$CHASE_2$ - Rule Based Chase Algorithm for Information Systems of Type λ

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

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

Abstract. A rule-based chase algorithm (called $Chase_2$), presented in this paper, provides a strategy for predicting what values should replace the null values in a relational database. When information about an object is partially incomplete (a set of weighted values of the same attribute can be treated as an allowed attribute value), $Chase_2$ is decreasing that incompleteness. In other words, when several weighted values of the same attribute are assigned to an object, $Chase_2$ will increase their standard deviation. To make the presentation clear and simple, we take an incomplete information system S of type λ as the model of data. To begin $Chase_2$ process, each attribute in S that has either unknown or incomplete values for some objects in S is set, one by one, as a decision attribute and all other attributes in S are treated as condition attributes. Assuming that d is the decision attribute, we take a subsystem S_1 of S by selecting from S any object x such that $d(x) \neq NULL$. Now, the subsystem S_1 is used for extracting rules describing values of attribute d. In the next step, each incomplete slot in S which is in the column corresponding to attribute d is chased by previously extracted rules from S_1, describing d. All other incomplete attributes in a database are processed the same way. This concludes the first loop of $Chase_2$. The whole process is recursively repeated till no more new values can be predicted by $Chase_2$. In this case, we say that a fixed point in values prediction was reached.

1 Introduction

Common problems encountered by Query Answering Systems (QAS), introduced by Raś in [15], [18], either for Information Systems or for Distributed Autonomous Information Systems ($DAIS$) include the handling of incomplete attributes when answering a query. One plausible solution to answer a query involves the generation of rules describing all incomplete attributes used in a query and then chasing the unknown values in the local database with respect to the generated rules. These rules can be given by domain experts and also can be discovered either locally or at remote sites of $DAIS$. Since all unknown values would not necessarily be found, the process is repeated on the enhanced database until all unknowns are found or no new information is generated. When the fixed point is reached, QAS will run the original query against the enhanced database. The results of the query run on three versions of the information system have been compared by Dardzińska and Raś in [5]: $DAIS$ with a complete local database, $DAIS$ with incomplete local database (where incomplete information can be represented only in terms of null values), and $DAIS$ with a local incomplete database enhanced by rule-based chase algorithm. The chase algorithm presented in [5] was based only on consistent set of rules. The notion of a tableaux system and the chase algorithm based on functional dependencies F is presented for instance in [1]. Chase algorithm based on functional dependencies always terminates if applied to a finite tableaux system. It was shown that, if one execution of the algorithm generates a tableaux system that satisfies F, then every execution of the algorithm generates the same tableaux system.

There are many methods to replace missing values with predicted values or estimates [23], [19], [11], [7]. Some of them are given below:

- **Most Common Attribute Value**. It is one of the simplest methods to deal with missing attribute values. The value of the attribute that occurs most often is selected to be the value for all the unknown values of the attribute.
- **Concept Most Common Attribute Value**. This method is a restriction of the first method to the concept, i.e., to all examples with the same value of the decision. The value of the attribute, which occurs the most common within the concept is selected to be the value for all the unknown values of the attribute within that concept. This method is also called maximum relative frequency method, or maximum conditional probability method.
- **C4.5**. This method is based on entropy and splitting the example with missing attribute values to all concepts [14].
- **Method of Assigning all Possible Values of the Attribute**. In this method, an example with a missing attribute value is replaced by a set of new examples, in which the missing attribute value is replaced by all possible values of the attribute.
- **Method of Assigning all Possible Values of the Attribute Restricted to the Given Concept**. This method is a restriction of the previous method to the concept, indicated by an example with a missing attribute value.
- **Event-Covering Method**. This method is also a probabilistic approach to fill in the unknown attribute values by event-covering. Event covering is defined as a strategy of selecting a subset of statistically independent events in the outcome space of variable-pairs, disregarding whether or not the variables are statistically independent.

To impute categorical dimension missing values, two types of approaches can be used:

- **Rule Based Techniques** (e.g., association rule, rule induction techniques, etc.)
- **Statistical Modelling** (e.g., multinomial, log-linear modelling, etc.)

