Handling Semantic Inconsistencies in Distributed Knowledge Systems using Ontologies

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
15-351 Bialystok, Poland
ras@uncc.edu or agnadar@wp.pl

Abstract. Traditional query processing provides exact answers to queries. It usually requires that users fully understand the database structure and content to issue a query. Due to the complexity of the database applications, the so called global queries can be posed which traditional query answering systems can not handle. In this paper a query answering system based on distributed data mining is presented to rectify these problems. Task ontologies are used as a tool to handle semantic inconsistencies between sites.

1 Introduction

In many fields, such as medical, banking and educational, similar databases are kept at many sites. An attribute may be missing in one database, while it occurs in many others. Missing attributes lead to problems. A user may issue a query to a local database S_1 in search for objects in S_1 that 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 definition of a_1 may be extracted from databases at other sites and used to identify objects in S_1 having property a_1. The simplicity of this approach is no longer in place when the semantics of terms used to describe objects in a client and remote sites differ. Sometime, such a difference in semantics can be repaired quite easily. For instance if "Temperature in Celsius" is used at one site and "Temperature in Fahrenheit" at the other, a simple mapping will fix the problem. If databases are complete and two attributes have the same name and differ only in their granularity level, a new hierarchical attribute can be formed to fix the problem. If databases are incomplete, the problem is more complex because of the number of options available to interpret incomplete values (including null vales). The problem is especially difficult when rule-based chase techniques are used to replace null values by values which are less incomplete.

The notion of an intermediate model, proposed by [Maluf and Wiederhold] [1], is very useful to deal with heterogeneity problem, because it describes the

database content at a relatively high abstract level, sufficient to guarantee homogeneous representation of all databases. Knowledge bases built jointly with task ontologies proposed in this paper, can be used for a similar purpose. Knowledge bases contain rules extracted from databases at remote sites.

In this paper, the heterogeneity problem is introduced from the query answering point of view. Query answering system linked with a client site transforms, so called, global queries using definitions extracted at remote sites. These definitions may have so many different interpretations as the number of remote sites used to extract them. Task ontologies are used to find new interpretations representing consensus of all these sites.

2 Distributed Information Systems

In this section, we recall the notion of a distributed information system and a knowledge base for a client site formed from rules extracted at remote sites. We introduce the notion of local queries and give their standard semantics.

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 $a \in A$.

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

By *distributed information system* we mean $DS = (\{S_i\}_{i \in I}, L)$ 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:

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

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 knowledge base, 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 $(t_1 + t_2), (t_1 * t_2), \sim t_1 \in T_i$.

By a set of $s(i)$-formulas we mean a least set F_i such that:

- if $t_1, t_2 \in T_i$, then $(t_1 = t_2) \in F_i$.

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

We say that:

- $s(i)$-term t is *primitive* if it is of 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.

Clearly, it is easy to give an example of a local query. 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 defined by the standard interpretation M_i in a distributed information system $DS = (\{S_j\}_{j \in I}, L)$ 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
  $M_i(t_1 + t_2) = M_i(t_1) \cup M_i(t_2)$,
  $M_i(t_1 * t_2) = M_i(t_1) \cap M_i(t_2)$,
  $M_i(\sim t_1) = X_i - M_i(t_1)$.
  $M_i(t_1 = t_2) =$
  (if $M_i(t_1) = M_i(t_2)$ then T else F)
  where T stands for $True$ and F for $False$

The sound and complete axiomatization of the above semantics for a complete distributed information system is given, for instance, in paper by [Ras][5]. This semantics is slightly modified for distributed incomplete information systems (see paper by [Ras and Joshi][7]).

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

Let us assume that $r_1 = (c_1, t_1, s_1)$, $r_2 = (c_2, t_2, s_2)$ are (k,i)-rules. We say that: r_1, r_2 are strongly consistent, if either c_1, c_2 are values of two different attributes in S_k or a DNF form equivalent to $t_1 * t_2$ does not contain simple conjuncts.

Now, we are ready to define a knowledge base 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 have definition of $\sim c$ in D_{ki} if already definition of c is in D_{ki}. It will allow us to approximate (learn) concept c from both sites, if needed.

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

3 Semantic Inconsistencies and Distributed Knowledge Systems

In this section, we introduce the notion of a distributed knowledge system (DKS) and next we present problems related to its query answering system QAS. We discuss the process of handling semantic inconsistencies in knowledge extracted at different DKS sites and next we outline query transformation steps based on distributed knowledge mining.

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 knowledge base for $i \in I$.

