Action rules mining

Angelina A. Tzacheva[1] and Zbigniew W. Raś[1,2],

[1] UNC-Charlotte, Computer Science Dept., Charlotte, NC 28223, USA
[2] Polish Academy of Sciences, Institute of Computer Science, Ordona 21, 01-237 Warsaw, Poland

Abstract. Action rules introduced in [6] and investigated further in [7] assume that attributes in a database are divided into two groups: stable and flexible. In general, an action rule can be constructed from two rules extracted earlier from the same database. Furthermore, we assume that these two rules describe two different decision classes and our goal is to re-classify objects from one of these classes into the other one. Flexible attributes are essential in achieving that goal since they provide a tool for making hints to a user what changes within some values of flexible attributes are needed for a given group of objects to re-classify them into a new decision class. Ras and Gupta [9] proposed a method for constructing action rules when information system is distributed and its sites are built idependently. In paper by Ras and Tzacheva [8], a new subclass of attributes called semi-stable attributes is introduced. Semi-stable attributes are typically a function of time, and undergo deterministic changes (for instance attribute *age* or *height*). So, the set of conditional attributes is partitioned into stable, semi-stable, and flexible. Depending on semantics of attributes, some semi-stable attributes can be treated as flexible and the same new action rules can be constructed. These new action rules are usually built to replace some existing action rules which confidence is too low to be of any interest to a user. The confidence of new action rules is always higher than the confidence of rules they replace. Additionally, the notion of a cost and feasibility of an action rule is introduced in this paper. A heuristic strategy for constructing feasible action rules which have high confidence and possibly the lowest cost is proposed.

1 Introduction

Action rules, introduced by Raś and Wieczorkowska in [6] and investigated further in [7], [9] may be utilized by any type of industry maintaining large databases, including financial sector and e-commerce. Built from classification rules extracted from a decision system, these rules suggest ways to re-classify consumers to a desired state. However, quite often, such a change cannot be done directly to a chosen attribute (for instance to the attribute *profit*). In that situation, definitions of such an attribute in terms of other attributes have to be learned. These definitions are used to construct action rules showing what changes in values of attributes, for a given consumer, are needed in order to re-classify this consumer the way business user wants. This re-classification may mean that a consumer not interested in a certain product, now may buy it, and therefore may shift into a group of more profitable customers. These groups of customers are described by values of classification attributes in a decision system schema. Raś and Gupta [9] start with distributed autonomous information systems in their investigations. They claim that it is wise to search for action rules at remote sites when action rules extracted at the client site can not be implemented in practice (they are too expensive, too risky, or business user is unable to make such changes). Also, they show under what assumptions two action rules extracted at two different sites can be composed. One of these assumptions says that semantics of attributes, including the interpretation of null values, have to be the same at both of these sites. In the present paper, this assumption is relaxed. Additionally, we introduce the notion of a cost and feasibility of an action rule. Usually, a number of action rules or chains of action rules can be applied to re-classify a given set of objects. Changes of values within one attribute can be achieved more easily than changes of values within another attribute. We present a strategy for constructing chains of action rules, driven by a change of attribute values suggested by another action rule, which are needed to reclassify certain objects. This chain of action rules uniquely defines a new action rule and it is built with a goal to lower the reclassification cost for these objects.

In papers [6], [7] and, [9] all attributes are divided into stable and flexible.

In paper by Raś and Tzacheva [8], a new subclass of attributes called semi-stable attributes is introduced. Semi-stable attributes are typically a function of time, and undergo deterministic changes (for instance attribute *age* or *height*). So, in general, it is assumed that the set of conditional attributes is partitioned into stable, semi-stable, and flexible. It was shown in [8] that some semi-stable attributes can be treated the same way as flexible attributes. By providing a strategy for identifying flexible attributes among semi-stable attributes we also give a new tool for lowering the cost of action rules.

Two methods applicable for detection of non-standard semantics are proposed. The first method is based on detection of abnormal relationships between values of two semi-stable attributes. The second one is based on the property of random selection of sites from Distributed Information System *DIS* and testing how many of them support a rule under consideration. Support only coming from a small number of sites gives us a hint that the decision attribute of that rule has most probably non-standard semantics.

2 Information System and Action Rules

An information system is used for representing knowledge. Its definition, given here, is due to Pawlak [3].

By an information system we mean a pair $S = (U, A, V)$, where:

- U is a nonempty, finite set called the universe,
- A is a nonempty, finite set of attributes i.e. $a : U \longrightarrow V_a$ is a function for $a \in A$,
- $V = \bigcup\{V_a : a \in A\}$, where V_a is a set of values of the attribute $a \in A$.

Elements of U are called objects. In this paper, they are often seen as customers. Attributes are interpreted as features, offers made by a bank, characteristic conditions etc.

By a decision table we mean any information system where the set of attributes is partitioned into conditions and decisions. Additionally, we assume that the set of conditions is partitioned into stable and flexible. For simplicity reason, we assume that there is only one decision attribute. Date of Birth is an example of a stable attribute. Interest rate on any customer account is an example of a flexible attribute (dependable on bank). We adopt the following definition of a decision table:

By a decision table we mean an information system of the form $S = (U, A_{St} \cup A_{Fl} \cup \{d\})$, where $d \notin A_{St} \cup A_{Fl}$ is a distinguished attribute called decision. The elements of A_{St} are called stable conditions, whereas the elements of A_{Fl} are called flexible conditions.

As an example of a decision table we take $S = (\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}, \{a, c\} \cup \{b\} \cup \{d\})$ represented by Table 1. The set $\{a, c\}$ lists stable attributes, b is a flexible attribute and d is a decision attribute. Also, we assume that H denotes a *high* profit and L denotes a *low* one.

