Query Answering based on Collaboration and Chase

Zbigniew W. Raś[1,2] and Agnieszka Dardzińska[3,1]

[1] UNC-Charlotte, Department of Computer Science, Charlotte, N.C. 28223, USA
[2] Polish Academy of Sciences, Institute of Computer Science, Ordona 21, 01-237 Warsaw, Poland
[3] Bialystok Technical Univ., Dept. of Mathematics, ul. Wiejska 45A, 15-351 Bialystok, Poland

Abstract. Query Answering System (QAS) for a Distributed Information System (DIS) which is based on collaboration and $Chase$ is presented in this paper. Collaboration among systems is driven by the QAS requests for knowledge needed to resolve, so called, non-local queries at a client site. As the result of these requests, knowledge in the form of definitions of incomplete or locally foreign attribute values is extracted at a number of sites in DIS and next exchanged between all of them. The final outcome of this step is a new knowledge-base KB created at the client site. Now, the chase process will begin: each incomplete slot in a column corresponding to attribute listed in a query, let us say d, is chased with respect to rules from KB describing d. All other incomplete attributes listed in a query are processed the same way. When the chase process ends, the query is handled by QAS in a standard way.

1 Introduction

Distributed information system is a system that connects a number of information systems using network communication technology. In this paper, we assume that these systems are autonomous and incomplete. Incompleteness is understood by allowing to have a set of weighted attribute values as a value of an attribute. Additionally, we assume that the sum of these weights has to be equal 1. The definition of an information system of type λ and Distributed Information System (DIS) proposed in this paper is a modification of definitions given by Raś in [13] and used later by Raś and Dardzińska in [14] to talk about semantic inconsistencies among sites of DIS from the query answering point of view. The type λ is introduced mainly to monitor the weights assigned to values of attributes by $Chase$ algorithm, if they are greater than or equal to λ. If the weight assigned by $Chase$ to one of the attribute values is less than the allowed threshold value, then this attribute value has to be ruled out. Semantic inconsistencies are due to different interpretations of attributes and their values among sites (for instance one site can interpret the concept *young* differently than other sites). Different interpretations are also due to the way each site is handling null values. Null value replacement by a value suggested either by statistical or some rule-based methods is quite common before a query is answered by QAS. Ontologies ([9], [10], [16], [17], [18], [2], [3], [19], [7]) are widely used as a sort of semantical bridge between information systems built independently so they can collaborate and understand each other. In [14], the notion of the optimal rough semantics and a method of its construction was proposed. The rough semantics can be used to model and nicely handle semantic inconsistencies among sites due to different interpretations of incomplete values. Distributed chase is a chase algorithm [1] linked with a site S of DIS, called a client, which is similar to $Chase1$ [6] and $Chase2$ [4] with additional assumption concerning the creation of knowledge bases at all sites of DIS involved in the process of solving a query submitted to S. The knowledge base at the client site contains rules extracted from S and also rules extracted from information systems at its remote sites. The structure of the knowledge base and its properties are the same as properties of the knowledge bases used in $Chase1$ or $Chase2$ algorithms. The difference only lies in the process required to collect these rules. In a distributed framework, these rules are extracted from the local and remote sites, usually under different semantics. Although the names of the attributes are often the same among sites, their granularity levels may differ from site to site. As the result of these differences, the knowledge base has to satisfy certain properties in order to be used by $Chase$. The same properties are required by the query answering system based on $Chase$ and will be given in this paper.

2 Query Processing with Incomplete Data

In real life, data are often collected and stored in information systems residing at many different locations, built independently, instead of collecting them and storing at only one single location. In this case we talk about distributed (autonomous) information systems. It is very possible that an attribute is missing in one of them while it occurs in many others. Also, in one information system, an attribute might be strongly incomplete, while in other systems the same attribute is either complete or close to being complete. Saying that attribute is strongly incomplete, we mean that the value of this attribute is either unknown or only partially known for most of the objects. Assume that user submits a query to one of the information systems (called a client), which rather would prefer not to answer it (we assume that the query answering system for the client site is possessing some intelligence) due to the fact that some of the attributes used in a query are missing or strongly incomplete at the client site. In such a case, network communication technology is used to get definitions of these unknown or strongly incomplete attributes from other information systems (called servers). All these new definitions form a knowledge base which can be used to chase that missing or strongly incomplete attributes at the client site. But, before any chase algorithm, called *rule-based chase*, can be applied, semantic inconsistencies among sites have to be somehow resolved. For instance, it can be done by taking rough semantics [Raś & Dardzińska], mentioned earlier.

