

# ARAS: Action rules discovery based on agglomerative strategy

Zbigniew W. Ras<sup>1,3</sup> and Elżbieta Wyrzykowska<sup>2</sup>

<sup>1</sup> Univ. of North Carolina, Dept. of Comp. Science, Charlotte, N.C. 28223, USA;  
e-mail: ras@uncc.edu

<sup>2</sup> Univ. of Information Technology and Management, ul. Newelska, Warsaw, Poland;  
email: ewyrzyko@wit.edu.pl

<sup>3</sup> Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008  
Warsaw, Poland; e-mail: ras@pjwstk.edu.pl

**Abstract.** Action rules can be seen as logical terms describing knowledge about possible actions associated with objects which is hidden in a decision system. Classical strategy for discovering them from a database requires prior extraction of classification rules which next are evaluated pair by pair with a goal to build a strategy of action based on condition features in order to get a desired effect on a decision feature. An actionable strategy is represented as a term  $r = [(\omega) \wedge (\alpha \rightarrow \beta)] \Rightarrow [\phi \rightarrow \psi]$ , where  $\omega$ ,  $\alpha$ ,  $\beta$ ,  $\phi$ , and  $\psi$  are descriptions of objects or events. The term  $r$  states that when the fixed condition  $\omega$  is satisfied and the changeable behavior  $(\alpha \rightarrow \beta)$  occurs in objects represented as tuples from a database so does the expectation  $(\phi \rightarrow \psi)$ . This paper proposes a new strategy, called *ARAS*, for constructing action rules with the main module resembling *LEERS* [5]. *ARAS* system is more simple than *DEAR* and its time complexity is also lower.

## 1 Introduction

Finding useful rules is an important task of a knowledge discovery process. Most researchers focus on techniques for generating patterns from a data set such as classification rules, association rules...etc. They assume that it is user's responsibility to analyze these patterns in order to infer solutions for specific problems within a given domain. The classical knowledge discovery algorithms have the potential to identify enormous number of significant patterns from data. Therefore, people are overwhelmed by a large number of uninteresting patterns and it is very difficult for a human being to analyze them in order to form timely solutions. Therefore, a significant need exists for a new generation of techniques and tools with the ability to assist users in analyzing a large number of rules for a useful knowledge.

There are two aspects of interestingness of rules that have been studied in data mining literature, objective and subjective measures [1], [7]. Objective measures are data-driven and domain-independent. Generally, they evaluate the rules

based on their quality and similarity between them. Subjective measures, including unexpectedness, novelty and actionability, are user-driven and domain-dependent.

For example, classification rules found from a bank's data are very useful to describe who is a good client (whom to offer some additional services) and who is a bad client (whom to watch carefully to minimize the bank losses). However, if bank managers hope to improve their understanding of customers and seek specific actions to improve services, mere classification rules will not be convincing for them. Therefore, we can use the classification rules to build a strategy of action based on condition features in order to get a desired effect on a decision feature [8]. Going back to the bank example, the strategy of action would consist of modifying some condition features in order to improve their understanding of customers and then improve services.

Action rules, introduced in [8] and investigated further in [11], [13], [10], are constructed from certain pairs of classification rules. Interventions, defined in [4], are conceptually very similar to action rules.

The process of constructing action rules from pairs of classification rules is not only unnecessarily expensive but also gives too much freedom in constructing their classification parts. In [10] it was shown that action rules do not have to be built from pairs of classification rules and that single classification rules are sufficient to achieve the same goal. However, the paper only proposed a theoretical lattice-theory type framework without giving any detailed algorithm for action rules construction. In this paper we propose a very simple *LERS*-type algorithm for constructing action rules from a single classification rule. *LERS* is a classical example of a bottom-up strategy which constructs rules with a conditional part of the length  $k+1$  after all rules with a conditional part of the length  $k$  have been constructed. Relations representing rules produced by *LERS* are marked. System *ARAS* assumes that *LERS* is used to extract classification rules. This way *ARAS* instead of verifying the validity of certain relations only has to check if these relations are marked by *LERS*. The same, by using *LERS* as the pre-processing module for *ARAS*, the overall complexity of the algorithm will decrease.

