Reducts-driven Query Answering for Distributed Autonomous Knowledge Systems

Zbigniew W. Raś

Computer Science Dept., University of North Carolina, Charlotte, N.C. 28223
Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
e-mail: ras@uncc.edu or ras@wars.ipipan.waw.pl

Abstract. In this paper we show the role of equations and rules as definitions of attribute values. Such definitions can be used in many ways but in this paper we concentrate on their applications to intelligent query answering. They are used as a tool for knowledge exchange between independently built knowledge systems. Intelligent Query Answering System (IQAS) decides which sites are optimal for extracting definitions needed to solve a query. These optimal sets are identified by a search strategy based on rough sets theory, introduced by Z. Pawlak [8]. We introduce the notion of shared operational semantics. To put the shared operational semantics on a firm theoretical foundation we proposed a formal interpretation which justifies empirical equations and rules in their definitional role.

Keywords: discovery of equations, distributed knowledge systems, operational semantics, intelligent query answering, reducts.

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 Database System (DADS). A knowledge system is a database coupled with a certain set of definitions extracted from other databases. Informally a Distributed Autonomous Knowledge System (DAKS) is obtained from DADS by replacing a database at each of its sites by a knowledge system. The knowledge systems in DAKS are constantly changing and these changes are activated by user queries.

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],[13] (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. Also, Prodromidis & Stolfo [9] mentioned attribute definitions as a useful task. In this paper attribute definitions are in the form of rules and equations. We call them operational because of 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 Intelligent 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.

1.3 Reducts and its applications in knowledge discovery

The notion of a reduct, based on yet another concept, attribute dependency, was introduced by Pawlak [8]. In the process of rule discovery, redundant attributes in rules can be avoided, provided that these rules are constructed from reducts. Several exact algorithms computing all reducts and some subclasses of reducts have been developed by RS community. The most known collection of such algorithms is the RS library developed at Warsaw University of Technology.

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. $a = r_m(a_1, a_2, ..., a_n)$ can be directly used to predict values $a(x)$ of $a$ for object $x$ by substituting the values of $a_1(x), a_2(x), ..., a_n(x)$ 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, their definitions (in a form of equations or rules) imported from another databases may be used to verify them. Definitions 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 [11], [12] (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 LEaR5 (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 for a definition

For the purpose of inducing definitions from data we could adapt various discovery systems [7],[2]. System 4rer (Zytkow and Zemkowicz, 1993) 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.

Special requirements are needed for an equation that can be used as a definition of a given numerical attribute. Equations often provide rough estimates of patterns, but those estimates may be not good for definitions.

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.

4 Shared semantics of equations in a Distributed Autonomous Knowledge System

In this section each database in Distributed Autonomous Database Systems is extended to a knowledge system. First, we 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 $a \neq b$,
- $a : X \rightarrow V_a$ 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* [11] 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 $I$.

A distributed information system $DS = (\{ S_i \}_{i \in I}, L)$ is consistent if the following condition holds:

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

In the remainder of this paper we assume that $DS = (\{ S_i \}_{i \in I}, L)$ is consistent. Also, we assume that $S_j = (X_j, A_j, V_j)$ and $V_j = \bigcup \{ V_{a_j} : a \in A_j \}$, 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 [16]) 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 $S_i$; 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. These in turn are the building blocks for partial definitions. Assume that \( x \) is a variable over \( X_i \) and \( r_1, r_2, \ldots, r_k \) are functors. Also, we assume here that \( m_j \) is the number of arguments of the functor \( r_j, \ j = 1, 2, \ldots, k \). 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 \( T_0 \) such that:

- \( 0, 1 \in T_0 \),

for any attribute \( a \in A_j \),

- \([a(x) = w] \in T_0 \) for any \( a \in A_i \) and \( w \in V_ia \),

- \(~[a(x) = w] \in T_0 \) for any \( a \in A_i \) and \( w \in V_ia \),

for any numerical attributes \( a, a_1, a_2, \ldots, a_m, \) in \( A_i \),

- \([a \rho r_j(a_1, a_2, \ldots, a_m)](x) \in T_0 \), where \( \rho \in \{=, \leq, \geq\} \)

\( s(i)\)-atomic-terms of the form \([a(x) = w] \) and \([a = r_j(a_1, a_2, \ldots, a_m)](x) \) are called equations.

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

- if \( t(x) \in T_0 \) 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 \([t(x) \rightarrow s(x)] \in T_i \),

- if \( t_1(x), t_2(x) \in T_i \), then \((t_1(x) \lor t_2(x)), (t_1(x) \land t_2(x)) \in T_i \).