Definition 1:
We say that $S = (X, A, V)$ is a partially incomplete information system of type λ, if S is an incomplete information system and the following three conditions hold:

- $a_S(x)$ is defined for any $x \in X$, $a \in A$,
- $(\forall x \in X)(\forall a \in A)[(a_S(x) = \{(a_i, p_i) : 1 \leq i \leq m\}) \rightarrow \sum_{i=1}^{m} p_i = 1]$,
- $(\forall x \in X)(\forall a \in A)[(a_S(x) = \{(a_i, p_i) : 1 \leq i \leq m\}) \rightarrow (\forall i)(p_i \geq \lambda)]$.

Now, let us assume that S_1, S_2 are partially incomplete information systems, both of type λ. The same set X of objects is stored in both systems and the same set A of attributes is used to describe them. The meaning and granularity of values of attributes from A in both systems S_1, S_2 is also the same. Additionally, we assume that $a_{S_1}(x) = \{(a_{1i}, p_{1i}) : 1 \leq m_1\}$ and $a_{S_2}(x) = \{(a_{2i}, p_{2i}) : 1 \leq m_2\}$.

We say that containment relation Ψ holds between S_1 and S_2, if the following two conditions hold:

- $(\forall x \in X)(\forall a \in A)[card(a_{S_1(x)}) \geq card(a_{S_2(x)})]$,
- $(\forall x \in X)(\forall a \in A)[[card(a_{S_1}(x)) = card(a_{S_2}(x))] \rightarrow$
  $[\sum_{i \neq j} |p_{2i} - p_{2j}| > \sum_{i \neq j} |p_{1i} - p_{1j}|]]$.

Instead of saying that containment relation holds between S_1 and S_2, we can equivalently say that S_1 was transformed into S_2 by containment mapping Ψ. This fact can be presented as a statement $\Psi(S_1) = S_2$ or $(\forall x \in X)(\forall a \in A)[\Psi(a_{S_1}(x)) = \Psi(a_{S_2}(x))]$. Similarly, we can either say that $a_{S_1}(x)$ was transformed into $a_{S_2}(x)$ by Ψ or that containment relation Ψ holds between $a_{S_1}(x)$ and $a_{S_2}(x)$.

So, if containment mapping Ψ converts an information system S to S', then S' is more complete than S. Saying another words, for a minimum one pair $(a, x) \in A \times X$, either Ψ has to decrease the number of attribute values in $a_S(x)$ or the average difference between confidences assigned to attribute values in $a_S(x)$ has to be increased by Ψ.

To give an example of a containment mapping Ψ, let us take two information systems S_1, S_2 both of the type λ, represented as Table 2 and Table 2.

It can be easily checked that the values assigned to $e(x_1)$, $b(x_2)$, $c(x_2)$, $a(x_3)$, $e(x_4)$, $a(x_5)$, $c(x_7)$, and $a(x_8)$ in S_1 are different than the corresponding values in S_2. In each of these eight cases, an attribute value assigned to an object in S_2 is less general than the value assigned to the same object in S_1. It means that $\Psi(S_1) = S_2$.