## 2 Action Rules

In paper [8], the notion of an action rule was introduced. The main idea was to generate, from a database, special type of rules which basically form a hint to users showing a way to re-classify objects with respect to values of some distinguished attribute (called a decision attribute).

We start with a definition of an information system given in [6].

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

- $U$  is a nonempty, finite set of objects (object identifiers),

- $A$  is a nonempty, finite set of attributes (partial functions) i.e.  $a : U \rightarrow V_a$  for  $a \in A$ , where  $V_a$  is called the domain of  $a$ .

We often write  $(a, v)$  instead of  $v$ , assuming that  $v \in V_a$ . Information systems can be used to model decision tables. In any decision table together with the set of attributes a partition of that set into conditions and decisions is given. Additionally, we assume that the set of conditions is partitioned into stable and flexible conditions [8]. Attribute  $a \in A$  is called stable for the set  $U$ , if its values assigned to objects from  $U$  can not change in time. Otherwise, it is called flexible. "Date of Birth" is an example of a stable attribute. "Interest rate" on any customer account is an example of a flexible attribute. For simplicity reason, we will consider decision tables with only one decision. We adopt the following definition of a decision table:

By a decision table we mean an information system  $S = (U, A_1 \cup A_2 \cup \{d\})$ , where  $d \notin A_1 \cup A_2$  is a distinguished attribute called decision. Additionally, it is assumed that  $d$  is a total function. The elements of  $A_1$  are called stable attributes, whereas the elements of  $A_2 \cup \{d\}$  are called flexible. Our goal is to suggest changes in values of attributes in  $A_2$  for some objects from  $U$  so the values of the attribute  $d$  for these objects may change as well. A formal expression describing such a property is called an action rule [8], [11].

<i>Stable</i>	<i>Flexible</i>	<i>Stable</i>	<i>Flexible</i>	<i>Stable</i>	<i>Flexible</i>	<i>Decision</i>
$A$	$B$	$C$	$E$	$G$	$H$	$D$
$a_1$	$b_1$	$c_1$	$e_1$			$d_1$
$a_1$	$b_2$			$g_2$	$h_2$	$d_2$

**Table 1.** Two classification rules extracted from S

To construct an action rule [11], let us assume that two classification rules, each one referring to a different decision class, are considered. We assume here that these two rules have to be equal on their stable attributes, if they are both defined on them. We use Table 1 to clarify the process of action rule construction. Here, "Stable" means stable attribute and "Flexible" means flexible one. In a standard representation, these two classification rules have a form:

$$r_1 = [(a_1 \wedge b_1 \wedge c_1 \wedge e_1) \rightarrow d_1], r_2 = [(a_1 \wedge b_2 \wedge g_2 \wedge h_2) \rightarrow d_2].$$

Assume now that object  $x$  supports rule  $r_1$  which means that it is classified as  $d_1$ . In order to reclassify  $x$  to a class  $d_2$ , we need to change not only the value of  $B$  from  $b_1$  to  $b_2$  but also to assume that  $G(x) = g_2$  and that the value  $H$  for object  $x$  has to be changed to  $h_2$ . This is the meaning of the  $(r_1, r_2)$ -action rule defined by the expression below:

$$r = [(a_1 \wedge g_2 \wedge (B, b_1 \rightarrow b_2) \wedge (H, \rightarrow h_2)) \rightarrow (D, d_1 \rightarrow d_2)].$$

The term  $[a_1 \wedge g_2]$  is called the header of the action rule. Assume now that by  $Sup(t)$  we mean the number of tuples having property  $t$ . By the support of  $(r_1, r_2)$ -action rule (given above) we mean:  $Sup[a_1 \wedge b_1 \wedge g_2 \wedge d_1]$ . By the confidence  $Conf(r)$  of  $(r_1, r_2)$ -action rule  $r$  (given above) we mean (see [11], [12]):

$$[Sup[a_1 \wedge b_1 \wedge g_2 \wedge d_1]/Sup[a_1 \wedge b_1 \wedge g_2]] \cdot [Sup[a_1 \wedge b_2 \wedge c_1 \wedge d_2]/Sup[a_1 \wedge b_2 \wedge c_1]].$$

