applies when we want to determine a functional relation in a set of data tuples D of the size $1 + k$ for $k \geq 2$. If the test is successful, equations in the form $b(x) = r(b_1, ..., b_k)$ are sought. If the test fails, it will be applied to subsets of data when they are generated by 49er.

3.2 Equation Finder's search

The task of equation finding can be formally defined by the input of n datapoints which come from projection of attributes a and b from data table S, and the computation of required accuracy of b: $(v_i, w_i, \varepsilon_i)$, $i = 1, ..., n$. The output is the list of acceptable equations. Since the equations are initially 2-d and can be subsequently refined, the acceptance threshold is at this stage less demanding ($Q \geq 0.0001$)

Equation Finder's search can be decomposed into (1) generation of new terms, (2) selection of pairs of terms, (3) generation and evaluation of equations for each pair of terms. The combination of these three searches can be summarized by the following algorithm:

Algorithm: Find Equation

T $\leftarrow$ (A B) ; the initial list of terms for search #1
old-T $\leftarrow$ NIL ; the list of terms already used
E $\leftarrow$ a set of polynomial equation models ; list of models for search #3
loop until new terms in T exceed threshold of complexity

```
2T ← list of new pairs of terms created from T and old-T
                    ; the list generated by search #2, initially (A B)
for each pair in 2T and for each model in E
    find and evaluate the best equation
if at least one equation accepted, then
    return all accepted equations and HALT the search
old-T ← old-T augmented with T
T ← list of new terms created from old-T
```

For each pair of terms (either original attributes a and b or terms x and y) generated by search #1, and for each polynomial up to the maximum pre-specified degree, search # 3 proposes polynomial models $y = f(x, a_0, \ldots, a_q)$, which are then solved for b, if possible, and compared with the models considered earlier. For each equation which comes out as a new one, the best values are found for the parameters (coefficients) a_0, ..., a_q, and error values ε_{a_0}, ..., ε_{a_q} for each parameter. Each polynomial coefficient for which $|a_i| < \varepsilon_{a_i}$ is removed. The equation is accepted as a definition of b by a if the significance measure exceeds a threshold, set to 0.01. If that threshold is not met, a refinement process (not treated in this paper) applies to the equation if the significance measure exceeds a threshold, set by default at 0.0001. The significance is based on χ^2 test and the number of degrees of freedom, that is on the number of data points minus the number of parameters in the equation.

Correlation analysis is often used as a measure of linearity of a relation. Our approach offers a far broader search for equations. Many textbook examples show that correlation values are close to zero (that means, no correlation) even though a sharp functional dependency occurs in the data. Our Equation Finder returns well-fitted equations in many such cases.

3.3 Efficiency

The functionality test operates on contingency tables. Since the size of the table is typically small compared to the size of data, and the test requires one pass through the table, it is extremely efficient. It also saves large amount of time because it prevents a far more costly equation finding search, when it cannot be successful. Generation of a contingency table is linear in the number of records. The number of contingency tables is linear in the number of attributes considered. If the number of original attributes is very large, various techniques of feature selection can be used to reduce their number. For a comprehensive treatment of feature selection, see [3]. Sampling, in turn, can reduce the number of records. Equation finding is linear in the number of records and is proportional to the number of models considered. The space of Equation Finder search can be limited in different ways by setting the parameter values for each search, such as depth of search and the list of operators. The potentially most costly is search in the subsets of data, but it can be also adjusted to the available resources, by limiting the depth of search.

4 A shared semantics of equations in a Distributed Autonomous Knowledge System

In this section each *database* in Distributed Autonomous Database Systems will be extended to a *knowledge system*. We first recall the notions of an information system and a distributed information system. Next we define the shared meaning of attributes in a Distributed Autonomous Knowledge System $DAKS$.

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.

By a **distributed information system** [9] we mean a pair $DS = (\{S_i\}_{i \in I}, L)$ where:

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

In this paper we assume a distributed information system $DS = (\{S_i\}_{i \in I}, L)$ which is consistent, that is,

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

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, $A = \bigcup\{A_j : j \in I\}$.

4.1 Shared operational semantics

The shared semantics (see [12]) is defined for the set A of all attributes in all information systems in DS. For each attribute a in A, the operational meaning of a is defined by:

1. the set of (pointers to) information systems in which a is available: $\{S_i : a \in A_i\}$;
2. the set of information systems in which a definition of a has been derived, jointly with the set of definitions in each information system. Definitions can be equations, boolean forms, etc.
3. the set of information systems in which a definition 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 an S_i 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.

4.2 Equations as partial definitions: the syntax

We will now define the syntax of definitions in the form of equations. Partial definitions are included, as they are often useful. In the next subsection we give an interpretation of partial definitions.