X	a	b	c	d
x_1	0	S	0	L
x_2	0	R	1	L
x_3	0	S	0	L
x_4	0	R	1	L
x_5	2	P	2	L
x_6	2	P	2	L
x_7	2	S	2	H
x_8	2	S	2	H

Table 1. Decision System S

In order to induce rules in which the THEN part consists of the decision attribute d and the IF part consists of attributes belonging to $A_{St} \cup A_{Fl}$, subtables $(U, B \cup \{d\})$ of S where B is a d-reduct (see [3]) in S should be used for rules extraction. By $L(r)$ we mean all attributes listed in the IF part of a rule r. For example, if $r = [(a, 2) * (b, S) \rightarrow (d, H)]$ is a rule then

$L(r) = \{a, b\}$. By $d(r)$ we denote the decision value of a rule. In our example $d(r) = H$. If r_1, r_2 are rules and $B \subseteq A_{St} \cup A_{Fl}$ is a set of attributes, then $r_1/B = r_2/B$ means that the conditional parts of rules r_1, r_2 restricted to attributes B are the same. For example if $r_1 = [(b, S) * (c, 2) \to (d, H)]$, then $r_1/\{b\} = r/\{b\}$.

In our example, we get the following optimal rules:
$(a, 0) \to (d, L)$, $(c, 0) \to (d, L)$,
$(b, R) \to (d, L)$, $(c, 1) \to (d, L)$,
$(b, P) \to (d, L)$, $(a, 2) * (b, S) \to (d, H)$, $(b, S) * (c, 2) \to (d, H)$.

Now, let us assume that $(a, v \to w)$ denotes the fact that the value of attribute a has been changed from v to w. Similarly, the term $(a, v \to w)(x)$ means that $a(x) = v$ has been changed to $a(x) = w$. Saying another words, the property (a, v) of object x has been changed to property (a, w).

Let $S = (U, A_{St} \cup A_{Fl} \cup \{d\})$ is a decision table and rules r_1, r_2 have been extracted from S. Assume that B_1 is a maximal subset of A_{St} such that $r_1/B_1 = r_2/B_1$, $d(r_1) = k_1$, $d(r_2) = k_2$ and $k_1 \leq k_2$. Also, assume that $(b_1, b_2, ..., b_p)$ is a list of all attributes in $L(r_1) \cap L(r_2) \cap A_{Fl}$ on which r_1, r_2 differ and $r_1(b_1) = v_1$, $r_1(b_2) = v_2$,..., $r_1(b_p) = v_p$, $r_2(b_1) = w_1$, $r_2(b_2) = w_2$,..., $r_2(b_p) = w_p$.

By (r_1, r_2)-action rule r on $x \in U$ we mean the expression below (see [6]):
$[(b_1, v_1 \to w_1) \wedge (b_2, v_2 \to w_2) \wedge ... \wedge (b_p, v_p \to w_p)](x) \Rightarrow$
$[(d, k_1 \to k_2)](x)$.

We say that object x *supports* (r_1, r_2)-action rule r if:

- $(\forall i \leq p)[[b_i \in L(r)] \longrightarrow [b_i(x) = v_i]]$ and $d(x) = k_1$,
- object x supports rule r_1,
- if object y is the outcome of (r_1, r_2)-action rule r applied on x, then y supports rule r_2.

By the support of (r_1, r_2)-action rule r in S, denoted by $Sup_S(r)$, we mean the set of all objects in S supporting r.

3 Distributed Information System

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 showing which systems can directly communicate with each other.

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

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

Consistency basically means that information about any object x in one system has to be either more general or more specific than in the other. Saying another words two consistent systems can not have conflicting information stored about any object x belonging to both of them.

Another problem which has to be taken into consideration is semantics of attributes used at more than one site. This semantics may easily differ among sites. 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 two information systems are complete and they use the same attribute but of a different granularity level to describe objects, 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 system is distributed and chase techniques, based on rules extracted at more than one site, are used by a client site to replace its incomplete values of attributes by new values which are more complete (see [5]).

In this paper we 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 $S_i's$ 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 action rules extracted from S_i, which means that $D = \bigcup_{i \in I} L(D_i)$ is a set of action rules which can be used in the process of distributed action rules discovery. Now, let us say that system S_k, $k \in I$ is queried be a user for an action rule re-classifying objects with respect to decision attribute d. Any strategy for discovering action rules from S_k based on action rules $D' \subset D$ is called *sound* if the following three conditions are satisfied:

- for any action rule in D', the value of its decision attribute d is of the granularity level either equal to or finer than the granularity level of the attribute d in S_k.
- for any action rule in D', the granularity level of any attribute a used in the classification part of that rule is either equal or softer than the granularity level of a in S_k.
- attribute used in the decision part of a rule has to be classified as flexible in S_k.

We assume, in the following sections, that the interpretation of any attribute and the granularity level of its values is the same at all sites in which it is used.

4 Cost and Feasibility of Action Rules

Assume now that $DS = (\{S_i : i \in I\}, L)$ is a distributed information system (*DIS*), where $S_i = (X_i, A_i, V_i)$, $i \in I$. Let $b \in A_i$ is a flexible attribute in S_i and $b_1, b_2 \in V_i$ are its two values. By $\rho_{S_i}(b_1, b_2)$ we mean a number from $(0, +\infty]$ which describes the average cost needed to change the attribute value from b_1 to b_2 for any of the qualifying objects in S_i. Object $x \in X_i$ qualifies for the change from b_1 to b_2, if $b(x) = b_1$. If the above change is not feasible in practice, for one of the qualifying objects in S_i, then we write $\rho_{S_i}(b_1, b_2) = +\infty$. The value of $\rho_{S_i}(b_1, b_2)$ close to *zero* is interpreted that the change of values from b_1 to b_2 is quite trivial to accomplish for qualifying objects in S_i whereas any large value of $\rho_{S_i}(b_1, b_2)$ means that this change of values is practically very difficult to achieve for some of the qualifying objects in S_i.

If $\rho_{S_i}(b_1, b_2) < \rho_{S_i}(b_3, b_4)$, then we say that the change of values from b_1 to b_2 is *more feasible* than the change from b_3 to b_4.

We assume here that the values $\rho_{S_i}(b_{j1}, b_{j2})$ are provided by experts for each of the information systems S_i. They are seen as atomic expressions and will be used to introduce the formal notion of the feasibility and the cost of action rules in S_i.

So, let us assume that $r = [(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \wedge (b_p, v_p \rightarrow w_p)](x) \Rightarrow (d, k_1 \rightarrow k_2)(x)$ is a (r_1, r_2)-action rule. By the *cost* of r denoted by $cost(r)$ we mean the value $\sum\{\rho_{S_i}(v_k, w_k) : 1 \leq k \leq p\}$. We say that r is *feasible* if $cost(r) < \rho_{S_i}(k_1, k_2)$.