Assume now that  $S = (U, A_1 \cup A_2 \cup \{d\})$  is decision system, where  $A_1 = \{a, b\}$  are stable attributes,  $A_2 = \{c, e, f\}$  are flexible attributes, and  $d$  is the decision. For a generality reason, we take an incomplete decision system. It is represented as Table 2. Our goal is to re-classify objects in  $S$  from  $(d, 2)$  to  $(d, 1)$ . Additionally, we assume that  $Dom(a) = \{2, 3, 10\}$ ,  $Dom(b) = \{2, 3, 4, 5\}$ , and the null value is represented as  $-1$ . We will follow optimistic approach in the process of action rules discovery, which means that the Null value is interpreted as the disjunction of all possible attribute values in the corresponding domain.

<i>Stable</i>	<i>Stable</i>	<i>Flexible</i>	<i>Flexible</i>	<i>Flexible</i>	<i>Decision</i>
<i>a</i>	<i>b</i>	<i>c</i>	<i>e</i>	<i>f</i>	<i>d</i>
2	-1	-1	7	8	1
2	5	4	6	8	1
-1	-1	4	9	4	2
10	4	5	8	7	2
2	2	5	-1	9	3
2	2	4	7	6	3
-1	2	4	7	-1	2
2	-1	-1	6	8	3
3	2	4	6	8	2
3	3	5	7	4	2
3	3	5	6	2	3
2	5	4	9	4	1

**Table 2.** Incomplete Decision System  $S$

Now, we present the preprocessing step for action rules discovery. We start with our incomplete decision system  $S$  as the root of the Reduction Tree. The next step is to split  $S$  into sub-tables taking an attribute with the minimal number of distinct values as the splitting one. In our example, we chose attribute  $a$ . Because the 3rd and the 7th tuple in  $S$  contain null values in column  $a$ , we move them both to all three newly created sub-tables. This process is recursively continued for all stable attributes. Sub-tables corresponding to outgoing edges from the root node which are labelled by  $a = 10$ ,  $a = 3$  are removed because

they do not contain decision value 1. Any remaining node in the resulting tree can be used for discovering action rules. Clearly, if node  $n$  is used to construct action rules, then its children are not used for that purpose.