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:

- \( M_i(0) = \emptyset \), \( M_i(1) = X_i \),

- \( M_i(a(x) = w) = \{x \in X_i : a_{[S]}(x) = w\} \),
two types specified below which are satisfied, by means of the interpretation $M_k$, by the majority of objects in $S_k$ (support for $s(k)$-defs has to be high):

- $[(a = r_m(a_1, a_2, ..., a_m)(x))]$, where $a_1, a_2, ..., a_m \in A_l \cap A_k$ and $a \in A_k$,
- $[t \rightarrow (a(x) = w)]$, where $a \in A_k$ and $t$ is a conjunction of atomic terms that contain attributes that occur both in $A_l$ and in $A_k$.

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 $i \in I$. 

Atomic term $[a(x) = w]$ can be seen as a simple query which extracts all objects having property $w$, whereas the term $[a \rho r_j(a_1, a_2, ..., a_m)]$ can be seen as a definition of an attribute $a$. Term $[t \rightarrow s]$ can be interpreted as a partial definition of $s$.

Let us assume that $[t_1 \rightarrow (a_1(x) = w_1)], [t_2 \rightarrow (a_2(x) = w_2)]$ are $s(i)$-defs. We say that they are $S_i$-consistent, if either $a_1 \neq a_2$ or $M_i(t_1 \land t_2) = \emptyset$ or $w_1 = w_2$. Otherwise, these two $s(i)$-defs are called $S_i$-inconsistent.

Similar definitions apply when $w_1$ and $w_2$ in those partial definitions are replaced by $r_j(a_1, a_2, ..., a_m)(x)$ and $r_j(a_1, a_2, ..., a_m)(x)$.

In [10] we gave a strategy to repair inconsistent rules. Similar method can be used to repair $S_i$-inconsistent $s(i)$-defs.

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 $s(i)$-defs have been interpreted at the sites at which all relevant attributes have been present, now consider $s(l)$-defs imported from site $k$ to site $i$.

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

- $[(a = r_m(a_1, a_2, ..., a_m)(x))]$, where $a_1, a_2, ..., a_m \in A_l \cap A_k$ and $a \in A_k$,
- $[t \rightarrow (a(x) = w)]$, where $a \in A_k$ and $t$ is a conjunction of atomic terms that contain attributes that occur both in $A_l$ and in $A_k$.
Figure 1 shows the basic architecture of DAKS (a query answering system IQAS that can handle $s(I)-p-defs$ is also added to each site of DAKS).

Predicate calculus and operational semantics are used to represent knowledge in DAKS. Many other representations are, of course, possible. We have chosen formal logic approach because of the need to manipulate $s(I)-p-defs$ syntactically without losing information about their semantical meaning. Syntactical manipulation of $s(I)-p-defs$ is handled by IQAS. By designing an axiomatic system which is sound we have an assurance that the transformation process for $s(I)-p-defs$ based on these axioms may change (if it has to) their semantical meaning only in a controlled way. An outcome of the transformation process is a set of $s(i)-p-defs$ approximating the initial $s(I)-p-defs$.

Figure 2 shows a part of IQAS which is responsible for query transformation. This part of IQAS can be replaced by a rough transformation engine shown in Figure 3.
If for each non-local attribute we collect rules and equations from many sites of DAKS and then resolve all inconsistencies among them, the local confidence in resulting operational definitions is high since they represent consensus of many sites.

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 $V_i = \bigcup \{V_i : a \in A_i\}$, we mean the interpretation $N_i$ such that:

- $N_i(\emptyset) = \emptyset$, $N_i(1) = X_i$
- for any $w \in V_{ia}$,
  - $N_i(a(x) = w) = M_i(a(x) = w)$, $N_i(\neg (a(x) = w)) = M_i(\neg a(x) = w))$
- for any $w \in C_i \cap V_{ia}$ where $k \neq i$,
  - $N_i(a(x) = w) = \{x \in X_i : ([t \rightarrow [a(x) = w]] \in D_i \land x \in M_i(t))\}$
  - $N_i(\neg (a(x) = w)) = \{x \in X_i : (\exists v \in V_a)((v \neq w) \land ([t \rightarrow [a(x) = v]] \in D_i) \land (x \in M_i(t))\}$
- for any $w \in C_i \cap V_{ia}$ where $k \neq i$ and $a$ is a numeric attribute,
  - $N_i(\{(a(x) = w) = \bigcup \{x \in X_i : (\exists y \in M_k[a(y) = w = r_m(a_1, a_2, ..., a_m)]\}$
  - $[M_i([a_{1[s_1]}(x) = a_{2[s_1]}(y)]) \land [a_{2[s_1]}(x) = a_{3[s_1]}(y)] \land ... \land [a_{m[s_1]}(x) = a_{1[s_1]}(y)])$\}
  - $N_i(\neg (a(x) = w)) = X_i - N_i(a(x) = w)$
- for any $s(i)$-terms $t_1, t_2$
  - $N_i(t_1 \lor t_2) = N_i(t_1) \cup N_i(t_2)$, $N_i(\neg (t_1 \lor t_2)) = (N_i(\neg t_1)) \cap (N_i(\neg t_2))$,
  - $N_i(t_1 \land t_2) = N_i(t_1) \cap N_i(t_2)$, $N_i(\neg (t_1 \land t_2)) = (N_i(\neg t_1)) \cup (N_i(\neg t_2))$,
  - $N_i(\neg \neg t) = N_i(t)$.
- for any $s(i)$-terms $t_1, t_2$
  - $N_i(t_1 = t_2) = (\text{if } N_i(t_1) = N_i(t_2) \text{ then } \text{True else } \text{False})$