It means that for any feasible rule r, the cost of the conditional part of r is lower than the cost of its decision part and clearly $cost(r) < +\infty$.

Assume now that d is a decision attribute in S_i, $k_1, k_2 \in V_d$, and the user would like to re-classify customers in S_i from the group k_1 to the group k_2. To achieve that, he may look for an appropriate action rule, possibly of the lowest cost value, to get a hint which attribute values have to be changed. To be more precise, let us assume that $R_{S_i}[(d, k_1 \rightarrow k_2)]$ denotes the set of all action rules in S_i having the term $(d, k_1 \rightarrow k_2)$ on their decision site. For simplicity reason, in Section 5 of this paper, attribute d will be omitted in $(d, k_1 \rightarrow k_2)$. Now, among all action rules in $R_{S_i}[(d, k_1 \rightarrow k_2)]$ he may identify a rule which has the lowest cost value. But the rule he gets may still have the cost value much to high to be of any help to him. Let us notice that the cost of the action rule

$r = [(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge ... \wedge (b_p, v_p \rightarrow w_p)](x) \Rightarrow (d, k_1 \rightarrow k_2)(x)$
might be high only because of the high cost value of one of its sub-terms in the conditional part of the rule.

Let us assume that $(b_j, v_j \rightarrow w_j)$ is that term. In such a case, we may look for an action rule in $R_{S_i}[(b_j, v_j \rightarrow w_j)]$ which has the smallest cost value.

Assume that $r_1 = [(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge ... \wedge (b_{jq}, v_{jq} \rightarrow w_{jq})](y) \Rightarrow (b_j, v_j \rightarrow w_j)(y)$ is such a rule which is also feasible in S_i. Since $x, y \in X_i$, we can compose r with r_1 getting a new feasible rule which is given below:

$[(b_1, v_1 \rightarrow w_1) \wedge ... \wedge [(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge ... \wedge (b_{jq}, v_{jq} \rightarrow w_{jq})] \wedge ... \wedge (b_p, v_p \rightarrow w_p)](x) \Rightarrow (d, k_1 \rightarrow k_2)(x)$.

Clearly, the cost of this new rule is lower than the cost of r. However, if its support in S_i gets too low, then such a rule has no value to the user. Otherwise, we may recursively follow this strategy trying to lower the cost needed to re-classify objects from the group k_1 into the group k_2. Each successful step will produce a new action rule which cost is lower than the cost of the current rule. Obviously, this heuristic strategy always ends.

One can argue that if the set $R_{S_i}[(d, k_1 \rightarrow k_2)]$ contains all action rules reclassifying objects from group k_1 into the group k_2 then any new action rule, obtained as the result of the above recursive strategy, should be already in that set. We agree with this statement but in practice $R_{S_i}[(d, k_1 \rightarrow k_2)]$ is only a subset of all action rules. Firstly, it is too expensive to generate all possible classification rules from an information system (available knowledge discovery packages extract only the shortest or close to the shortest rules) and secondly even if we extract such rules it still takes too much time to generate all possible action rules from them. So the applicability of the proposed recursive strategy, to search for new rules possibly of the lowest cost, is highly justified.

Again, let us assume that the user would like to reclassify some objects in S_i from the class b_1 to the class b_2 and that $\rho_{S_i}(b_1, b_2)$ is the current cost to do that. Each action rule in $R_{S_i}[(d, k_1 \rightarrow k_2)]$ gives us an alternate way to achieve the same result but under different costs. If we limit ourself to the system S_i, then clearly we can not go beyond the set $R_{S_i}[(d, k_1 \rightarrow k_2)]$. But, if we allow to extract action rules at other information systems and use them jointly with local action rules, then the number of attributes which can be involved in reclassifying objects in S_i will increase and the same we may further lower the cost of the desired reclassification.

So, let us assume the following scenario. The action rule $r = [(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge ... \wedge (b_p, v_p \rightarrow w_p)](x) \Rightarrow (d, k_1 \rightarrow k_2)(x)$, extracted from the information system S_i, is not feasible because at least one of its terms, let us say $(b_j, v_j \rightarrow w_j)$ where $1 \leq j \leq p$, has too high cost $\rho_{S_i}(v_j, w_j)$ assign to it.

In this case we look for a new feasible action rule $r_1 = [(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge ... \wedge (b_{jq}, v_{jq} \rightarrow w_{jq})](y) \Rightarrow (b_j, v_j \rightarrow w_j)(y)$ which concatenated with r will decrease the cost value of desired reclassification. So, the current setting looks the same to the one we already had except that this time we additionally assume that r_1 is extracted from another information system in DS. For simplicity reason, we also assume that the semantics and the granularity levels of all attributes listed in both information systems are the same.

By the concatenation of action rule r_1 with action rule r we mean a new feasible action rule $r_1 \circ r$ of the form:
$[(b_1, v_1 \rightarrow w_1) \wedge ... \wedge [(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge ... \wedge (b_{jq}, v_{jq} \rightarrow w_{jq})] \wedge ... \wedge (b_p, v_p \rightarrow w_p)](x) \Rightarrow (d, k_1 \rightarrow k_2)(x)$
where x is an object in $S_i = (X_i, A_i, V_i)$.

Some of the attributes in $\{b_{j1}, b_{j2}, ..., b_{jq}\}$ may not belong to A_i. Also, the support of r_1 is calculated in the information system from which r_1 was extracted. Let us denote that system by $S_m = (X_m, A_m, V_m)$ and the set of objects in X_m supporting r_1 by $Sup_{S_m}(r_1)$. Assume that $Sup_{S_i}(r)$ is the set of objects in S_i supporting rule r. The domain of $r_1 \circ r$ is the same as the domain of r which is equal to $Sup_{S_i}(r)$. Before we define the notion of a similarity between two objects belonging to two different information systems, we assume that $A_i = \{b_1, b_2, b_3, b_4\}$, $A_m = \{b_1, b_2, b_3, b_5, b_6\}$, and objects $x \in X_i$, $y \in X_m$ are defined by the table below:

	b_1	b_2	b_3	b_4	b_5	b_6
x	v_1	v_2	v_3	v_4		
y	v_1	w_2	w_3		w_5	w_6

Table 2. Object x from S_i and y from S_m

The similarity $\rho(x, y)$ between x and y is defined as: $[1 + 0 + 0 + 1/2 + 1/2 + 1/2] = [2 + 1/2]/6 = 5/12$. To give more formal definition of similarity, we assume that:
$\rho(x, y) = [\Sigma\{\rho(b_i(x), b_i(y)) : b_i \in (A_i \cup A_m)\}]/card(A_i \cup A_m)$, where:

- $\rho(b_i(x), b_i(y)) = 0$, if $b_i(x) \neq b_i(y)$,
- $\rho(b_i(x), b_i(y)) = 1$, if $b_i(x) = b_i(y)$,
- $\rho(b_i(x), b_i(y)) = 1/2$, if either $b_i(x)$ or $b_i(y)$ is undefined.