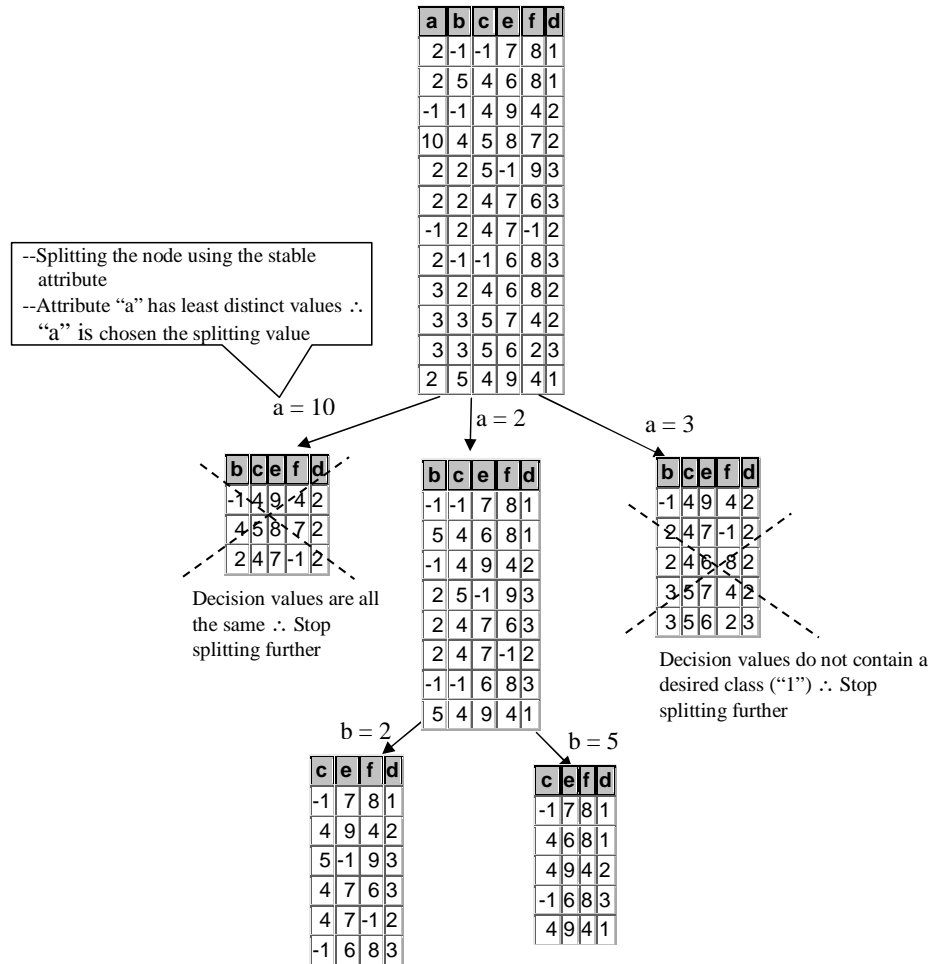


Fig. 1. Table Reduction Process

### 3 ARAS: algorithm for discovering action rules

This section covers only complete information systems. For an incomplete information system, we can use *ERID* [2] to discover classification rules. Their syntax is the same as the syntax of rules discovered from a complete system.

Let us assume that  $S = (U, A_1 \cup A_2 \cup \{d\})$  is a complete decision system, where  $d \notin A_1 \cup A_2$  is a distinguished attribute called the decision. The elements of  $A_1$  are stable conditions, whereas the elements of  $A_2 \cup \{d\}$  are flexible. Assume that  $d_1 \in V_d$  and  $x \in U$ . We say that  $x$  is a  $d_1$ -object if  $d(x) = d_1$ . We also assume that  $\{a_1, a_2, \dots, a_p\} \subseteq A_1$ ,  $\{a_{p+1}, a_{p+2}, \dots, a_n\} = A_1 - \{a_1, a_2, \dots, a_p\}$ ,  $\{b_1, b_2, \dots, b_q\} \subseteq A_2$ ,  $a_{[i,j]}$  denotes a value of attribute  $a_i$ ,  $b_{[i,j]}$  denotes a value of attribute  $b_i$ , for any  $i, j$  and that

$$r = [[a_{[1,1]} \wedge a_{[2,1]} \wedge \dots \wedge a_{[p,1]} \wedge b_{[1,1]} \wedge b_{[2,1]} \wedge \dots \wedge b_{[q,1]}] \longrightarrow d_1]$$

is a classification rule extracted from  $S$  supporting some  $d_1$ -objects in  $S$ . Class  $d_1$  is a preferable class and our goal is to reclassify  $d_2$ -objects into  $d_1$  class, where  $d_2 \in V_d$ .

By an action rule schema  $r_{[d_2 \rightarrow d_1]}$  associated with  $r$  and the above reclassification task  $(d, d_2 \rightarrow d_1)$  we mean the following expression:

$$r_{[d_2 \rightarrow d_1]} = [[a_{[1,1]} \wedge a_{[2,1]} \wedge \dots \wedge a_{[p,1]} \wedge (b_1, \longrightarrow b_{[1,1]}) \wedge (b_2, \longrightarrow b_{[2,1]}) \wedge \dots \wedge (b_q, \longrightarrow b_{[q,1]})] \longrightarrow (d, d_2 \longrightarrow d_1)]$$