Two main rule based models have been considered: rule induction techniques and association rules. For categorical attributes with low cardinality domains (few values), rule induction techniques such as decision tree [14] and decision systems [10] can be used to derive the missing values. However, for categorical attributes with large cardinality domains (many values), the rule induction techniques may suffer due to too many predicted classes. In this case, the combination of association relationships among categorical attributes and statistical features of possible attribute values can be used to predict the best possible values of missing data. The discovered association relationships among different attributes can be thought as constraint information of their possible values and can then be used to predict the true values of missing attributes.

Algorithm, presented in this paper, for predicting what attribute value should replace an incomplete value of a categorical attribute in a given dataset has a clear advantage over many other methods for predicting incomplete values mainly because of the use of existing associations between values of attributes, in a chase strategy, repeatedly for each newly generated dataset till some fix-point is reached. To find these associations we can use either any association rule mining algorithm or any rule discovery algorithm like $LERS$ (see [8]), or Rosetta (see [20]). Unfortunately, these algorithms, including Chase algorithm presented by us in [5], do not handle partially incomplete data, where $a(x)$ is equal, for instance, to $\{(a_1, 1/4), (a_2, 1/4), (a_3, 1/2)\}$.

We assume here that a is an attribute, x is an object, and $\{a_1, a_2, a_3\} \subseteq V_a$. By V_a we mean the set of values of attribute a. The weights assigned to these three attribute values should be read as:

- the confidence that $a(x) = a_1$ is 1/4,
- the confidence that $a(x) = a_2$ is 1/4,
- the confidence that $a(x) = a_3$ is 1/2.

In this paper we present a new chase algorithm (called $Chase_2$) which can be used for chasing incomplete information systems with rules which do not have to be consistent (this assumption was required in $Chase_1$ presented in [5]). We propose how to compute the confidence of inconsistent rules and next we show how these rules are used by Chase2.

So, the assumption placed on incompleteness of data in this paper allows 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 given in this paper is a modification of definitions given by Dardzińska and Raś in [5],[4] and used later by Raś and Dardzińska in [17] to talk about semantic inconsistencies among sites of DIS from the query answering point of view. Type λ is introduced mainly to monitor the weights assigned to values of attributes by $Chase_2$ algorithm (the algorithm checks if they are greater than or equal to λ). If the weight assigned by $Chase_2$ to one of the attribute values describing object x is below the acceptable threshold, then this attribute value is no longer considered as a value which describes x.

2 Handling Incomplete Values using Chase Algorithms

There is a relationship between interpretation of queries and the way incomplete information in an information system is seen. Assume that we are concerned with identifying all objects in the system satisfying a given description. For example, an information system might contain information about students in a class and classify them using four attributes of *hair color, eye color, gender* and *size*. A simple query might be to find all students with *brown hair* and *blue eyes*. When the information system is incomplete, students having *brown hair* and unknown *eye color* can be handled by either including or excluding them from the answer to the query. In the first case we talk about optimistic approach to query interpretation while in the second one we talk about pessimistic approach. Another option to handle such a query is to discover rules for *eye color* in terms of the attributes *hair color, gender,* and *size*. Then, these rules can be applied to students with unknown *eye color* to discover that color and possibly the same to identify more objects satisfying the query. Consider that in our example one of the generated rules said:

$$(hair, brown) \wedge (size, medium) \rightarrow (eye, brown)$$

Thus, if one of the students having *brown hair* and *medium size* has no value for *eye color*, then this student should not be included in the list of students with *brown hair* and *blue eyes*. Attributes *hair color* and *size* are classification attributes and *eye color* is the decision attribute.

Now, let us give an example showing the relationship between incomplete information about objects in an information system and the way queries (attribute values) are interpreted. Namely, the confidence in object x that he has *brown hair* is 1/3 can be either written as $(brown, 1/3) \in hair(x)$ or $(x, 1/3) \in I(brown)$, where I is an interpretation of queries (the term *brown* is treated here as a query).

In [5] we presented $Chase_1$ strategy based on the assumption that only consistent subsets of rules extracted from an incomplete information system S can be used for replacing Null values by new less incomplete values in S. Clearly, rules discovered from S do not have to be consistent in S. Taking this fact into consideration, the algorithm $Chase_2$ proposed in this paper has less restrictions and it allows chasing information system S with inconsistent rules as well.