X	a	b	c	d	e
x_1	$\{(a_1,\frac{1}{3}),(a_2,\frac{2}{3})\}$	$\{(b_1,\frac{2}{3}),(b_2,\frac{1}{3})\}$	c_1	d_1	$\{(e_1,\frac{1}{2}),(e_2,\frac{1}{2})\}$
x_2	$\{(a_2,\frac{1}{4}),(a_3,\frac{3}{4})\}$	$\{(b_1,\frac{1}{3}),(b_2,\frac{2}{3})\}$		d_2	e_1
x_3		b_2	$\{(c_1,\frac{1}{2}),(c_3,\frac{1}{2})\}$	d_2	e_3
x_4	a_3		c_2	d_1	$\{(e_1,\frac{2}{3}),(e_2,\frac{1}{3})\}$
x_5	$\{(a_1,\frac{2}{3}),(a_2,\frac{1}{3})\}$	b_1	c_2		e_1
x_6	a_2	b_2	c_3	d_2	$\{(e_2,\frac{1}{3}),(e_3,\frac{2}{3})\}$
x_7	a_2	$\{(b_1,\frac{1}{4}),(b_2,\frac{3}{4})\}$	$\{(c_1,\frac{1}{3}),(c_2,\frac{2}{3})\}$	d_2	e_2
x_8		b_2	c_1	d_1	e_3

Table 1. Information System S_1

X	a	b	c	d	e
x_1	$\{(a_1,\frac{1}{3}),(a_2,\frac{2}{3})\}$	$\{(b_1,\frac{2}{3}),(b_2,\frac{1}{3})\}$	c_1	d_1	$\{(e_1,\frac{1}{3}),(e_2,\frac{2}{3})\}$
x_2	$\{(a_2,\frac{1}{4}),(a_3,\frac{3}{4})\}$	b_1	$\{(c_1,\frac{1}{3}),(c_2,\frac{2}{3})\}$	d_2	e_1
x_3	a_1	b_2	$\{(c_1,\frac{1}{2}),(c_3,\frac{1}{2})\}$	d_2	e_3
x_4	a_3		c_2	d_1	e_2
x_5	$\{(a_1,\frac{3}{4}),(a_2,\frac{1}{4})\}$	b_1	c_2		e_1
x_6	a_2	b_2	c_3	d_2	$\{(e_2,\frac{1}{3}),(e_3,\frac{2}{3})\}$
x_7	a_2	$\{(b_1,\frac{1}{4}),(b_2,\frac{3}{4})\}$	c_1	d_2	e_2
x_8	$\{(a_1,\frac{2}{3}),(a_2,\frac{1}{3})\}$	b_2	c_1	d_1	e_3

Table 2. Information System S_2

3 Query Processing with Distributed Data and Chase

Assume now that $L(D) = \{(t \to v_c) \in D : c \in In(A)\}$ (called a knowledge-base) is a set of all rules extracted initially from $S = (X, A, V)$ by $ERID(S, \lambda_1, \lambda_2)$, where $In(A)$ is the set of incomplete attributes in S and λ_1, λ_2 are thresholds for minimum support and minimum confidence, correspondingly. *ERID* is the algorithm for discovering rules from incomplete information systems, presented by Dardzińska and Raś in [5]. The type of incompleteness in [5] is the same as in this paper but we did not provide there a threshold value λ for the minimal confidence of attribute values assigned to objects. The algorithm *ERID* works the same way for incomplete information systems of type λ, since the knowledge discovery process in *ERID* is independent from the largeness of parameter λ. Assume now that a query $q(B)$ is submitted to system $S = (X, A, V)$, where B is the set of all attributes used in $q(B)$ and that $A \cap B \neq \emptyset$. All attributes in $B - [A \cap B]$ are called foreign for S. If S is a part of a distributed information system, definitions of foreign attributes for S can be extracted at its remote sites (see [14]). Clearly, all semantic inconsistencies and differences in granularity of attribute values among sites have to be resolved first. To simplify the problem, we take the same assumption as in [14]. It mean that we only allow different granularity of attribute values and different semantics related to different interpretations of incomplete attribute values among sites. In [14], it was shown that we can process a query of the type $q(B)$ at site S by discovering definitions of values of attributes from $B - [A \cap B]$ at remote sites for S and use them to answer $q(B)$.