In a similar way, by an action rule schema  $r_{[\rightarrow d_1]}$  associated with  $r$  and the reclassification task  $(d, \rightarrow d_1)$  we mean the following expression:

$$r_{[\rightarrow d_1]} = [[a_{[1,1]} \wedge a_{[2,1]} \wedge \dots \wedge a_{[p,1]} \wedge (b_1, \longrightarrow b_{[1,1]}) \wedge (b_2, \longrightarrow b_{[2,1]}) \wedge \dots \wedge (b_q, \longrightarrow b_{[q,1]})] \longrightarrow (d, \longrightarrow d_1)]$$

The term  $[a_{[1,1]} \wedge a_{[2,1]} \wedge \dots \wedge a_{[p,1]}]$ , built from values of stable attributes, is called the header of  $r_{[d_2 \rightarrow d_1]}$  and its values can not be changed. It is denoted by  $h[r_{[d_2 \rightarrow d_1]}]$ .

The support set of the action rule schema  $r_{[d_2 \rightarrow d_1]}$  is defined as  $Sup(r_{[d_2 \rightarrow d_1]}) = \{x \in U : (a_1(x) = a_{[1,1]}) \wedge (a_2(x) = a_{[2,1]}) \wedge \dots \wedge (a_p(x) = a_{[p,1]}) \wedge (d(x) = d_2)\}$ .

Now, we outline *ARAS* strategy for generating the set **AR** of Action Rules from the action rule schema  $r_{[d_2 \rightarrow d_1]}$ .

Assume that:

- $V_{a_{p+1}} = \{a_{[p+1,1]}, a_{[p+1,2]}, \dots, a_{[p+1,J(1)]}\}$
- $V_{a_{p+2}} = \{a_{[p+2,1]}, a_{[p+2,2]}, \dots, a_{[p+2,J(2)]}\}$
- ...
- $V_{a_{p+n}} = \{a_{[p+n,1]}, a_{[p+n,2]}, \dots, a_{[p+n,J(n)]}\}$
- $V_{b_1} = \{b_{[1,1]}, b_{[1,2]}, \dots, b_{[1,J(n+1)]}\}$
- $V_{b_2} = \{b_{[2,1]}, b_{[2,2]}, \dots, b_{[2,J(n+2)]}\}$
- .....
- $V_{b_q} = \{b_{[q,1]}, b_{[q,2]}, \dots, b_{[q,J(n+q)]}\}$

To simplify the presentation of the algorithm we assume that:

- $c_k = a_{p+k}$  and  $c_{[k,i]} = a_{[p+k,i]}$ , for  $1 \leq i \leq J(k)$ ,  $1 \leq k \leq n$ ,
- $c_{n+m} = b_m$  and  $c_{[n+m,i]} = b_{[m,i]}$ , for  $1 \leq i \leq J(n+m)$ ,  $1 \leq m \leq q$ .

For simplicity reason, we use  $U_{[r,d_2]}$  to denote  $Sup(r_{[d_2 \rightarrow d_1]})$ . We assume that the term  $c_{[i_1,j_1]} \wedge c_{[i_2,j_2]} \wedge \dots \wedge c_{[i_r,j_r]}$  is denoted by  $[c_{[i_k,j_k]}]_{k \in \{1,2,\dots,r\}}$ , where all  $i_1, i_2, \dots, i_r$  are distinct integers and  $j_p \leq J(i_p)$ ,  $1 \leq p \leq r$ . Following **LEERS** notation [5], we also assume that  $t^*$  denotes the set of all objects in  $S$  having property  $t$ .