Let us assume that $\rho(x, Sup_{S_m}(r_1)) = max\{\rho(x, y) : y \in Sup_{S_m}(r_1)\}$, for each $x \in Sup_{S_i}(r)$. By the confidence of $r_1 \circ r$ we mean $Conf(r_1 \circ r) = [\sum\{\rho(x, Sup_{S_m}(r_1)) : x \in Sup_{S_i}(r)\}/card(Sup_{S_i}(r))] \cdot Conf(r_1) \cdot Conf(r)$, where $Conf(r)$ is the confidence of the rule r in S_i and $Conf(r_1)$ is the confidence of the rule r_1 in S_m.

If we allow to concatenate action rules extracted from S_i with action rules extracted at other sites of DIS, we are increasing the total number of generated action rules and the same our chance to lower the cost of reclassifying objects in S_i is also increasing but possibly at a price of their decreased confidence.

5 Semi-Stable Attributes

The notion of action rules introduced by Ras and Wieczorkowska (see [6]), and recalled in Section 2, is based on attributes divided into two groups: stable and flexible. In the process of action rule extraction from $S_i = (X_i, A_i, V_i)$, $i \in A_i$, stable attributes are highly undesirable. Ras and Tzacheva proposed in [8] a new classification of attributes dividing them into stable, semi-stable, and flexible by taking into consideration semantics (*interpretation*) of attributes which may easily differ between information systems.

Value $a(x)$ of a stable attribute $a \in A_i$ for any object $x \in X_i$ cannot be changed in any interpretation of a (except its granularity level). All such interpretations are called standard.

An example of a stable attribute is *place of birth*. As we already mentioned, standard interpretations of this attribute may only differ in a granularity level (we can provide street address and the town name or just the town name). It is possible that one stable attribute implies another one. There is a special subset of attributes called semi-stable, which at the first impression may look stable, but they are functions of time and undergo changes in a deterministic way. Therefore, they cannot be called stable. The change is not necessarily in a linear fashion (see Figure 1).

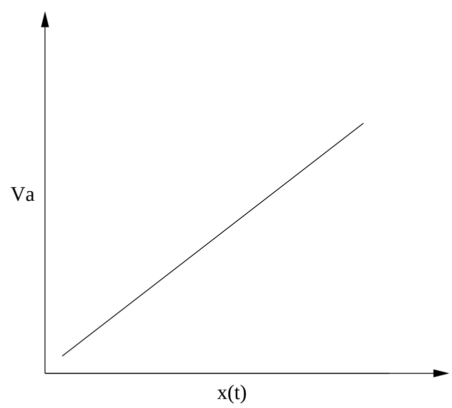

Fig. 1. Semi-stable attribute *age*

An attribute may be stable for a period of time, and then begin changing in certain direction as shown in Figure 2. Semi-stable attributes may have many interpretations, some of which are nonstandard. We denote by $SInt(a)$ the set of standard interpretations of attribute a and by $NSInt(a)$ the set of non-standard interpretations of a. If the attribute a has a nonstandard interpretation, $I(a) \in NSInt(a)$, then there is a possibility that it can be changed, and thus it may be seen as a flexible attribute in action rule extraction.

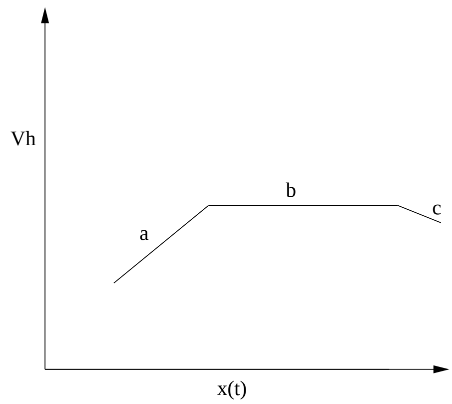

Fig. 2. Semi-stable attribute *height*

For instance, if $a =$ *age* and $V_a = \{$*young, middle-aged, old*$\}$, the author of the information system may indeed input *young* for a person who behaves as young when their actual age is *middle-aged*. Then the interpretation is nonstandard. The user can therefore influence this attribute. For example, if the following action rule was mined for object x

$$r_1 = [[(a, \textit{young} \rightarrow \textit{middle-aged})](x) \Rightarrow [(d, L \rightarrow H)](x)]$$

with respect to decision attribute d (ex. *loyalty*), the user may like to change the attribute value *young* to *middle-aged* for object x. Since the interpretation of a is nonstandard related to the behavior associated with certain age, if the object is put into special conditions that can affect its behavior, such as top university, the attribute value can be changed, and the same object x might be re-classified from *low loyalty* to *high loyalty*.

Many cases of nonstandard interpretations of attributes can be found in information systems. It is particularly important to detect them when we mine for rules in distributed information systems (see [5]). An example is the attribute *height*. Consider the following situation: Chinese people living in the mountains are generally taller than majority of Chinese population.

Let $V_a = \{short, medium, tall\} \subset V_i$ for attribute $a = height$, and system S_i contains data about Chinese population only in the mountains area. The author of the information system may consider a certain Chinese person living in the mountains *medium height* in relation to the rest. Now assume another information system S_m containing data about Chinese people living in popular urban area. In distributed action rules extraction, if S_m is mined for rules, the interpretation would regard the height value *medium* from S_i as *tall*. Therefore, the interpretation of a in S_i is nonstandard.

Numeric attributes may possess nonstandard interpretations as well. Consider for instance the attribute *number of children*. When one is asked about the number of children one has, that person may count step-children, children he is taking care of, or even children who have died. In such a case, the interpretation is nonstandard.