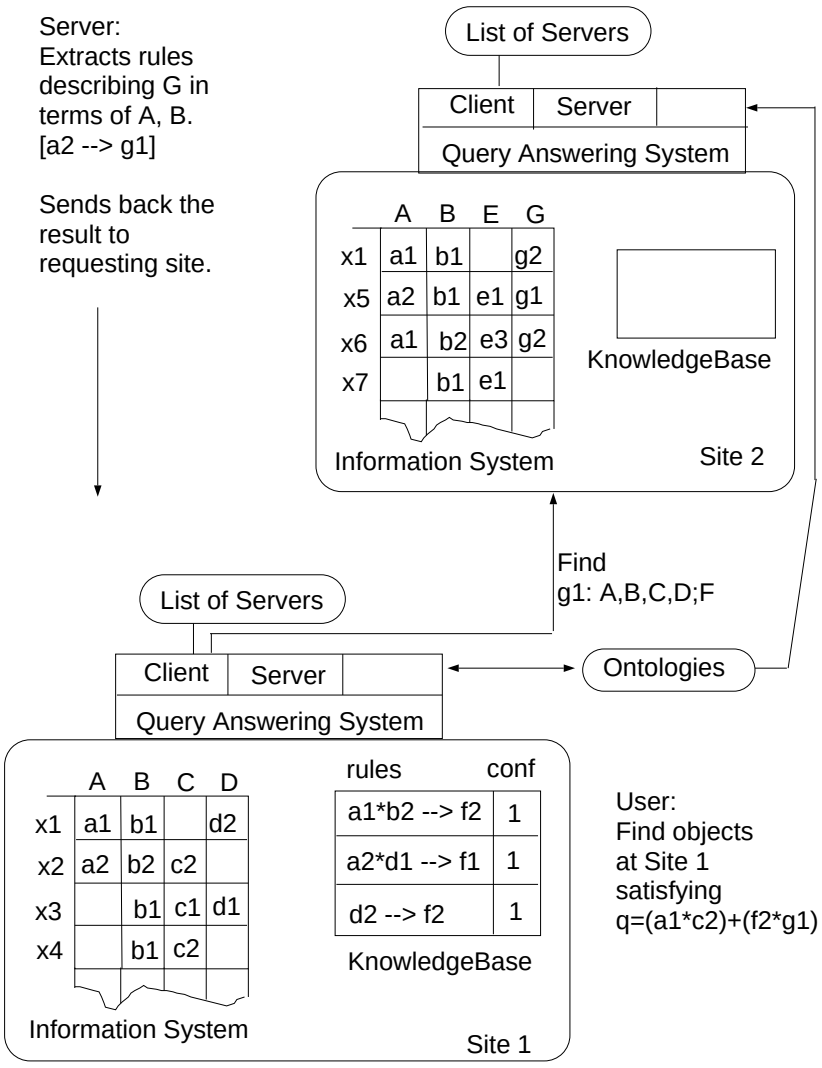

Fig. 1. Sample Model of DKS

Figure 1 shows an example of DKS and its query answering system QAS that handles global queries.

For simplicity reason only two sites of DKS are considered in our example. A user queries site 1 asking for all its objects satisfying $q = (a_1 * c_2) + (f_2 * g_1)$. The user is interested only in objects at site 1. The first part of the query, which is $(a_1 * c_2)$, can be handled by local QAS because both its attribute values are within the domain of the information system at site 1. For instance, taking optimistic interpretation of all attribute values at site 1 (null values are treated as supporting values), objects x_1 and x_2 will satisfy query $(a_1 * c_2)$. Let us consider the second part of the query q, which is $(f_2 * g_1)$. Attribute value f_2 is not in the domain V_1 of the information system S_1 at site 1 but its definition is in knowledge base D_1 at site 1. This definition can be used to replace the value f_2 in q by a term which defines f_2. In our example, either a partial definition $(f_2, a_1 * b_2 + d_2)$ or a definition $(f_2, a_1 * b_2 + d_2, \sim (a_1 * b_2 + d_2) * \sim d_2)$ of f_2 can be generated from D_1. The attribute value g_1 used in a query q is neither in the domain D_1 nor its definition is in D_1. In this case we have to search for a remote site, where definition of g_1 can be extracted from its information system. In our example site 2 satisfies this requirement. Expression (g_1, a_2) can be seen as a partial definition of g_1 which can be extracted from S_2. Alternatively, expression $(g_1, a_2, \sim a_1 * \sim a_2)$ can be used as a definition of g_1 found at site 2. To simplify the problem further, assume that partial definitions of f_2 and g_1 are used to replace query q by a new query which can be handled locally by QAS at site 1. This new query approximating q, described by a term $(a_1 + c_2) + (a_1 * b_2 + d_2) * a_2$, can be seen as a lower approximation of q in rough sets terminology. If we use definition of f_2 and definition of g_1 instead of partial definitions, query q can be replaced by a rough query. Rough queries are especially useful when the boundary area in a rough query representation is small. In a distributed scenario, similar to the one presented in this paper, this boundery area is getting smaller and smaller when more and more sites are used to search for definitions of non-local attributes (f_2 and g_1 in our example). Now, let's go back to term $(a_1 * c_2) + (a_1 * b_2 + d_2) * a_2$. If the distribution law can be applied then our term would be transformed to $(a_1 * c_2) + (a_1 * b_2 * a_2 + d_2 * a_2)$ and next assuming that properties $a_1 * a_2 = 0$ and $0 * t = 0$ hold we would get its final equivalent form which is $a_1 * c_2 + d_2 * a_2$. However if each of our three terms $a_1 * b_2$, d_2, a_2 is computed under three different semantics, then there is a problem with the above tranformation process and with a final meaning of $(a_1 * b_2 * a_2 + d_2 * a_2)$.