Assume that $S = (X, A, V)$, where $V = \bigcup\{V_a : a \in A\}$ and each $a \in A$ is a partial function from X into $2^{V_a} - \{\emptyset\}$. In the first step of $Chase$ algorithms, we identify all incomplete attributes used in S. An attribute is incomplete if there is an object in S with incomplete information on this attribute. The values of all incomplete attributes in S are treated as concepts to be learned (in a form of rules) either only from S or from S and its remote sites (if S is a part of a distributed autonomous information system).

The second step of $Chase$ algorithms is to extract rules describing these concepts. These rules are stored in a knowledge base D for S ([15],[18],[17]).

Algorithm $Chase_1$ presented in [5] assumes that all inconsistencies in D have to be repaired before they are used in the chase process. To get rules from S describing attribute value v_a of attribute a we extract them from the subsystem $S_1 = (X_1, A, V)$ of S where $X_1 = \{x \in X : card(a(x)) = 1\}$. $Chase_2$ does not have such restrictions.

The final step of $Chase$ algorithms is to replace incomplete information in S by values provided by rules in D.

3 Partially Incomplete Information Systems

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. Both systems are classifying the same set of objects X using the same set of attributes A. 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 $a_{S_2}(x)$. In other words, the containment mapping Ψ transforms any partially incomplete value $a_{S_1(x)}$ of any attribute a, describing object x, into a new value $a_{S_2(x)}$ which is more complete. We can easily agree that the condition $card(a_{S_1(x)}) > card(a_{S_2(x)})$ guarantees that. The intuitive explanation of the condition $[\sum_{i \neq j} |p_{2i} - p_{2j}| > \sum_{i \neq j} |p_{1i} - p_{1j}|]$ is the following: larger average distance between weights assigned to all possible values of attribute a describing x guarantees more precise knowledge about x with respect to attribute a. In other words, if $a_{S_1(x)} = \{(a_1, \frac{1}{2}), (a_2, \frac{1}{2})\}$, then there is no preference between a_1 or a_2. Now, if $a_{S_2(x)} = \{(a_1, \frac{1}{3}), (a_2, \frac{2}{3})\}$, then our believe in the value a_2 is higher than in the value a_1. It means that our knowledge about x with respect to a is improved because the new uncertainty is lower.

So, if containment mapping Ψ converts an information system S to S', then S' is more complete than S. In other words, it has to be a pair $(a, x) \in A \times X$ such that 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 (their standard deviation is increasing).

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 1 and Table 2, respectively.

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

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 from 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 $\Psi(S_1) = S_2$.

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 [4].

The type of incompleteness in [4] is the same as in this paper but we did not provide a threshold value λ for the minimal confidence of attribute values assigned to objects. The algorithm *ERID* works the same way for incomplete

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

information systems of type λ, since the knowledge discovery process in *ERID* is independent from the largeness of parameter λ.

Now, 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 expressions) 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 [18]):

- $N_S(v) = \{(x,p) : (v,p) \in a(x)\}$, 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 interpretation N_S was proposed by Raś & Joshi in [18]. It preserves a number of properties required for the transformation process of terms including the distributive property: $t_1 * (t_2 + t_3) = (t_1 * t_2) + (t_1 * t_3)$. This property is used in the incomplete value imputation algorithm $Chase_2$, presented in this paper, to compute correctly the confidence of rules approximating incomplete values in S.

Assume that $N_S(t_1) = \{(x_i,p_i) : i \in K\}$ and $N_S(t_2) = \{(x_i,q_i) : i \in K\}$. This notation allows to have weights p_i, q_i equal to zero. There is a number of well known options available to interpret $N_S(t_1) * N_S(t_2)$ and $N_S(t_1) + N_S(t_2)$. Some of them are listed below (see [16]).