A flexible attribute is an attribute which value for a given object varies with time, and can be changed by a user. Also, flexible attributes may have many interpretations. *Interest rate* is an example of a flexible attribute.

Assume that $S_i = (X_i, A_i, V_i)$ is an information system which represents one of the sites of a distributed information system DIS. Also, let us assume that each attribute in A_i is either *stable* or *flexible* but we may not have sufficient temporal information about *semi-stable* attributes in S_i to decide which one is stable and which one is flexible. In such cases we can search for additional information, usually at remote sites for S_i, to classify uniquely these semi-stable attributes either as stable or flexible.

Clearly, by increasing the number of flexible attributes in S_i we also increase our chance to find cheaper action rules and the same lower the cost of re-classifying objects in S_i.

6 Discovering Semantic Inconsistencies

Different interpretations of *flexible* and *semi-stable* attributes may exist. Semi-stable attributes, in a non-standard interpretation, can be classified as flexible attributes and therefore can be used in action rule extraction. We discuss a detection method of nonstandard interpretations of a semi-stable attribute at local information system level, and next at distributed information systems level.

Detection of a nonstandard interpretation at local level is limited to the dependency of one semi-stable attribute to another semi-stable attribute for which it is known that its interpretation is standard. Attribute related to *time* must be available in the information system, such as the attribute *age*. Furthermore, information about certain breakpoints in attribute behavior is required, such as the break points shown in Figure 2. This information can be stored in the information system ontology [2].

Assume that $S = (X, A, V)$ is an information system and I_S is the interpretation of attributes in S. Also, assume that both $a, b \in A$ are semi-stable attributes, $I_S(a) \in SInt(a)$ and the relation $\prod_{\{a,b\}}(S) \subseteq \{(v_a, v_b) : v_a \in V_a, v_b \in V_b\}$ is obtained by taking projection of the system S on $\{a, b\}$. The ontology information about break points for attributes a and b in S, represented in the next section as relation $R_{\{I_S(a),I_S(b)\}}$, is assuming that the interpretation I_S in S is standard for the attribute b. It is possible that some tuples in $\prod_{\{a,b\}}(S)$ do not satisfy the break points requirement given. In such a case the interpretation I_S of b is nonstandard, $I_S(b) \in NSInt(a)$.

Consider the following situation:

Example 1. Assume that $S_1 = (X_1, \{a\} \cup \{h, j\} \cup \{b\}, V)$ is an information system represented by Table 6, where $\{a\}$ is a set of stable attributes, $\{h, j\}$ is a set of semi-stable attributes, and $\{b\}$ is a set of flexible attributes, where h is *height* and j is *number of cousins*. The interpretation of j in S_1 is known to be standard, $I_{S_1}(j) \in SInt(j)$. The system represents a local site.

	a	h	b	j
x_1	0	a	S	m
x_2	0	b	R	m
x_3	0	b	S	n
x_4	0	c	R	m
x_5	2	b	P	n
x_6	2	b	P	n
x_7	2	a	S	m
x_8	2	c	S	m

Table 3. Information system S_1.

Figure 3 shows the break points defined by the system's ontology for attributes h and j as a function of time t. The number of cousins grows as the height grows, since the person is young, and the parents' brothers and sisters have newborn children. The number of cousins decreases, as the height becomes constant or shrinks, since for a person who is middle-aged or old, the number of his/her cousins naturally decreases as they die. Therefore,

if $I_{S_1}(h) \in SInt(h)$ and $I_{S_1}(j) \in SInt(j)$, then
$$R_{\{I_{S_1}(h),I_{S_1}(j)\}} = \{(a, m), (b, m), (b, n), (c, n)\}$$

is placed in the ontology for System S_1.

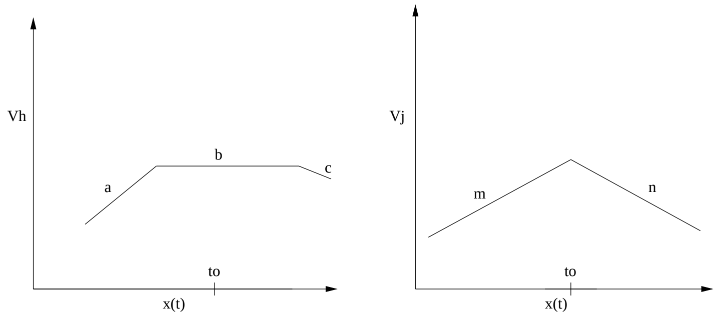

Fig. 3. Dependency relation between attributes h and j assuming standard interpretation

From Figure 3, we see that relation instance $(c, m) \in \prod_{h,j}(S_1)$ representing objects x_4, x_8 does not belong to $R_{\{I_{S_1}(h), I_{S_1}(j)\}}$. Therefore, $I(h) \in NSInt(j)$.

In other words, objects x_4, x_8 do not satisfy the break point requirement given in Figure 3, thus the interpretation of attribute *height* is nonstandard.

Standalone information systems provide limited capability of detecting nonstandard semantics whereas distributed information systems supply greater ability to have it detected. They also give the opportunity to seek for an alternative solution at remote sites. This is particularly important in a situation when they are either not willing or not able to undertake the steps suggested by action rules extracted locally. In a distributed information system (*DIS*) environment, semantic inconsistencies can be detected even if temporal information is not available. With large number of sites in *DIS* containing similar attributes, certain trends can be observed, such as classification rules with high confidence and support common for many sights. In such a situation, it is also possible that a small number of remaining sites does not support those common rules, or even contradict them. This gives us a hint that there is every likelihood for nonstandard attribute interpretation at information systems in that small number of sites.