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 the 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,

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

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

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

- if $t(x) \in T0_i$ is an equation, then $t(x) \in T_i$,
- if $t(x)$ 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 *s(I)-p-defs* represent all possible candidate definitions built from attributes that can come from different information systems in DS. *s(I)-p-defs* is defined in a similar way to *s(i)-p-defs*: the set V_i is replaced by $\bigcup\{V_j : j \in I\}$ and the set A_i is replaced by $\bigcup\{A_j : j \in I\}$.

4.3 Equations as partial definitions: the interpretation

By a standard interpretation of *s(i)-p-defs* in $S_i = (X_i, A_i, V_i)$ of a distributed information system DS we mean a function M_i such that:

- $M_i(\mathbf{0}) = \emptyset$, $M_i(\mathbf{1}) = X_i$
- $M_i(a(x) = w) = \{x \in X_i : a_{[S_i]}(x) = w\}$,
- $M_i(\sim (a(x) = w)) = \{x \in X_i : a_{[S_i]}(x) \neq w\}$,
- for any $\rho \in \{=, \leq, \geq\}$,
  $M_i((a\ \rho\ r_j(a_1, a_2, ..., a_{m_j}))(x)) =$
  $\{x \in X_i : a_{[S_i]}(x)\ \rho\ r_j(a_{1[S_i]}(x), a_{2[S_i]}(x), ..., a_{m_j[S_i]}(x))\}$,

- $M_i([t \longrightarrow s]) = \{x \in X_i : \text{if } [x \in M_i(t)] \text{ then } x \in M_i(s)]\}$,
- if t_1, t_2 are *s(i)-p-defs*, then
  $M_i(t_1 \vee t_2) = M_i(t_1) \cup M_i(t_2)$, $M_i(t_1 \wedge t_2) = M_i(t_1) \cap M_i(t_2)$,
  $M_i(t_1 = t_2) = (\text{if } M_i(t_1) = M_i(t_2) \text{ then } True \text{ else } False)$.

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 $r_2(a_1, a_2, ..., a_{m_j})(x)$.

5 Discovery layer

In this section, we introduce the notions of a discovery layer and a distributed autonomous knowledge system. Also, 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 $s(i)-p-defs$ have been interpreted at the sites at which all relevant attributes have been present, we now consider $s(I)-defs$ imported from site k to site i.

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

- $[t \longrightarrow [(a = r_m(a_1, a_2, ..., a_m))(x)]]$, 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 $D_i = \bigcup\{D_{ki} : k \in K_i\}$.

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 $i \in I$.

Figure 1 shows the basic architecture of $DAKS$ (WWW interface and a query answering system $kdQAS$ that can request and use *s(I)-p-defs* are also added to each site of $DAKS$).

Predicate logic and i-operational semantics are used to represent knowledge in $DAKS$. Many other representations are, of course, possible. We have chosen predicate logic because of the need to manipulate $s(I)-defs$ syntactically without changing their meaning. This syntactical manipulation of *s(I)-defs* will be handled by $IQAS$. By designing an axiomatic system which is sound we are certain that the transformation process for *s(I)-p-defs* based on these axioms

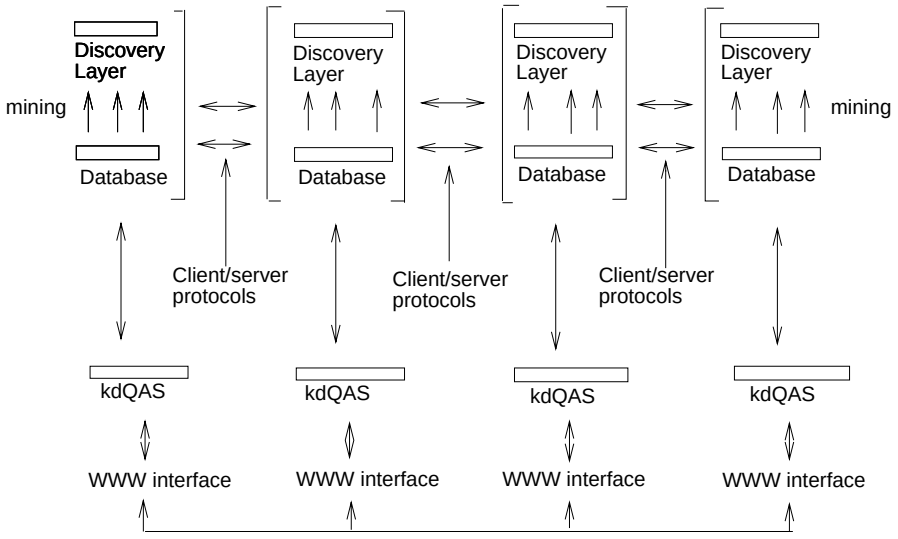

Fig. 1. Distributed Autonomous Knowledge System

either will not change their meaning or will change it in a controlled way. It will produce *s(i)-p-defs* approximating the initial *s(I)-p-defs*.