Interpretations T_0, T_1, T_2, T_3, T_4, T_5 for the functor $*$:

– Interpretation T_0: $N_S(t_1)*N_S(t_2) = \{(x_i, S_1(p_i, q_i)) : i \in K\}$, where $S_1(p_i, q_i) =$ [if $max(p_i, q_i) = 1$, then $min(p_i, q_i)$, else 0].
– Interpretation T_1: $N_S(t_1) * N_S(t_2) = \{(x_i, max\{0, p_i + q_i - 1\}) : i \in K\}$.
– Interpretation T_2: $N_S(t_1) * N_S(t_2) = \{(x_i, \frac{[p_i \cdot q_i]}{[2-(p_i+q_i-p_i \cdot q_i)]}) : i \in K\}$.
– Interpretation T_3: $N_S(t_1) * N_S(t_2) = \{(x_i, p_i \cdot q_i) : i \in K\}$.
– Interpretation T_4: $N_S(t_1) * N_S(t_2) = \{(x_i, \frac{[p_i \cdot q_i]}{[p_i+q_i-p_i \cdot q_i]}) : i \in K\}$.
– Interpretation T_5: $N_S(t_1) * N_S(t_2) = \{(x_i, min\{p_i, q_i\}) : i \in K\}$.

Interpretations S_0, S_1, S_2, S_3, S_4, S_5 for the functor $+$:

– Interpretation S_0: $N_S(t_1) + N_S(t_2) = \{(x_i, S_2(p_i, q_i)) : i \in K\}$, where $S_2(p_i, q_i) =$ [if $min(p_i, q_i) = 0$, then $max(p_i, q_i)$, else 1].
– Interpretation S_1: $N_S(t_1) + N_S(t_2) = \{(x_i, min\{1, p_i + q_i\}) : i \in K\}$.
– Interpretation S_2: $N_S(t_1) + N_S(t_2) = \{(x_i, \frac{[p_i+q_i]}{[1+p_i \cdot q_i]}) : i \in K\}$.
– Interpretation S_3: $N_S(t_1) + N_S(t_2) = \{(x_i, p_i + q_i - p_i \cdot q_i) : i \in K\}$.
– Interpretation S_4: $N_S(t_1) + N_S(t_2) = \{(x_i, \frac{[p_i+q_i-2 \cdot p_i \cdot q_i]}{[1-p_i \cdot q_i]}) : i \in K\}$.
– Interpretation S_5: $N_S(t_1) + N_S(t_2) = \{(x_i, max\{p_i, q_i\}) : i \in K\}$.

So, by taking all combinations of (T_i, S_j), we can consider 36 possible interpretations for the pair of functors $(\cdot, +)$. Only 7 of them satisfy the distributivity law $t \cdot (t_1 + t_2) = (t \cdot t_1) + (t \cdot t_2)$. Here they are: (T_0, S_5), (T_0, S_0), (T_2, S_5), (T_3, S_5), (T_4, S_5), (T_5, S_5), (T_5, S_0). It can be easily checked that for any conjunct term t, $T_0(t) < T_1(t) < T_2(t) < T_3(t) < T_4(t) < T_5(t)$. So, T_0 is the most pessimistic whereas T_5 is the most optimistic interpretation of the operator $\cdot$. Similarly, for any disjunct term t, $S_5(t) < S_4(t) < S_3(t) < S_2(t) < S_1(t) < S_0(t)$. So, S_5 is the most pessimistic whereas S_0 is the most optimistic interpretation of the operator $+$. The choice of (T_3, S_5) for the interpretation of $(\cdot, +)$ is easily justified assuming that terms in the form of conjuncts are built only from values of different attributes.

