Collaborative query processing in DKS controlled by reducts

Zbigniew W. Raś[*,°] and Agnieszka Dardzińska[+]

[*] University of North Carolina, Department of Computer Science
Charlotte, N.C. 28223, USA
[°] Polish Academy of Sciences, Institute of Computer Science
Ordona 21, 01-237 Warsaw, Poland
[+]Bialystok University of Technology, Department of Mathematics
Wiejska 45 A, 15-351 Bialystok, Poland
ras@uncc.edu or adardzin@uncc.edu

Abstract. Traditional query processing provides exact answers to queries. In this paper, we introduce, so called, non-local queries which allow us to use attributes outside the local domain. Definitions of these attributes, if only exist, can be extracted from databases at other sites. Before, these definitions can be locally applied, problems related to their different semantics have to be resolved first. Rough-ontology is one of the possible tools which can be used here quite successfully to tackle the problem. We introduce a tree-resolution for a non-local query which helps us to identify all steps which should be followed to replace a non-local query by a semantically similar local one.

1 Introduction

In the common experience of data miners, each new database acquired for the purpose of data mining is a source of recurring problems with data. Some attributes are not understood, some others seem to be understood in principle, but some of their values seem strange, out of range or missing. Overcoming problems of this sort is among the prominent consumers of research effort at the stage typically called data preprocessing.

We must treat those problems carefully, as serious misconceptions will result when the users of data and knowledge understand them differently from providers and as knowledge derived from problematic data may be misleading.

A seemingly different problem is created by attributes, whose values are codes, such as disease or treatment code, occupation code, or customer category. They are nominal attributes with large numbers of values. Such attributes do not lead to useful generalizations because they have very many values and must be treated as nominal, whereas they hide plenty of information. When we ask how those values were encoded, we can find their definitions in terms of attributes which are much more conducive to data analysis and knowledge discovery. With the use of its definition, one coded attribute can be typically replaced by several attributes, each with a small numbers of values and clear meaning. For instance, a code for broken bone indicates the broken bone, the location, the type of fracture, and several other properties.

The problems experienced in mining a single database multiply when we try combining data and knowledge from many databases. When we combine relational tables, for instance, we want to avoid JOIN on attributes the values of which are only superficially equal. We must carefully compare attributes which occur under same names and verify compatibility of their values. Ontologies and operational definitions can be instrumental in those comparisons.

Despite problems, when multiple databases are available, new sources of knowledge are enabled. Operational definitions (see [Ras][8]) can not only explain meaning of attributes in one database but can be used to bring meaning to attributes in other databases. Operational definitions can be even more useful in creating a shared semantics for distributed databases.

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 locally collected information, 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 but equivalent tests that measure a common medical condition may be applied in different hospitals. A doctor who researches effectiveness of a particular treatment may find it difficult to compare the treatment in several hospitals, because they use different tests and because one test replaces another over the course of time. But if the relations between values of different tests can be discovered from data or found in medical references, many attributes acquire a shared meaning and many datasets can be used together. As the meaning can be expanded from data to knowledge, the shared semantics can be used to combine knowledge coming from many sources ([3], [4], [5], [6], [7], [8]).

In this paper, we show how operational definitions (see [Ras][8]) can be constructed in an efficient way. We search first for sequences of definitions at server sites and next apply them to transfer a user query to a form which is manageable by a client site. To handle differences in semantics, if any, rough-ontologies can also be used.

2 Distributed Information Systems

In this section, we recall definitions of an information system and a distributed knowledge system. Also, we introduce the notion of local and global queries.

By an *information system* we mean $S = (X, A, V)$, 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 $a \neq b$,
- $a : X \longrightarrow 2^{V_a} - \{\emptyset\}$ is a function for every $a \in A$.

By a *distributed information system* 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.

Let $S_j = (X_j, A_j, V_j)$ for any $j \in I$. In the remainder of this paper we assume that $V_j = \bigcup\{V_{ja} : a \in A_j\}$.

From now on, in this section, we use A to denote the set of all attributes in DS, $A = \bigcup\{A_j : j \in I\}$. Also, by V we mean $\bigcup\{V_j : j \in I\}$.