Foreign attributes for S, can be seen as attributes entirely incomplete in S, which means values (either exact or partially incomplete) of such attributes have to be ascribed to all objects in S. Stronger the consensus among sites on a value to be ascribed to x, *better* the result of the ascription process for x can be expected. Assuming that systems S_1, S_2 are storing the same sets of objects and using the same attributes to describe them, system S_1 is *better* than system S_2, if $\Psi(S_2) = S_1$. So, clearly *the best* information system will be the one which is complete. The question remains, if the values predicted by the imputation process are really correct, and if not, how far they are (assuming that some distance measure can be set up) from the correct values which clearly are unknown? Classical approach, to this kind of problems, is to start with a complete information system and remove randomly from it, let's say, 10 percent of its values and next run the imputation algorithm on the resulting system. The next step is to compare the descriptions of objects in the system which is the outcome of the imputation algorithm with descriptions of the same objects in the original system. But, before we can continue any further this discussion, we have to decide first on the interpretation of functors *or* and *and*, denoted in this paper by $+$ and $*$, correspondingly. We will adopt the semantics of terms proposed by Raś & Joshi in [15] as their semantics has all the properties required for the query transformation process to be semantically correct [see [15]]. It was proved that, under their semantics, the following distributive property holds: $t_1 * (t_2 + t_3) = (t_1 * t_2) + (t_1 * t_3)$.

So, let us assume that $S = (X, A, V)$ is an information system of type λ and t is a term constructed in a standard way (for predicate calculus expression) from values of attributes in V seen as *constants* and from two functors $+$ and $*$. By $N_S(t)$, we mean the standard interpretation of a term t in S defined as (see [15]):

- $N_S(v) = \{(x, p) : (v, p) \in a(x)\}$, for any $v \in V_a$,
- $N_S(t_1 + t_2) = N_S(t_1) \oplus N_S(t_2)$,
- $N_S(t_1 * t_2) = N_S(t_1) \otimes N_S(t_2)$,

where, for any $N_S(t_1) = \{(x_i, p_i)\}_{i \in I}$, $N_S(t_2) = \{(x_j, q_j)\}_{j \in J}$, we have:

- $N_S(t_1) \oplus N_S(t_2) =$
  $\{(x_i, p_i)\}_{i \in (I-J)} \cup \{(x_j, p_j)\}_{j \in (J-I)} \cup \{(x_i, max(p_i, q_i))\}_{i \in I \cap J}$,
- $N_S(t_1) \otimes N_S(t_2) = \{(x_i, p_i \cdot q_i)\}_{i \in (I \cap J)}$.

The incomplete value imputation algorithm *Chase*, given below, converts information system S of type λ to a new more complete information system $Chase(S)$ of the same type. Our algorithm is entirely new in comparison to known strategies for chasing incomplete data in relational tables because of the assumption about partial incompleteness of data (sets of weighted attribute values can be assigned to an object as its value). This forced us to develop a new discovery algorithm, called *ERID*, which can extract rules from this type of incomplete data (see [5]) so it can be used in the *Chase* algorithm given below.

```
Algorithm Chase($S, In(A), L(D)$);
  Input   System $S = (X, A, V)$,
          set of incomplete attributes $In(A) = \{a_1, a_2, ..., a_k\}$ in $S$,
          set of rules $L(D)$ discovered from $S$ by $ERID$.
  Output  System $Chase(S)$
   begin
   S':= S;
   $j := 1$;
   while $j \leq k$ do
   begin
   $S_j := S$;
   for all $x \in X$ do
      $q_j := 0$;
      begin
      $b_j(x) := \emptyset$;
      $n_j := 0$;
      for all $v \in V_{a_j}$
      begin
      if $card(a_j(x)) \neq 1$ and $\{(t_i \rightarrow v) : i \in I\}$
         is a maximal subset of rules from $L(D)$
```

```
        such that $(x, p_i) \in N_{S_j}(t_i)$ **then**
        **if** $\sum_{i \in I}[p_i \cdot conf(t_i \to v) \cdot sup(t_i \to v)] \geq \lambda$ **then**
        **begin**
        $b_j(x) := b_j(x) \cup \{(v, \sum_{i \in I}[p_i \cdot conf(t_i \to v) \cdot sup(t_i \to v)])\};$
        $n_j := n_j + \sum_{i \in I}[p_i \cdot conf(t_i \to v) \cdot sup(t_i \to v)]$
        **end**
      **end**
      $q_j := q_j + n_j;$
      **end**
  **if** $\Psi(a_j(x)) = [b_j(x)/q_j]$ **then** $a_j(x) := [b_j(x)/q_j];$
  $j := j + 1;$
  **end**
  $S := \bigcap\{S_j : 1 \leq j \leq k\};$ /definition of $\bigcap\{S_j : 1 \leq j \leq k\}$ is given below/
  **if** $S \neq S'$ **then** $Chase(S, In(A), L(D))$
  **end**
```