For instance, let us assume a scenario where partial definition $(f_2, a_1 * b_2)$ was extracted under semantics M_1, partial definition (f_2, d_2) under semantics M_2, and (g_1, a_2) under semantics M_3. Null value is interpreted below as a set of all possible values for a given attribute. Also, x has the same meaning as the pair $(x, 1)$.

Semantics M_1 (see [Ras and Joshi][7]) is defined as:

- $M_1(v) = \{(x, k) : v \in a(x)\ \&\ k = card(a(x))\}$ if $v \in V_a$,
- $M_1(t_1 * t_2) = M_1(t_1) \otimes M_1(t_2)$, $M_1(t_1 + t_2) = M_1(t_1) \oplus M_1(t_2)$.

To define $\otimes$, $\oplus$, let us assume that $P_i = \{(x, p_{<x,i>}) : p_{<x,i>} \in [0,1]$ & $x \in X\}$ where X is a set of objects. Then, we have:

- $P_i \otimes P_j = \{(x, p_{<x,i>} \cdot p_{<x,j>})$: $x \in X\}$,
- $P_i \oplus P_j = \{(x, max(p_{<x,i>}, p_{<x,j>}))$: $x \in X\}$.

Semantics M_2 is defined as:

- $M_2(v) = \{x : v \in a(x)$ & $card(a(x)) = 1\}$ if $v \in V_a$,
- $M_2(t_1 * t_2) = M_2(t_1) \cap M_2(t_2)$, $M_2(t_1 + t_2) = M_2(t_1) \cup M_2(t_2)$.

Finally, semantics M_3 is defined as:

- $M_3(v) = \{x : v \in a(x)\}$ if $v \in V_a$,
- $M_3(t_1 * t_2) = M_3(t_1) \cap M_3(t_2)$, $M_3(t_1 + t_2) = M_3(t_1) \cup M_3(t_2)$.

Assume now, the following relationship between semantics M_i and M_j:
$M_i \preceq M_j$ iff $[\ (x,k) \in M_i(a) \longrightarrow (\exists k_1 \geq k)[(x,k_1) \in M_j(a)]\ ]$.

It can be easily proved that $\preceq$ is a partial order relation.

In our example, for any $v \in V_a$, we have: $M_2(v) \preceq M_1(v) \preceq M_3(v)$.

We say that functors $+$ and $*$ preserve monotonicity property for semantics N_1, N_2 if the following two conditions hold:

- $[N_1(t_1) \preceq N_2(t_1)$ & $N_1(t_2) \preceq N_2(t_2)]$ implies $N_1(t_1 + t_2) \preceq N_2(t_1 + t_2)$,
- $[N_1(t_1) \preceq N_2(t_1)$ & $N_1(t_2) \preceq N_2(t_2)]$ implies $N_1(t_1 * t_2) \preceq N_2(t_1 * t_2)$.

Let $(\Omega, \preceq)$ be a partially ordered set of semantics. We say that it preserves monotonicity property for $+$ and $*$, if $+$ and $*$ preserve monotonicity property for any N_1, N_2 in Ω. It can be easily checked that $(\{M_1, M_2, M_3\}, \preceq)$ preserves monotonicity property for $+$ and $*$.

We adopt the definition of ontology proposed by Mizoguchi [2]. He claims that ontology should consist of *task* ontology which characterizes the computational architecture of a (distributed) knowledge system which performs a task and *domain* ontology which characterizes the domain knowledge where the task is performed.

In the scenario presented in our paper, a number of remote sites for a given client site has to be accessed. The same terms, used in knowledge extraction or local query processing, can have different interpretations at each of these sites. In a query transformation process, many subterms forming any of these intermidiate queries may come from definitions extracted not necessarily from the same site of DKS. Clearly, in this situation our query can not be processed unless a common, possibly optimal, semantics for all these subterms is found.