Before introducing the notion of a knowledgebase, we begin with a definition of $s(i)$-terms and their standard interpretation M_i in $DS = (\{S_j\}_{j \in I}, L)$, where $S_j = (X_j, A_j, V_j)$ and $V_j = \bigcup\{V_{ja} : a \in A_j\}$, for any $j \in I$.

By a set of $s(i)$-terms (also called a set of local queries for site i) we mean a least set T_i such that:

- $\mathbf{0}, \mathbf{1} \in T_i$,
- $w \in T_i$ for any $w \in V_i$,
- if $t_1, t_2 \in T_i$, then $(t_1 + t_2), (t_1 * t_2), \sim t_1 \in T_i$.

Definition of DS-terms (also called a set of global queries) is quite similar (we only replace T_i by $\bigcup\{T_i : i \in I\}$ in the definition above).

We say that:

- $s(i)$-term t is *primitive* if it has the form $\prod\{w : w \in U_i\}$ for any $U_i \subseteq V_i$,
- $s(i)$-term is in *disjunctive normal form* (DNF) if $t = \sum\{t_j : j \in J\}$ where each t_j is primitive.

Similar definitions can be given for DS-terms.

The expression:

select $*$ from $Flights$
where $airline$ = "$Delta$"
and $departure\_time$ = "$morning$"
and $departure\_airport$ = "$Charlotte$"
and $aircraft$ = "$Boeing$"

is an example of a non-local query (DS-term) in a database

$Flights(airline, departure\_time, arrival\_time,$
$departure\_airport, arrival\_airport)$.

Semantics of $s(i)$-terms is seen as the standard interpretation M_i in a distributed information system $DS = (\{S_j\}_{j \in I}, L)$. It is defined as follows:

- $M_i(\mathbf{0}) = \emptyset$, $M_i(\mathbf{1}) = X_i$
- $M_i(w) = \{x \in X_i : w \in a(x)\}$ for any $w \in V_{ia}$,
- if t_1, t_2 are s(i)-terms, then
  $M_i(t_1 + t_2) = M_i(t_1) \cup M_i(t_2)$,
  $M_i(t_1 * t_2) = M_i(t_1) \cap M_i(t_2)$,
  $M_i(\sim t_1) = X_i - M_i(t_1)$.

Now, we are ready to introduce the notion of (k,i)-rules, for any $i \in I$. We use them to build a knowledgebase at site $i \in I$.

By (k,i)-rule in $DS = (\{S_j\}_{j \in I}, L)$, $k, i \in I$, we mean a triple (c,t,s) such that:

- $c \in V_k - V_i$,
- t, s are $s(k)$-terms in DNF and they both belong to $T_k \cap T_i$,
- $M_k(t) \subseteq M_k(c) \subseteq M_k(t + s)$.

Any (k,i)-rule (c,t,s) in DS can be seen as a definition of c which is extracted from S_k and can be used in S_i.

For any (k,i)-rule (c,t,s) in $DS = (\{S_j\}_{j \in I}, L)$, we say that:

- $(t \rightarrow c)$ is a k-certain rule in DS,
- $(t + s \rightarrow c)$ is a k-possible rule in DS.

Now, we are ready to define a knowledgebase D_{ki}. Its elements are called definitions of values of attributes from $V_k - V_i$ in terms of values of attributes from $V_k \cap V_i$.

Namely, D_{ki} is defined as a set of (k,i)-rules such that:
if $(c,t,s) \in D_{ki}$ and the equation $t_1 = \sim (t + s)$ is true in M_k, then $(\sim c, t1, s) \in D_{ki}$.
The idea here is to add definition of $\sim c$ to D_{ki} if a definition of c is already in D_{ki}. It will allow us to approximate (learn) concept c from both sites, if needed.

By a knowledgebase for site i, denoted by D_i, we mean any subset of $\bigcup\{D_{ki} : (k,i) \in L\}$. If definitions can not be extracted at a remote site, partial definitions (c,t) corresponding to $(t \rightarrow c)$, if available, can be extracted and stored in a knowledgebase at a client site.

By Distributed Knowledge System (DKS) we mean $DS = (\{(S_i, D_i)\}_{i \in I}, L)$ where $(\{S_i\}_{i \in I}, L)$ is a distributed information system, $D_i = \bigcup\{D_{ki} : (k,i) \in L\}$ is a knowledgebase for $i \in I$.