Information system $S = \bigcap\{S_j : 1 \leq j \leq k\}$ is defined as:

$a_S(x)$=$\{a_{S_j}(x)$: *if* $a = a_j$ for any $j \in \{1, 2, ..., k\}\}$
for any attribute a and object x.

Still, one more definition is needed to complete the presentation of the algorithm *Chase*. Namely, we say that:

$[b_j(x)/p] = \{(v_i, p_i/p)\}_{i \in I}$, if $b_j(x) = \{(v_i, p_i)\}_{i \in I}$.

Algorithm *Chase* converts any incomplete or partially incomplete information system S to a new information system which is more complete. At each recursive call of *Chase*, its input data including S, $L(D)$, and from time to time $In(A)$ are changing. So, before any recursive call is executed, these new data have to be computed first.

Now, we give the time complexity $(T - Comp)$ of the algorithm *Chase*. Assume first that $S = S(0) = (X, A, V)$, $card(In(A)) = k$, and $n = card(X)$. We also assume that $S(i) = Chase^i(S)$ and
$n(i) = card\{x \in X : (\exists a \in A)[a_{S(i)}(x) \neq 1]\}$, both for $i \geq 0$.

Clearly, $n(0) > n(1) > n(2) > ... > n(p) = n(p+1)$, because information system $Chase^{i+1}(S)$ is more complete than information system $Chase^i(S)$, for any $i \geq 0$.

$T - Comp(Chase) = \bigcirc[\sum_{i=0}^{p}[k \cdot [n + n(i) \cdot card(L(D)) \cdot n] + n(i)]] =$
$\bigcirc[\sum_{i=0}^{p}[k \cdot [n(i) \cdot card(L(D)) \cdot n]]] = \bigcirc[k^2 \cdot n^3 \cdot m]$.

The final worst case complexity of *Chase* is based on the observation that p can not be larger than $k \cdot n$. We also assume here that $m = card(L(D))$.

Explanation. Algorithm *Chase* is called recursively p times (this is why we have $\sum_{i=0}^{p}$). Each recursive call has to improve minimum one slot in S, otherwise algorithm will stop. How large is p depends on the additional threshold for *Chase* which defines the acceptable improvement for a slot in S so the algorithm does not stop. For instance, if we set up the threshold in a way that maximum 10 improvements per slot in S can take place, then $10 \cdot In(S)$ is the worst value for p (where $In(S)$ is the number of incomplete slots in S). In practice, the value p is usually low. Now, for each incomplete attribute in $In(A)$, where $card[In(A)] = k$, we identify which slots have incomplete values (the total number of incomplete slots in $Chase^i(S)$ is equal to $n(i)$). It takes $n \cdot k$ steps to do that. For each row in S, which has an incomplete slot, we identify all rules in $L(D)$ supported by that row. It takes $n(i) \cdot card(L(D))$ steps to do that. For all these rules, we calculate their confidence and support. So, the worst number of steps needed to achieve that in i-iteration of $Chase$ will be $n(i) \cdot card(L(D)) \cdot n$.