Assume that $DS = (\{S_i : i \in I\}, L)$ is a distributed information system (*DIS*), where $S_i = (X_i, A_i, V_i)$ for any $i \in I$. Association rules are extracted from all information systems in *DIS* assuming some threshold *ms* for minimum support and *mc* for minimum confidence. If a classification rule

$$r_1 = [[(a, w_a) * (b, w_b) * ... * (c, w_c)] \Rightarrow (d, v_1)] \qquad [ms,\ mc]$$

where $d \in A$ is a semi-stable attribute in S_i is extracted from S_i and supported by a large number of sites in *DIS*, it is called a *trade*, or a *common rule* for *DIS*. If r_1 is supported either only by S_i or by a very small number of sites in *DIS* and at the same time rule

$$r_2 = [[(a, w_a) * (b, w_b) * ... * (c, w_c)] \Rightarrow (d, v_2)] \qquad [ms,\ mc]$$

where $v_1 \neq v_2$, is supported by a large number of sites in *DIS* then either the attribute d has nonstandard interpretation in S_i or r_1 is an exception rule (see [12]). To verify if r_1 is an exception rule, we have to check if there is a reasonably large number of objects having property v_1 which are described by $[(a, w_a) * (b, w_b) * ... * (c, w_c)]$ at minimum one of the sites supporting r_2. Otherwise, we may assume that d has a non-standard interpretation at S_i.

Assume that we do not know if the interpretation of a semi-stable attribute $d \in A_i$ at $S_i = (X_i, A_i, V_i)$ is standard. We have discussed the case when attribute d is defined in S_i in terms of a semi-stable attribute b, which has standard interpretation in S_i. Namely, if we identify an object in S_i which description contradicts information about attributes d, b stored in S_i ontology, then attribute d has non-standard interpretation in S_i. However, it can happen that we do not have any information about the interpretation of b in S_i. We can either look for a definition of d in terms of another semi-stable attribute in S_i or look for a definition of d in terms of attribute b at another site of DIS. If we cannot find any attribute other than d, which is semi-stable and has non-standard interpretation in S_i, then we contact another site.

Let A_i^{ss} be the set of all semi-stable attributes in S_i. We search for a site S_j such that $d \in A_j$ and $A_i^{ss} \cap A_j^{ss} \neq \emptyset$. Let I_d be the collection of all such sites and $b \in A_i^{ss} \cap A_j^{ss} \neq \emptyset$, where $j \in I_d$.

In the case where the interpretation of both attributes d, b is standard, if $j \in I_d$ satisfies the property that any $b \in A_i^{ss} \cap A_j^{ss} \neq \emptyset$ has standard interpretation in S_j, then the site j is not considered. Thus, we need to observe another site from I_d.

Now let us assume that $I_{\{d,b\}} = \{j \in I_d : b \in A_j \cap A_i^{ss}\}$. We extract rules at sites $I_{\{d,b\}}$ describing d and having b on their left side. Either classification rules discovered at site S_i will support rules discovered at the large number of sites in $I_{\{d,b\}}$ or conflict many of them. We claim that the interpretation of attribute d is standard in the first case. In the second case, it is non-standard.

This strategy was implemented and tested on a DIS consisting of thirteen sites representing thirteen Insurance Company *Datasets*, with a total of $5,000$ tuples in all DIS sites. Semi-stable attributes with non-standard interpretation have been detected and used jointly with flexible attributes in action rules mining. The confidence of action rules was usually higher than the confidence of the corresponding action rules based only on flexible attributes.

7 Heuristic Strategy for the Lowest Cost Reclassification of Objects

Let us assume that we wish to reclassify as many objects as possible in the system S_i, which is a part of DIS, from the class described by value k_1 of the attribute d to the class k_2. The reclassification $k_1 \to k_2$ jointly with its cost $\rho_{S_i}(k_1, k_2)$ is seen as the information stored in the initial node n_0 of the search graph built from nodes generated recursively by feasible action rules taken initially from $R_{S_i}[(d, k_1 \to k_2)]$. For instance, the rule

$$r = [(b_1, v_1 \to w_1) \wedge (b_2, v_2 \to w_2) \wedge ... \wedge (b_p, v_p \to w_p)](x) \Rightarrow (d, k_1 \to k_2)(x)$$

applied to the node $n_0 = \{[k_1 \to k_2, \rho_{S_i}(k_1, k_2)]\}$ generates the node

$$n_1 = \{[v_1 \to w_1, \rho_{S_i}(v_1, w_1)], [v_2 \to w_2, \rho_{S_i}(v_2, w_2)], ..., [v_p \to w_p, \rho_{S_i}(v_p, w_p)]\},$$

and from n_1 we can generate the node

$$n_2 = \{[v_1 \to w_1, \rho_{S_i}(v_1, w_1)], [v_2 \to w_2, \rho_{S_i}(v_2, w_2)], ..., [v_{j1} \to w_{j1}, \rho_{S_i}(v_{j1}, w_{j1})], [v_{j2} \to w_{j2}, \rho_{S_i}(v_{j2}, w_{j2})], ..., [v_{jq} \to w_{jq}, \rho_{S_i}(v_{jq}, w_{jq})], ..., [v_p \to w_p, \rho_{S_i}(v_p, w_p)]\}$$

assuming that the action rule

$$r_1 = [(b_{j1}, v_{j1} \to w_{j1}) \wedge (b_{j2}, v_{j2} \to w_{j2}) \wedge ... \wedge (b_{jq}, v_{jq} \to w_{jq})](y) \Rightarrow (b_j, v_j \to w_j)(y)$$

from $R_{S_m}[(b_j, v_j \to w_j)]$ is applied to n_1. /see Section 4/

This information can be written equivalently as: $r(n_0) = n_1$, $r_1(n_1) = n_2$, $[r_1 \circ r](n_0) = n_2$. Also, we should notice here that r_1 is extracted from S_m and $Sup_{S_m}(r_1) \subseteq X_m$ whereas r is extracted from S_i and $Sup_{S_i}(r) \subseteq X_i$.

By $Sup_{S_i}(r)$ we mean the domain of action rule r (set of objects in S_i supporting r).

The search graph can be seen as a directed graph G which is dynamically built by applying action rules to its nodes. The initial node n_0 of the graph G contains information coming from the user, associated with the system S_i, about what objects in X_i he would like to reclassify and how and what is his current cost of this reclassification. Any other node n in G shows an alternative way to achieve the same reclassification with a cost that is lower than the cost assigned to all nodes which are preceding n in G. Clearly, the confidence of action rules labelling the path from the initial node to the node n is as much important as the information about reclassification and its cost stored in node n. Information from what sites in DIS these action rules have been extracted and how similar the objects at these sites are to the objects in S_i is important as well.