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

Let M_i be a standard interpretation of *s(i)-p-defs* in S_i and $C_i = \bigcup\{V_k : k \in I\} - V_i$. By i-operational semantics of *s(I)-p-defs* in $DS = (\{(S_i, D_i)\}_{i\in I}, L)$ where $S_i = (X_i, A_i, V_i)$ and $V_i = \bigcup\{V_{ia} : a \in A_i\}$, we mean the interpretation N_i such that:

- $N_i(\mathbf{0}) = \emptyset$, $N_i(\mathbf{1}) = X_i$
- for any $w \in V_{ia}$,
  $N_i(a(x) = w) = M_i(a(x) = w)$, $N_i(\sim (a(x) = w)) = M_i(\sim (a(x) = w))$
- for any $w \in C_i \cap V_{ka}$ where $k \neq i$,
  $N_i(a(x) = w) = \{x \in X_i : ([t \longrightarrow [a(x) = w]] \in D_i \wedge x \in M_i(t))\}$
  $N_i(\sim (a(x) = w)) = \{x \in X_i : (\exists v \in V_a)[(v \neq w) \wedge ([t \longrightarrow [a(x) = v]] \in D_i) \wedge (x \in M_i(t))]\}$
- for any $w \in C_i \cap V_{ka}$ where $k \neq i$ and a is a numeric attribute,
  $N_i((a(x) = w)) = \bigcup\{x \in X_i : (\exists y \in M_k[a(y) = w = r_m(a_1, a_2, ..., a_m)])$
  $[M_i([a_{1[S_i]}(x) = a_{1[S_k]}(y)] \wedge [a_{2[S_i]}(x) = a_{2[S_k]}(y)] \wedge ... \wedge [a_{m[S_i]}(x) = a_{1[S_k]}(y)])$
  $\wedge [a(y) = w = r_m(a_1, a_2, ..., a_m)] \in D_i\}$
  $N_i(\sim (a(x) = w)) = X_i - N_i(a(x) = w)$
- for any $s(I)$-terms t_1, t_2
  $N_i(t_1 \vee t_2) = N_i(t_1) \cup N_i(t_2)$, $N_i(\sim (t_1 \vee t_2)) = (N_i(\sim t_1)) \cap (N_i(\sim t_2))$,

$N_i(t_1 \wedge t_2) = N_i(t_1) \cap N_i(t_2)$, $N_i(\sim (t_1 \wedge t_2)) = (N_i(\sim t_1)) \cup (N_i(\sim t_2))$, $N_i(\sim\sim t) = N_i(t)$.

- for any $s(I)$-terms t_1, t_2
  $N_i(t_1 = t_2) = (\text{ if } N_i(t_1) = N_i(t_2) \text{ then } True \text{ else } False)$

The i-operational semantics N_i represents a pessimistic approach to evaluation of *s(I)-p-defs* because of the way the non-local *s(I)-atomic-terms* are interpreted (their lower approximation is taken).

References

1. Batini, C., Lenzerini, M., Navathe, S., "A comparative analysis of methodologies for database schema integration", in *ACM Computing Surveys*, Vol 18, No. 4, 1986, 325-364
2. Dzeroski, S. & Todorovski, L. 1993. Discovering Dynamics, *Proc. of 10th International Conference on Machine Learning*, 97-103.
3. Liu, H. & Motoda, H. 1998. *Feature selection for knowledge discovery and data mining,* Kluwer.
4. 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
5. Maluf, D., Wiederhold, G., "Abstraction of representation for interoperation", in *Proceedings of Tenth International Symposium on Methodologies for Intelligent Systems*, LNCS/LNAI, Springer-Verlag, No. 1325, 1997, 441-455
6. Navathe, S., Donahoo, M., "Towards intelligent integration of heterogeneous information sources", in *Proceedings of the Sixth International Workshop on Database Re-engineering and Interoperability*, 1995
7. Nordhausen, B. & Langley, P. 1993. An Integrated Framework for Empirical Discovery, *Machine Learning, 12*, 17-47.
8. Prodromidis, A.L. & Stolfo, S., "Mining databases with different schemas: Integrating incompatible classifiers", in *Proceedings of The Fourth Intern. Conf. onn Knowledge Discovery and Data Mining*, AAAI Press, 1998, 314-318
9. 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
10. 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
11. 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
12. 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.
13. Żytkow, J. & Zembowicz, R., "Database Exploration in Search of Regularities", in *Journal of Intelligent Information Systems*, No. 2, 39-81
14. 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.

This article was processed using the LaTeX macro package with LLNCS style