In [Ras/Dardzinska][6] we gave an example of DKS and its query answering system that handles global queries.

Also, in [Ras/Dardzinska][6], we stated that the set of semantics $\{M_i\}_{i \in I}$ representing all involved sites in processing query q is a partially ordered set. If it preserves monotonicity property for $+$ and $*$, then $M_{min} = \bigcap\{M_i\}_{i \in I}$ (the greatest lower bound) and $M_{max} = \bigcup\{M_i\}_{i \in I}$ (the least upper bound) can be taken as two common semantics for processing query q. The pair $[M_{min}, M_{max}]$ can be seen as a rough-semantics.

3 Query Answering Based on Reducts

In this section we recall the notion of a reduct (see [Pawlak][2]) and show how a query can be processed by a distributed knowledge system.

Let us assume that $S = (X, A, V)$, is an information system and $V = \bigcup\{V_a : a \in A\}$. Let $B \subset A$. 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 $\approx_{B_2} \subset \approx_{B_1}$. Also, we say that B_1 is a covering of B_2 if B_2 depends on B_1 and B_1 is minimal.

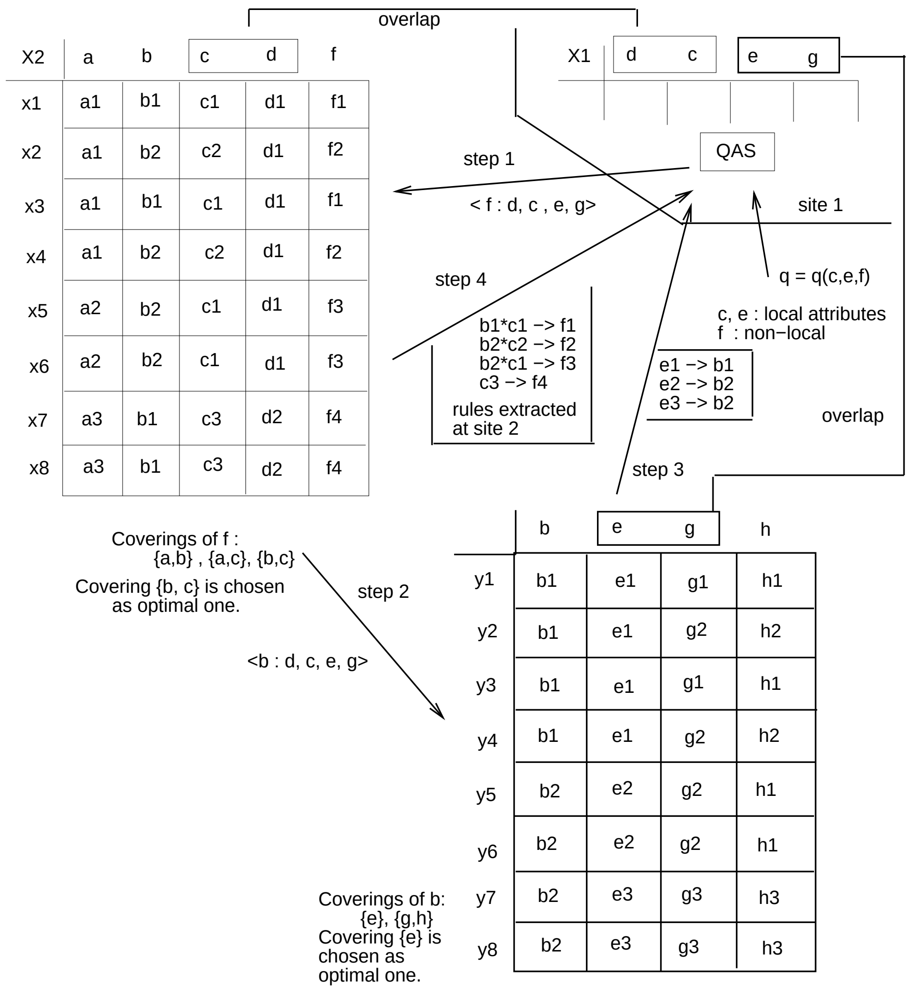
By a reduct of A in S (for simplicity reason we say A-reduct of S) we mean any covering of A.