Information stored at the node
$\{[v_1 \to w_1, \rho_{S_i}(v_1, w_1)], [v_2 \to w_2, \rho_{S_i}(v_2, w_2)], ..., [v_p \to w_p, \rho_{S_i}(v_p, w_p)]\}$
says that by reclassifying any object x supported by rule r from the class v_i to the class w_i, for any $i \leq p$, we also reclassify that object from the class k_1 to k_2. The confidence in the reclassification of x supported by node $\{[v_1 \to w_1, \rho_{S_i}(v_1, w_1)], [v_2 \to w_2, \rho_{S_i}(v_2, w_2)], ..., [v_p \to w_p, \rho_{S_i}(v_p, w_p)]\}$ is the same as the confidence of the rule r.

Before we give a heuristic strategy for identifying a node in G, built for a desired reclassification of objects in S_i, with a cost possibly the lowest among all the nodes reachable from the node n_0, we have to introduce additional notations.

So, assume that N is the set of nodes in our dynamically built directed graph G and n_0 is its initial node. For any node $n \in N$, by $f(n) = (Y_n, \{[v_{n,j} \to w_{n,j}, \rho_{S_i}(v_{n,j}, w_{n,j})]\}_{j \in I_n})$ we mean its domain, the reclassification steps related to objects in X_i, and their cost, all assigned by *reclassification function* f to the node n, where $Y_n \subseteq X_i$ /Graph G is built for the client site S_i/.

Let us assume that $f(n) = (Y_n, \{[v_{n,k} \to w_{n,k}, \rho_{S_i}(v_{n,k}, w_{n,k})]\}_{k \in I_n})$. We say that action rule r, extracted from S_i, is applicable to the node n if:

- $Y_n \cap Sup_{S_i}(r) \neq \emptyset$,
- $(\exists k \in I_n)[r \in R_{S_i}[v_{n,k} \to w_{n,k}]]$. /see Section 4 for definition of $R_{S_i}[...]$/

Similarly, we say that action rule r, extracted from S_m, is applicable to the node n if:

- $(\exists x \in Y_n)(\exists y \in Sup_{S_m}(r))[\rho(x,y) \leq \lambda]$, /$\rho(x,y)$ is the similarity relation between x, y (see Section 4 for its definition) and λ is a given similarity threshold/
- $(\exists k \in I_n)[r \in R_{S_m}[v_{n,k} \to w_{n,k}]]$.

It has to be noticed that reclassification of objects assigned to a node of G may refer to attributes which are not necessarily attributes listed in S_i. In this case, the user associated with S_i has to decide what is the cost of such a reclassification at his site, since such a cost may differ from site to site.

Now, let $RA(n)$ be the set of all action rules applicable to the node n. We say that the node n is completely covered by action rules from $RA(n)$ if $X_n = \bigcup\{Sup_{S_i}(r) : r \in RA(n)\}$. Otherwise, we say that n is partially covered by action rules.

What about calculating the domain Y_n of node n in the graph G constructed for the system S_i? The reclassification $(d, k_1 \to k_2)$ jointly with its cost $\rho_{S_i}(k_1, k_2)$ is stored in the initial node n_0 of the search graph G. Its domain Y_0 is defined as the set-theoretical union of domains of feasible action rules in $R_{S_i}[(d, k_1 \to k_2)]$ applied to X_i. This domain still can be extended by any object $x \in X_i$ if the following condition holds:

$$(\exists m)(\exists r \in R_{S_m}[k_1 \to k_2])(\exists y \in Sup_{S_m}(r))[\rho(x,y) \leq \lambda].$$

Each rule applied to the node n_0 generates a new node in G which domain is calculated in a similar way to n_0. To be more precise, assume that n is such a node and $f(n) = (Y_n, \{[v_{n,k} \to w_{n,k}, \rho_{S_i}(v_{n,k}, w_{n,k})]\}_{k \in I_n})$. Its domain Y_n is defined as the set-theoretical union of domains of feasible action rules in $\bigcup\{R_{S_i}[v_{n,k} \to w_{n,k}] : k \in I_n\}$ applied to X_i. Similarly to n_0, this domain still can be extended by any object $x \in X_i$ if the following condition holds:

$(\exists m)(\exists k \in I_n)(\exists r \in R_{S_m}[v_{n,k} \rightarrow w_{n,k}])(\exists y \in Sup_{S_m}(r))[\rho(x,y) \leq \lambda]$.

Clearly, for all other nodes, dynamically generated in G, the definition of their domains is the same as the one above.

Property 1. An object x can be reclassified according to the data stored in node n, only if x belongs to the domain of each node along the path from the node n_0 to n.

Property 2. Assume that x can be reclassified according to the data stored in node n and $f(n) = (Y_n, \{[v_{n,k} \rightarrow w_{n,k}, \rho_{S_i}(v_{n,k}, w_{n,k})]\}_{k \in I_n})$.

The cost $Cost_{k_1 \rightarrow k_2}(n, x)$ assigned to the node n in reclassifying x from k_1 to k_2 is equal to $\sum\{\rho_{S_i}(v_{n,k}, w_{n,k}) : k \in I_n\}$.

Property 3. Assume that x can be reclassified according to the data stored in node n and the action rules r, r_1, r_2,..., r_j are labelling the edges along the path from the node n_0 to n.

The confidence $Conf_{k_1 \rightarrow k_2}(n, x)$ assigned to the node n in reclassifying x from k_1 to k_2 is equal to $Conf[r_j \circ ... \circ r_2 \circ r_1 \circ r]$ /see Section 4/.

Property 4. If node n_{j2} is a successor of the node n_{j1}, then $Conf_{k_1 \rightarrow k_2}(n_{j2}, x) \leq Conf_{k_1 \rightarrow k_2}(n_{j1}, x)$.

Property 5. If a node n_{j2} is a successor of the node n_{j1}, then $Cost_{k_1 \rightarrow k_2}(n_{j2}, x) \leq Cost_{k_1 \rightarrow k_2}(n_{j1}, x)$.

Let us assume that we wish to reclassify as many objects as possible in the system S_i, which is a part of DIS, from the class described by value k_1 of the attribute d to the class k_2. We also assume that R is the set of all action rules extracted either from the system S_i or any of its remote sites in DIS. The reclassification $(d, k_1 \rightarrow k_2)$ jointly with its cost $\rho_{S_i}(k_1, k_2)$ represent the information stored in the initial node n_0 of the search graph G. By λ_{Conf} we mean the minimal confidence in a reclassification which is acceptable to the user and by λ_{Cost}, the maximal cost the user is willing to pay for that reclassification.