Now, let us assume that the client site is represented by a partially incomplete information system S of type λ. When a query is submitted to S, its query answering system QAS will replace S by $Chase(S)$ and next will solve the query using, for instance, the strategy proposed Raś & Joshi in [15]. Clearly, we can argue here why the resulting information system obtained by $Chase$ can not be stored aside and reused when a new query is submitted to S? If S does not have many updates, we can do that by keeping a copy of $Chase(S)$ and next reuse that copy when a new query is submitted to S. The original information system S has to be kept so when user wants to enter new data, they can be stored in the original system. System $Chase(S)$, if stored aside, can not be reused by QAS when the number of updates in the original S exceeds a given threshold value. It means that the newest information system S has to be chased by our $Chase$ algorithm before any query is answered by QAS.

4 Distribution, Inconsistency, and Distributed Chase

If rules in $L(D)$, called the knowledge base and used by algorithm $Chase$, are extracted from many information systems in DIS, including the client S, the possibility to have the data updated in on of them is greatly increasing. This means that the possibility of reusing $Chase(S)$ by QAS for a longer time is automatically decreasing.

As we already pointed out, the knowledge base $L(D)$, contains rules extracted locally at the client site (information system queried by user) as well as rules extracted from information systems at its remote sites. Since rules are extracted from different information systems, inconsistencies in semantics, if any, have to be resolved before any query can be processed. There are two options:

– a knowledge base $L(D)$ at the client site is kept consistent (in this scenario all inconsistencies have to be resolved before rules are stored in the knowledge base),

– a knowledge base at the client site is inconsistent (values of the same attribute used in two rules extracted at different sites may be of different granularity level and may have different semantics associated with them).

In general, we assume that the information stored in a local system ontology and in a global ontology (if they are provided) is sufficient to resolve inconsistencies in semantics of all sites involved in *Chase*. Inconsistencies related to the confidence of conflicting rules stored in $L(D)$ do not have to be resolved at all (algorithm *Chase* does not have such a requirement).

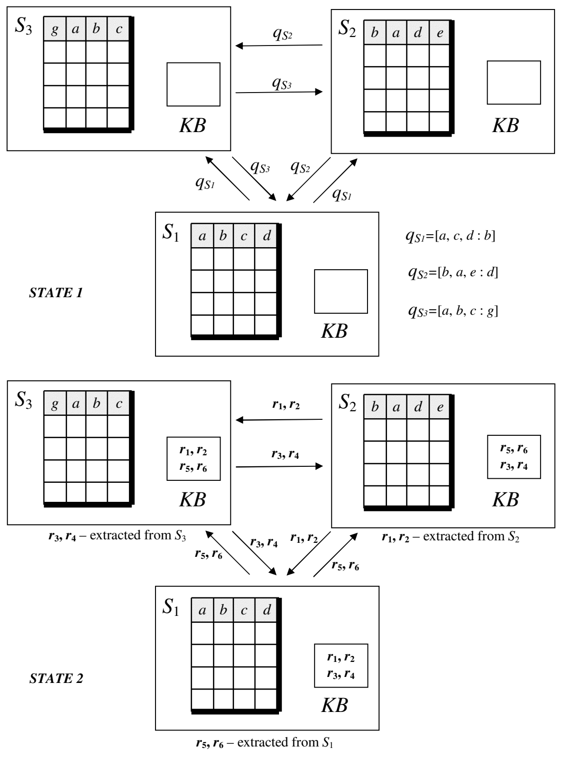

Fig. 1. Global extraction and exchange of knowledge

The fact that rules stored in $L(D)$ are often extracted at different sites and under different interpretations of incomplete values is not pleasant, if we want to use them in *Chase*. As we have already shown in [14], the rough semantics (see [14]) is very useful to handle inconsistencies related to different interpretations of incomplete values. Following the same approach as in [14], we can apply rough semantics to rules in $L(D)$ when information system S has to be chased by $Chase$ algorithm.

One of the problems related to an incomplete information system $S = (X, A, V)$ is a freedom how new values are constructed to replace incomplete values in S, before any rule extraction process begins. This replacement of incomplete attribute values in some of the slots in S can be done either by $Chase$ or/and by a number of available statistical methods (see [8]). This implies that semantics of queries submitted to S and driven (defined) by the query answering system QAS based on $Chase$, will often differ. Following the approach in [14], rough semantics can be also used by QAS to handle this problem.