Example. Assume the following scenario (see Figure 1):

- $S_1 = (X_1, \{c, d, e, g\}, V_1)$, $S_2 = (X_2, \{a, b, c, d, f\}, V_2)$, $S_3 = (X_3, \{b, e, g, h\}, V_3)$ are information systems,
- User submits a query $q = q(c, e, f)$ to the query answering system QAS associated with system S_1,
- Systems S_1, S_2, S_3 are parts of DKS.

Attribute f is non-local for a system S_1 so the query answering system associated with S_1 has to contact other sites of DKS requesting a definition of f in terms of $\{d, c, e, g\}$. Such a request is denoted by $< f : d, c, e, g >$. Assume that system S_2 is contacted. The definition of f, extracted from S_2, involves only attributes $\{d, c, e, g\} \cap \{a, b, c, d, f\} = \{c, d\}$. There are three f-reducts (coverings of f) in S_2. They are: $\{a, b\}, \{a, c\}, \{b, c\}$. The optimal f-reduct is the one which has minimal number of elements outside $\{c, d\}$. Assume that $\{b, c\}$ is chosen as an optimal f-reduct in S_2.

Then, the query answering system of S_2 will contact other sites of DKS requesting definition of b (which is non-local for S_1) in terms of attributes $\{d, c, e, g\}$. If definition of b is found, then it is sent to QAS of the site 1. Also, the definition of f in terms of attributes $\{b, c\}$ will be extracted from S_2 and send to S_1. Figure 1 shows all the steps needed to resolve query $q = q(c, e, f)$. This steps can be represented as a sequence of equations $q = q(c, e, f) = q(c, e, f(b, c)) = q(c, e, f(b(e), c))$. Also, a tree called "tree-resolution for q", represented by Figure 2, gives an alternative representation of that process. Definition of q is called



X2	a	b	c	d	f
x1	a1	b1	c1	d1	f1
x2	a1	b2	c2	d1	f2
x3	a1	b1	c1	d1	f1
x4	a1	b2	c2	d1	f2
x5	a2	b2	c1	d1	f3
x6	a2	b2	c1	d1	f3
x7	a3	b1	c3	d2	f4
x8	a3	b1	c3	d2	f4

	b	e	g	h
y1	b1	e1	g1	h1
y2	b1	e1	g2	h2
y3	b1	e1	g1	h1
y4	b1	e1	g2	h2
y5	b2	e2	g2	h1
y6	b2	e2	g2	h1
y7	b2	e3	g3	h3
y8	b2	e3	g3	h3

Fig. 1. Process of resolving a query by QAS in DKS

"operational".

To present more general scenario, let us assume that query $q = q(A_1)$ is submitted to the information system $S = S(A)$ which is a part of DKS and $a_1 \in A_1 - A$. Since attribute a_1 is non-local for S, QAS for the system S sends a system query $[a_1(A)]_?$ to all remote sites $S(B)$ in DKS satisfying the property $a_1 \in B$ and $A \cap B \neq \emptyset$. System query $[a_1(A)]_?$ should be read "find description of attribute a_1 in terms of attributes from A at minimum one of the remote sites for $S(A)$". Now, assuming that $S(B)$ is such a remote site, all a_1-reducts in $S(B)$ are computed. For all a_1-reducts R included in A, our procedure stops and the