Now, let us go back to $Chase_2$ converting information system S of type λ to a new and more complete information system $Chase_2(S)$ of the same type. This algorithm is new in comparison to known strategies for chasing incomplete data in relational tables because of the assumption concerning partial incompleteness of data (sets of weighted attribute values can be assigned to an object as its value). This assumption forced us to develop a new discovery algorithm, called *ERID*, for extracting rules from tables similar to incomplete systems of type λ (see [4]) so it can be applied in $Chase_2$ algorithm given below.

```
Algorithm $Chase_2(S, In(A), L(D))$;
  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 $\{(t_i \to v) : i \in I\}$
        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** $Chase_2(S, In(A), L(D))$ **else** $Chase(S) := S$
**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 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 $Chase_2$ converts any incomplete or partially incomplete information system S to a new information system which is more complete. At each recursive call of $Chase_2$, 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 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 $m = card(L(D))$.

To explain the algorithm, we apply $Chase_2$ to information system S_3 presented by Table 3. We assume that $L(D)$ contains the following rules (listed with their support and confidence) defining attribute e and extracted from S_3 by $ERID$:

$r_1 = [a_1 \to e_3]$	$(sup(r_1) = 1,\ conf(r_1) = 0.5)$
$r_2 = [a_2 \to e_2]$	$(sup(r_2) = 5/3,\ conf(r_2) = 0.51)$
$r_3 = [a_3 \to e_1]$	$(sup(r_3) = 17/12,\ conf(r_3) = 0.51)$
$r_4 = [b_1 \to e_1]$	$(sup(r_4) = 2,\ conf(r_4) = 0.72)$
$r_5 = [b_2 \to e_3]$	$(sup(r_5) = 8/3,\ conf(r_5) = 0.51)$
$r_6 = [c_2 \to e_1]$	$(sup(r_6) = 2,\ conf(r_6) = 0.66)$
$r_7 = [c_3 \to e_3]$	$(sup(r_7) = 7/6,\ conf(r_7) = 0.64)$
$r_8 = [a_3 \cdot c_1 \to e_3]$	$(sup(r_8) = 1,\ conf(r_8) = 0.8)$
$r_9 = [a_3 \cdot d_1 \to e_3]$	$(sup(r_9) = 1,\ conf(r_9) = 0.5)$
$r_{10} = [c_1 \cdot d_1 \to e_3]$	$(sup(r_{10}) = 1,\ conf(r_{10}) = 0.5)$

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	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	$\{(e_1, \frac{2}{3}), (e_2, \frac{1}{3})\}$
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 3. Information System S_3

It can be noticed that values $e(x_1)$, $e(x_4)$, $e(x_6)$ of the attribute e are changed in S_3 by $Chase_2$ algorithm. The next section shows how to compute these three values and how to convert them, if needed, to a new set of values satisfying the constraints required by system S_4 to remain its λ status. Similar process is

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	$\{(e_3,\frac{41}{100}),(e_2,\frac{59}{100})\}$
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	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	$\{(e_1,\frac{2}{3}),(e_2,\frac{1}{3})\}$
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_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	a_3	b_2	c_1	d_1	e_3

Table 4. Information System S_4

applied to all incomplete attributes in S_3. After all changes corresponding to all incomplete attributes are recorded, system S_3 is replaced by $\Psi(S_3)$ and the whole process is recursively repeated till a fix point is reached.

Algorithm $Chase_2$ will compute new value for $e(x_1) = \{(e_1, 1/2), (e_2, 1/2)\}$ denoted by $e_{new}(x_1) = \{(e_1, ?), (e_2, ?), (e_3, ?)\}$. To do that $Chase_2$ identifies all rules in $L(D)$ supported by x_1. It can be easily checked that r_1, r_2, r_4, r_5, and r_{10} are the rules supported by x_1. To calculate support of x_1 for r_1, we take: $1 \cdot \frac{1}{2} \cdot \frac{1}{3}$. In a similar way we calculate the support of x_1 for the remaining rules. As the result, we get the list of weighted values of attribute e supported by $L(D)$ for x_1, as follows:

$(e_3, \frac{1}{3} \cdot 1 \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{8}{3} \cdot \frac{51}{100} + 1 \cdot 1 \cdot \frac{1}{2}) = (e_3, 1.119)$
$(e_2, \frac{2}{3} \cdot \frac{5}{3} \cdot \frac{51}{100}) = (e_2, 1.621)$
$(e_1, \frac{2}{3} \cdot 2 \cdot \frac{72}{100}) = (e_1, 0.96)$.
So the value of attribute e for x_1 supported by $L(D)$ will be:
$e_{new}(x_1) = \{(e_1, \frac{0.96}{0.96+1.621+1.119}), (e_2, \frac{1.621}{0.96+1.621+1.119}), (e_3, \frac{1.119}{0.96+1.621+1.119})$
$= \{(e_1, 0.26), (e_2, 0.44), (e_3, 0.302)\}$

In a similar way we compute the value of e for x_4:
$(e_3, 1 \cdot 1 \cdot \frac{1}{2}) = (e_3, 0.5)$
$(e_2, 0)$
$(e_1, 1 \cdot \frac{17}{12} \cdot \frac{51}{100}) = (e_1, 0.722)$
we have:
$e_{new}(x_4) = \{(e_1, \frac{0.722}{0.5+0.722}), (e_3, \frac{0.5}{0.5+0.722})\} = \{(e_1, 0.59), (e_3, 0.41)\}$

And finally, for x_6:
$(e_3, \frac{8}{3} \cdot 1 \cdot \frac{51}{100} + 1 \cdot \frac{7}{6} \cdot \frac{64}{100}) = (e_3, 2.11)$
$(e_2, 1 \cdot \frac{5}{3} \cdot \frac{51}{100}) = (e_2, 0.85)$
$(e_1, 0)$
we have:
$e_{new}(x_6) = \{(e_2, \frac{0.85}{2.11+0.85}), (e_3, \frac{2.11}{2.11+0.85})\} = \{(e_2, 0.29), (e_3, 0.713)\}$

For $\lambda = 0.3$ the values of $e(x_1)$ and $e(x_6)$ will change to:
$e(x_1) = \{(e_2, 0.59), (e_3, 0.41)\}, \quad e(x_6) = \{(e_3, 1)\}$.

Table 4 shows the resulting table.

Initial testing performed on several incomplete tables of the size $50 \times 2,000$ with randomly generated data gave us quite promising results concerning the precision of $Chase_2$. We started with a complete table S and removed from it 10 percent of its values randomly. This new table is denoted by S'. For each incomplete column in S', let's say d, we use $ERID$ to extract rules defining d in terms of other attributes in S'. These rules are stored in $L(D)$. In the following step, we apply $Chase_2$, making d maximally complete. Independently, the same procedure is applied to all other incomplete columns. As the result, we obtain a new table S'. Now, the whole procedure is repeated again on S'. The process continues till the fix point is reached. Now, we compare the new values stored in the empty slots of the initial table S' with the corresponding values in S. Based on this comparison, we easily compute the precision of $Chase_2$.

4 Conclusion

We expect much better results if a single information system is replaced by distributed autonomous information systems investigated in [15], [17], [18]. This is justified by experimental results showing higher confidence in rules extracted through distributed data mining than in rules extracted through local mining.

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) 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
5. 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
6. Fensel, D., (1998), *Ontologies: a silver bullet for knowledge management and electronic commerce*, Springer-Verlag, 1998
7. Giudici, P. (2003) Applied Data Mining, Statistical Methods for Business and Industry, Wiley, West Sussex, England

8. Grzymala-Busse, J. (1991) On the unknown attribute values in learning from examples, in *Proceedings of ISMIS'91*, LNCS/LNAI, Springer-Verlag, Vol. 542, 1991, 368-377
9. Grzymala-Busse, J. (1997) A new version of the rule induction system *LERS*, in *Fundamenta Informaticae*, IOS Press, Vol. 31, No. 1, 1997, 27-39
10. Grzymala-Busse, J., Hu, M. (2000) A Comparison of several approaches to missing attribute values in data mining, in *Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing*, RSCTC'00, Banff, Canada, 340-347
11. Little, R., Rubin, D.B. (1987) Statistical analysis with missing data, New York, John Wiley and Sons
12. Pawlak, Z. (1991) Rough sets-theoretical aspects of reasoning about data, Kluwer, Dordrecht
13. Pawlak, Z. (1991) Information systems - theoretical foundations, in **Information Systems Journal**, Vol. 6, 1981, 205-218
14. Quinlan, J. (1989) Unknown attribute values in induction, in *Proceedings of the Sixth International Machine Learning Workshop*, 164-168
15. Raś, Z.W. (1997) 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
16. Raś, Z.W., Arramreddy, S. (2004) Rough sets approach for handling inconsistencies in semantics of queries, PPT presentation on http://www.cs.uncc.edu/~ras/KDD-02/1
17. 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
18. 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
19. Schafer, J.L. (1997) Analysis of incomplete multivariate data, Book 72, Chapman and Hall series Monographs on Statistics and Applied Probability, Chapman and Hall, London
20. Skowron, A (1993) Boolean reasoning for decision rules generation, in J. Komorowski & Z. W. Raś, eds., *Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems*, LNAI, Springer Verlag, No. 689, 1993, 295-305
21. Sowa, J.F. (2000b) Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks/Cole Publishing Co., Pacific Grove, CA.
22. 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.
23. Wu, X., Barbara, D. (2002) Learning missing values from summary constraints, in *KDD Explorations*, Vol. 4, No. 1
24. 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.