In this paper we only concentrate on granularity-based semantic inconsistencies. Assume first that $S_i = (X_i, A_i, V_i)$ is an information system for any $i \in I$ and that all of them form a Distributed Information System (DIS). Additionally, we assume that, if $a \in A_i \cap A_j$, then only the granularity levels of a in S_i and S_j may differ but conceptually its meaning, both in S_i and S_j is the same. Assume now that $L(D_i)$ is a set of rules extracted from S_i, which means that $D = \bigcup_{i \in I} L(D_i)$ is a set of rules which can be used by $Chase$ algorithm associated with any of the sites of DIS. Now, let us say that system S_k, $k \in I$ is queried be a user. $Chase$ algorithm, to be applicable to S_k, has to be based on rules from D which satisfy the following conditions:

- attribute value used in the decision part of a rule from D has the granularity level either equal to or finer than the granularity level of the corresponding attribute in S_k.
- the granularity level of any attribute used in the classification part of a rule from D is either equal or softer than the granularity level of the corresponding attribute in S_k.
- attribute used in the decision part of a rule from D either does not belong to A_k or is incomplete in S_k.

These three conditions are called distributed $Chase$ S_k-applicability conditions. Let $L_k(D)$ denotes the subset of rules in D satisfying these three $Chase$ S_k-applicability conditions.

Assuming now that a match between the attribute value used in the description of the tuple t and the attribute value used in a description of a rule $s \rightarrow d \in L_k(D)$ is found, the following two cases should be considered:

- an attribute involved in matching is the decision attribute in $s \rightarrow d$. If two attribute values, involved in that match, have different granularity, then the decision value d has to be replaced by a softer value which granularity will match the granularity of the corresponding attribute in S_k.
- an attribute a involved in matching is the classification attribute in $s \rightarrow d$. If two attribute values, involved in that match, have different granularity, then the value of attribute a has to be replaced by a finer value which granularity will match the granularity of a in S_k.

The new set of rules, constructed from $L_k(D)$ following the above two steps, is called granularity-repaired set of rules. So, the assumption that $L_k(D)$ satisfies distributed Chase S_k-applicability conditions is sufficient to run *Chase* successfully on S_k using this new granularity-repaired set of rules.

In Figure 1, we present two consecutive states of a distributed information system consisting of S_1, S_2, S_3.

In the first state, all incomplete attributes in all three information systems have to be identified. System S_1 sends request q_{S_1} to the other two information systems asking them for definitions of its incomplete attributes. Similarly, system S_2 sends request q_{S_2} to the other two information systems asking them for definitions of its incomplete attributes. Finally, system S_3 sends request q_{S_3} to the other two information systems asking them also for definitions of its incomplete attributes. Next, rules describing the requested definitions are extracted from each of these three information systems and sent to the systems which requested them. So, the set $L(D_1)$ is sent to S_2 and S_3, the set $L(D_2)$ is sent to S_1 and S_3, and the set $L(D_3)$ is sent to S_1 and S_2.

The second state of a distributed information system, presented in Figure 1, shows all three information systems with the corresponding $L(D_i)$ sets, $i \in \{1, 2, 3\}$, all abbreviated as KB. Now, the *Chase* algorithm is run independently at each of our three sites. Resulting information systems are: $Chase(S_1)$, $Chase(S_2)$, and $Chase(S_3)$. Now, the whole process is recursively repeated. It means, all incomplete attributes in all three new information systems are identified again. Next, each of these three systems is sending requests to the other two systems asking for definitions of its incomplete attributes and when these definitions are received, they are stored in the corresponding KB sets. Now, *Chase* algorithm is run again at each of these three sites. The whole process is repeated till some fixed point is reached (all 3 systems have to become stable). When this step is accomplished, a query submitted to any S_i, $i \in \{1, 2, 3\}$, can be processed in a standard way.