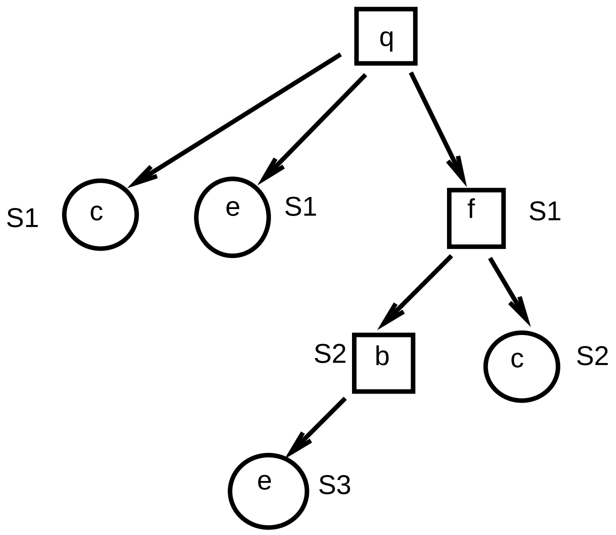


Fig. 2. Tree-resolution for q

system query $[a_1(A)]_?$ will be replaced by a term $[a_1(R)]_{S(B)}$. This term should be read "description of a_1 in terms of attributes from R can be discovered in $S(B)$". For any other a_1-reduct A_2 and $a_2 \in A_2 - A$, system $S(B)$ will send a system query $[a_1(A_2)]_{S(B)}, [a_2(A)]_?$ to all other remote sites. This system query should be read "find description of a_2 in terms of A which is needed to find description of a_1 in terms of A in $S(B)$. Assuming that $S(C)$ is one of the remote sites which can provide us with this kind of help, we compute all a_2-reducts in $S(C)$. For each a_2-reduct R included in A, our procedure stops and the system query $[a_1(A_2)]_{S(B)}, [a_2(A)]_?$ is replaced by a term $[a_1(A_2)]_{S(B)}, [a_2(R)]_{S(C)}$. For all other a_2-reducts A_3 and $a_3 \in A_3 - A$, system $S(C)$ sends a system query $[a_1(A_2)]_{S(B)}, [a_2(A_3)]_{S(C)}, [a_3(A)]_?$ to all other remote sites. This process will continue untill a reduct which is included in A is found. Such a system query is called successful. Clearly, each set in the sequence $A_i - A$, ...,$A_2 - A$, $A_1 - A$ may contain more than one attribute. This means that many system queries $[a_1(A_2)]_{S(B)}, [a_2(A_3)]_{S(C)}, [a_3(A)]_?$ can be formed. Some of them will be successful and some will fail. If one of the attributes is repeated in a system query, this query fails and the same can not be used in the process of constructing operational definition of a_1. Each system query which is successful can be used to construct a disjunct of the local term approximating query q.

4 Conclusion

Any non-local query, submitted to one of the sites of DKS, generates a collection of system queries represented by sequences of attributes. For instance, query

$[a_1(A_2)]_{S(B)}, [a_2(A_3)]_{S(C)}, [a_3(A)]_?$ is represented by a sequence of attributes (a_1, a_2, a_3). Only sequences with non-repeated attributes (no loops) are used in the final process of query resolution. All these sequence are used to find local approximation of the initial query. More sequences we have, more objects can be identified as the answer to the initial query. Also, it should be added that each such a sequence generates its own query processing steps based on its own rough-ontology (see [Ras/Dardzinska][6]).

References

1. Mizoguchi, R., "Ontological engineering: foundation of the next generation knowledge processing", in *Proceedings of Web Intelligence: Research and Development*, LNCS/LNAI, Springer-Verlag, No. 2198, 2001, 44-57
2. Pawlak, Z., "Rough classification", in *International Journal of Man-Machine Studies*, Vol. 20, 1984, 469-483
3. 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
4. Ras, Z., "Dictionaries in a distributed knowledge-based system", in *Concurrent Engineering: Research and Applications, Conference Proceedings*, Pittsburgh, Penn., Concurrent Technologies Corporation, 1994, 383-390
5. Ras, Z., "Query answering based on distributed knowledge mining", in *Intelligent Agent Technology, Research and Development*, Proceedings of IAT'01 (Eds. N. Zhong, J. Lin, S. Ohsuga, J. Bradshaw), World Scientific, 2001, 17-27
6. Ras, Z., Dardzinska, A., "Handling semantic inconsistencies in query answering based on distributed knowledge mining", in *Foundations of Intelligent Systems*, Proceedings of ISMIS'02 Symposium, Lyon, France, LNCS/LNAI, No. 2366, Springer-Verlag, 2002, 69-77
7. Ras, Z., Żytkow, J., "Mining for attribute definitions in a distributed two-layered DB system", *Journal of Intelligent Information Systems*, Kluwer, Vol. 14, No. 2/3, 2000, 115-130
8. Ras, Z., Żytkow, J.,"Discovery of equations to augment the shared operational semantics in distributed autonomous BD System", in *PAKDD'99 Proceedings*, LNCS/LNAI, No. 1574, Springer-Verlag, 1999, 453-463

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