The $i$-operational semantics $N_i$ represents a pessimistic approach to evaluation of $s(i)-p-defs$ because of the way the non-local terms $[a(x) = w]$ are interpreted (their lower approximation is taken).

We can replace $N_i$ by $i$-operational semantics $J_i$ representing optimistic approach to evaluation of $s(i)-p-defs$. Namely, we define:

- $J_i(a(x) = w) = X - N_i(\neg (a(x) = w))$, 

Fig. 3. Intelligent Query Rough Answering System (IQRAS)
- $J_i(\sim (a(x) = w)) = X - N_i(a(x) = w),$
- $J_i(t) = N_i(t)$ for any other $t$.

In optimistic approach to evaluation of queries, upper approximation of non-local terms $[a(x) = w], \sim (a(x) = w)$ is taken.

Also, we can propose rough operational semantics $R_i$ defined as $R_i(t) = [N_i(t), J_i(t)]$ for any $s(l)$-p-defs $t$. Rough operational semantics has a natural advantage of either $N_i$ or $J_i$. Clearly, if interpretations $N_i$ and $J_i$ of a term $t$ give us the same sets of objects, both approximations (lower and upper) are semantically equal.

6 Reducts-driven query answering

In this section we recall the notion of a reduct (see Pawlak [8]) and show how it can be used to improve query answering process in DAKS.

Let us assume that $S = (X, A, V)$, is an information system and $V = \bigcup \{V_a : a \in A\}$. Let $B \subseteq A$. We say that $x, y \in X$ are indiscernible by $B$, denoted $[x \approx_B y]$ if $(\forall a \in B)[a(x) = a(y)]$.

Now, assume that both $B_1, B_2$ are subsets of $A$. We say that $B_1$ depends on $B_2$ if $\approx_{B_2} \subseteq \approx_{B_1}$. Also, we say that $B_1$ is a covering of $B_2$ if $B_2$ depends on $B_1$ and $B_1$ is minimal.

By a reduct of $A$ in $S$ (for simplicity reason we say reduct of $S$) we mean any covering of $A$.

**Example.** Assume the following scenario:

- $S_1 = (X_1, \{C, D, E, F, G\}, V_1)$, $S_2 = (X_2, \{B, D, E, K, L\}, V_2)$,
- $S_3 = (X_3, \{K, E, F, B, L\}, V_3)$ are information systems,
- User submits a query $q = q(D, E, B, F)$ to the query answering system $IQAS_1$ associated with system $S_1$,
- Systems $S_1, S_2$ are parts of DAKS.

Attribute $B$ is foreign for a system $S_1$ so the query answering system associated with $S_1$ has to contact other sites of DAKS requesting a definition of $B$ in terms of $\{C, D, E, F, G\}$. Such a request is denoted by $< B : C, D, E, F, G >$. Assume that the system $S_2$ is contacted. The definition of $B$ extracted from $S_2$ involves only attributes $\{C, D, E, F, G\} \cap \{B, D, E, K, L\} = \{B, D, E\}$. Clearly it can happen that no reduct of $S_2$ is a superset of $\{B, D, E\}$. The situation can be even worst (the intersection of $\{B, D, E\}$ with any reduct of $S_2$ can be empty).

In each of these cases, the query answering system $IQAS$ for $S_1$ will search for a site of DAKS with a reduct being a superset of $\{B, D, E\}$. If the search
fails, then IQAS will look for a site of DAKS and its reduct which maximally overlaps with the set \{B, D, E\}. Assume that \{B, D, K\} in \(S_2\) is such a reduct. Then, the definition of \(B\) in terms of attributes \{D, K\} will be extracted from \(S_2\) and the query answering system of \(S_2\) will contact other sites of DAKS requesting a definition of \(K\) in terms of attributes \{C, D, E, F, G\}. This new definition will be sent directly to the site associated with \(S_1\).

Figure 4 shows the process of resolving query \(q\) in the example above.

![Figure 4](image-url)  
**Fig. 4.** Process of resolving a query by IQAS

7 Conclusion

This paper is an extension of papers \([11], [12]\) by allowing definitions in a form of equations and by involving reducts in the process of search for an optimal site for extracting such definitions.

References


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