We believe, that our query processing strategy for incomplete information systems guarantees the most accurate results because the incomplete values at all information systems have been replaced by new values representing the consensus of all sites. Our initial testing results for QAS based on distributed *Chase* are quite promising.

References

1. Atzeni, P., DeAntonellis, V. (1992) Relational database theory, The Benjamin Cummings Publishing Company
2. Benjamins, V. R., Fensel, D., Prez, A. G. (1998) Knowledge management through ontologies, in *Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM-98)*, Basel, Switzerland.
3. Chandrasekaran, B., Josephson, J. R., Benjamins, V. R. (1998) The ontology of tasks and methods, in *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada

4. Dardzińska, A., Raś, Z.W. (2003) Rule-Based Chase Algorithm for Partially Incomplete Information Systems, in **Proceedings of the Second International Workshop on Active Mining (AM'2003)**, Maebashi City, Japan, October, 2003, 42-51
5. Dardzińska, A., Raś, Z.W. (2003) On Rules Discovery from Incomplete Information Systems, in **Proceedings of ICDM'03 Workshop on Foundations and New Directions of Data Mining**, (Eds: T.Y. Lin, X. Hu, S. Ohsuga, C. Liau), Melbourne, Florida, IEEE Computer Society, 2003, 31-35
6. Dardzińska, A., Raś, Z.W. (2003) Chasing Unknown Values in Incomplete Information Systems, in **Proceedings of ICDM'03 Workshop on Foundations and New Directions of Data Mining**, (Eds: T.Y. Lin, X. Hu, S. Ohsuga, C. Liau), Melbourne, Florida, IEEE Computer Society, 2003, 24-30
7. Fensel, D., (1998), *Ontologies: a silver bullet for knowledge management and electronic commerce*, Springer-Verlag, 1998
8. Giudici, P. (2003) Applied Data Mining, Statistical Methods for Business and Industry, Wiley, West Sussex, England
9. Guarino, N., ed. (1998) Formal Ontology in Information Systems, IOS Press, Amsterdam
10. Guarino, N., Giaretta, P. (1995) Ontologies and knowledge bases, towards a terminological clarification, in *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, IOS Press
11. Pawlak, Z. (1991) Rough sets-theoretical aspects of reasoning about data, Kluwer, Dordrecht
12. Pawlak, Z. (1991) Information systems - theoretical foundations, in **Information Systems Journal**, Vol. 6, 1981, 205-218
13. Raś, Z.W. (1994) Dictionaries in a distributed knowledge-based system, in **Concurrent Engineering: Research and Applications**, Conference Proceedings, Pittsburgh, Penn., Concurrent Technologies Corporation, pp. 383-390
14. Raś, Z.W., Dardzińska, A. (2004) Ontology Based Distributed Autonomous Knowledge Systems, in **Information Systems International Journal**, Elsevier, Vol. 29, No. 1, 2004, 47-58
15. Raś, Z.W., Joshi, S. Query approximate answering system for an incomplete DKBS, in **Fundamenta Informaticae Journal**, IOS Press, Vol. 30, No. 3/4, 1997, 313-324
16. Sowa, J.F. (2000a) Ontology, metadata, and semiotics, in B. Ganter & G. W. Mineau, eds., *Conceptual Structures: Logical, Linguistic, and Computational Issues*, LNAI, No. 1867, Springer-Verlag, Berlin, 2000, pp. 55-81
17. Sowa, J.F. (2000b) Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks/Cole Publishing Co., Pacific Grove, CA.
18. Sowa, J.F. (1999a) Ontological categories, in L. Albertazzi, ed., *Shapes of Forms: From Gestalt Psychology and Phenomenology to Ontology and Mathematics*, Kluwer Academic Publishers, Dordrecht, 1999, pp. 307-340.
19. Van Heijst, G., Schreiber, A., Wielinga, B. (1997) Using explicit ontologies in KBS development, in *International Journal of Human and Computer Studies*, Vol. 46, No. 2/3, 183-292.