We claim that one way to solve this problem is to assume that partially ordered set of semantics $(\Omega, \preceq)$, preserving monotonicity property for $+$ and $*$, is a part of global ontology (or task ontology in Mizoguchi's [2] definition).

In our example, we evaluate query q taking first M_2 and next M_3 as two common semantics for all subterms obtained during the transformation process

of q because $M_2(v) \preceq M_1(v) \preceq M_3(v)$. In general, if $(\Omega, \preceq)$ is a lattice and $\{M_i\}_{i \in I}$ is a set of semantics involved in transforming query q, then we should take $M_{min} = \bigcap\{M_i\}_{i \in I}$ (the greatest lower bound) and $M_{max} = \bigcup\{M_i\}_{i \in I}$ (the least upper bound) as two common semantics for processing query q.

Common semantics is also needed because of the necessity to prove soundness and possibly completeness of axioms to be used in a query tranformation process.

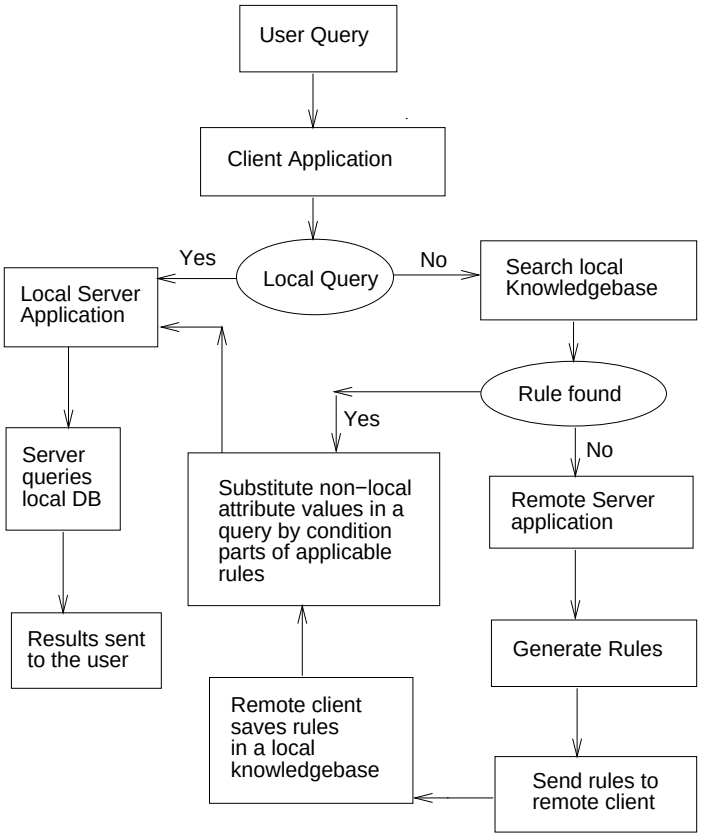

Fig. 2. Query and Rough Query Processing by DKS

Figure 2 gives a flowchart of a query transformation process in QAS assuming that local query semantics at all contacted sites is the same and DKS is consistent (granularity levels of the same attributes at remote sistes and the client site are the same). This flowchart will be replaced by two similar flowcharts (corresponding to M_{min} and M_{max}) if semantics at contacted remote sites and a client site differ. Semantics M_{min} and M_{max} can be seen jointly as a rough semantics.

The word *rough* is used here in the sense of rough sets theory (see [Pawlak][3]). Saying another words, the semantics used during query transformation process can be seen only as one which is between two semantics M_{min} and M_{max}.

If we increase the number of sites from which definitions of non-local attributes are collected and then resolve inconsistencies among them (see [Ras][4]), the local confidence in resulting definitions is expected to be higher since they represent consensus of more sites. At the same time, if the number of remote sites involved in a query transformation process is increased, the number of different semantics may increase as well which in result may also increase the roughness of the answer to the query.

4 Conclusion

Clearly, the easiest way to solve semantic inconsistencies problem is to apply the same semantics at all remote sites. However when databases are incomplete and we replace their null values using rule-based chase algorithms based on rules locally extracted then we are already commited to the semantics used by these algorithms. If we do not keep track what and how null values have been replaced by rule-based chase algorithms, there is no way back for us. Also, it sounds rather unrealistic that one semantics for incomplete databases can be chosen as a standard. We claim that in such cases a partially ordered set of semantics $(\Omega, \preceq)$ or its equivalent structure should be a part of *task* ontologies to solve the problem.

References

1. 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
2. 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
3. Pawlak, Z., "Rough classification", in *International Journal of Man-Machine Studies*, Vol. 20, 1984, 469-483
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., "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
6. 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
7. Ras, Z., Joshi, S.,"Query approximate answering system for an incomplete DKBS", in *Fundamenta Informaticae*, IOS Press, Vol. 30, No. 3/4, 1997, 313-324