The algorithm **Build-and-Search** generates for each object x in S_i, the reclassification rules satisfying thresholds for minimal confidence and maximal cost.

Algorithm Build-and-Search$(R, x, \lambda_{Conf}, \lambda_{Cost}, n, m)$;

Input Set of action rules R,
Object x which the user would like to reclassify,
Threshold value λ_{Conf} for minimal confidence,
Threshold value λ_{Cost} for maximal cost,
Node n of a graph G.

Output Node m representing an acceptable reclassification of objects from S_i.

begin

if $Cost_{k_1 \to k_2}(n, x) > \lambda_{Cost}$, **then**
generate all successors of n using rules from R;
while n_1 is a successor of n **do**
  if $Conf_{k_1 \to k_2}(n_1, x) < \lambda_{Conf}$ **then stop**
  else
    if $Cost_{k_1 \to k_2}(n_1, x) \leq \lambda_{Cost}$ **then** $Output[n_1]$
    else Build-and-Search$(R, x, \lambda_{Conf}, \lambda_{Cost}, n_1, m)$
end

Now, calling the procedure **Build-and-Search**$(R, x, \lambda_{Conf}, \lambda_{Cost}, n_0, m)$, we get the reclassification rules for x satisfying thresholds for minimal confidence and maximal cost.

8 Conclusions

We should notice that the recursive call, in the current version of
Build-and-Search$(R, x, \lambda_{Conf}, \lambda_{Cost}, n, m)$
procedure, stops on the first node n which satisfies both thresholds: λ_{Conf} for minimal confidence and λ_{Cost} for maximal cost. Clearly, this strategy can be enhanced by allowing recursive calls on any node n when both thresholds are satisfied by n and forcing recursive calls to stop on the first node n_1 succeeding n, if only $Cost_{k_1 \to k_2}(n_1, x) \leq \lambda_{Cost}$ and $Conf_{k_1 \to k_2}(n_1, x) < \lambda_{Conf}$. Then, the recursive procedure should terminate not on n_1 but on the node which is its direct predecessor.

The above procedure can be also enhanced by searching for reclassification rules not for a single object but simultaneously for a group of objects. This can be done by recursively splitting the domain Y_n of a node n by the action rules from R applied to n. The split is done in a natural way, since each action rule r applied to n has its supporting domain $Sup_{S_i}(r) \cap Y_n$.

Finally, each node n in a graph G can be reached by following not necessarily only one path from the initial node n_0 to n. Each path has its own cost and therefore it is not wise to chose the first acceptable path to compute the cost assigned to the node n. We should rather build all path from the node n_0 to n and next compute the cost assigned to n. Also, object x can belong to many domains, since it is not true that $Y_{n1} \cap Y_{n2} = \emptyset$. So, in order to solve the problem of finding optimal reclassification rules for a subset of X_i, we have to search the whole graph G.

The last remark concerns the identification problem: which semi-stable attributes are flexible? Clearly, by increasing the number of flexible attributes in S_i and other systems in DIS, we also enlarge the set R and the same we change the contents of nodes in the graph G and possibly even increase the number of edges in G. This may change entirely the output of **Build-and-Search**$(R, x, \lambda_{Conf}, \lambda_{Cost}, n_0, m)$.

References

1. Chmielewski M. R., Grzymala-Busse J. W., Peterson N. W., Than S., (1993), The Rule Induction System LERS - a version for personal computers in *Foundations of Computing and Decision Sciences*, Vol. 18, No. 3-4, 1993, Institute of Computing Science, Technical University of Poznan, Poland, 181-212
2. Fensel, D., (1998), *Ontologies: a silver bullet for knowledge management and electronic commerce*, Springer-Verlag, 1998
3. Pawlak Z., (1985), *Rough Ssets and decision tables*, in Lecture Notes in Computer Science 208, Springer-Verlag, 1985, 186-196.
4. Pawlak, Z., (1991), *Rough Sets: Theoretical aspects of reasoning about data*, Kluwer Academic Publisher, 1991.
5. Raś, Z., Dardzińska, A., "Handling semantic inconsistencies in query answering based on distributed knowledge mining", in *Foundations of Intelligent Systems*, Proceedings of ISMIS'02 Symposium, LNCS/LNAI, No. 2366, Springer-Verlag, 2002, 66-74
6. Raś, Z., Wieczorkowska, A., (2000), Action Rules: how to increase profit of a company, in *Principles of Data Mining and Knowledge Discovery*, (Eds. D.A. Zighed, J. Komorowski, J. Zytkow), Proceedings of PKDD'00, Lyon, France, LNCS/LNAI, No. 1910, Springer-Verlag, 2000, 587-592
7. Raś, Z.W., Tsay, L.-S., (2003), Discovering Extended Action-Rules (System DEAR), in *Intelligent Information Systems 2003*, Proceedings of the IIS'2003 Symposium, Zakopane, Poland, Advances in Soft Computing, Springer-Verlag, 2003, 293-300
8. Raś, Z.W., Tzacheva, A., (2003), Discovering semantic incosistencies to improve action rules mining, in *Intelligent Information Systems 2003*, Advances in Soft Computing , Proceedings of the IIS'2003 Symposium, Zakopane, Poland, Springer-Verlag, 2003, 301-310
9. Raś, Z., Gupta, S., (2002), Global action rules in distributed knowledge systems, in *Fundamenta Informaticae Journal*, IOS Press, Vol. 51, No. 1-2, 2002, 175-184
10. Skowron A., Grzymala-Busse J., (1991), From the Rough Set Theory to the Evidence Theory, in *ICS Research Reports*, 8/91, Warsaw University of Technology, October, 1991
11. Sowa, J.F., (2000), Ontology, Metadata and Semiotics, in *Conceptual Structures: Logical, Linguistic, and Computational Issues*, B. Ganter, G.W. Mineau (Eds), LNAI 1867, Springer-Verlag, 2000, 55-81
12. Suzuki, E., Kodratoff, Y., (1998), Discovery of surprising exception rules based on intensity of implication, in *Proc. of the Second Pacific-Asia Conference on Knowledge Discovery and Data mining (PAKDD)*, 1998