**Algorithm**  $AR(r, d_2)$

```

i:=1
while i ≤ n + q do
  begin
    j:=2; m:=1
    while j < J(i) do
      begin
        if [h[r[d2→d1]]] ∧ c(i,j)* ⊆ U[r,d2]] ∧ ci ∈ A2 then
          begin
            mark[c(i,j)];
            output Action Rule
              [[h[r[d2→d1]]] ∧ (ci, c(i,j) → c(i,1))] → [d, d2 → d1]]
            end
          if [h[r[d2→d1]]] ∧ c(i,j)* ⊆ U[r,d2]] ∧ ci ∈ A1 then
            begin
              mark[c(i,j)];
              output Action Rule
                [[h[r[d2→d1]]] ∧ (ci, c(i,j))] → [d, d2 → d1]]
              end
            j:=j+1
          end
        end
      Ik := {ik};
      (where ik - index randomly chosen from {2, 3, ..., q + n}).
      for all jk ≤ J(ik) do [c(ik,jk)]ik ∈ Ik := c(ik, jk);
      for all i, j such that both sets [c(ik,jk)]ik ∈ Ik, c(i,j) are not marked and
      i ∈ Ik
      do
        begin
          if [[h[r[d2→d1]]] ∧ [c(ik,jk)]ik ∈ Ik ∧ c(i,j)]* ⊆ U[r,d2]] ∧ ci ∈ A2 then
            begin
              mark [[c(ik,jk)]ik ∈ Ik ∧ c(i,j)];
              output Action Rule
                [[h[r[d2→d1]]] ∧ [c(ik,jk)]ik ∈ Ik ∧ (ci, c(i,j) → c(i,1))] → [d, d2 → d1]]
              end
            if [[h[r[d2→d1]]] ∧ [c(ik,jk)]ik ∈ Ik ∧ c(i,j)]* ⊆ U[r,d2]] ∧ ci ∈ A1 then
              begin
                mark [[c(ik,jk)]ik ∈ Ik ∧ c(i,j)];
                output Action Rule
                  [[h[r[d2→d1]]] ∧ [c(ik,jk)]ik ∈ Ik ∧ (ci, c(i,j))] → [d, d2 → d1]]
              end
          end
        end
      end
    end
  end
end

```

```

end
else
  begin
     $I_k := I_k \cup \{i\}; [c_{(i_k, j_k)}]_{i_k \in I_k} := [c_{(i_k, j_k)}]_{i_k \in I_k} \wedge c_{(i, j)}$ 
  end

```

The complexity of *ARAS* is lower than the complexity of *DEAR* system discovering action rules. The justification here is quite simple. *DEAR* system [11] groups classification rules into clusters of non-conflicting rules and then takes all possible pairs of classification rules within each cluster and tries to build action rules from them. *ARAS* algorithm is treating each classification rule describing the target decision value as a seed and grabs other classification rules describing non-target decision values to form a cluster and then it builds decision rules automatically from them. Rules grabbed into a seed are only compared with that seed. So, the number of pairs of rules which have to be checked, in comparison to *DEAR* is greatly reduced. Another advantage of the current strategy is that the module generating action rules in *ARAS* can just check if a mark is assigned by *LERS* to the relation  $[h[r_{[d_2 \rightarrow d_1]}] \wedge c_{(i, j)}]^* \subseteq U_{[r, d_2]}$  instead of checking its validity.

The confidence of generated action rules depends on the number of remaining objects supporting them. Also, if  $Conf(r) \neq 1$ , then some objects in  $S$  satisfying the description  $[a_{1,1} \wedge a_{2,1} \wedge \dots \wedge a_{p,1} \wedge b_{1,1} \wedge b_{2,1} \wedge \dots \wedge b_{q,1}]$  are classified as  $d_2$ . According to the rule  $r_{[d_2 \rightarrow d_1]}$  they should be classified as  $d_1$  which means that the confidence of  $r_{[d_2 \rightarrow d_1]}$  will get also decreased.

If  $Sup(r_{[d_2 \rightarrow d_1]}) = \emptyset$ , then  $r_{[d_2 \rightarrow d_1]}$  can not be used for reclassification of objects. Similarly,  $r_{[\rightarrow d_1]}$  can not be used for reclassification, if  $Sup(r_{[d_2 \rightarrow d_1]}) = \emptyset$ , for each  $d_2$  where  $d_2 \neq d_1$ . From the point of view of actionability, such rules are not interesting.

Let  $Sup(r_{[\rightarrow d_1]}) = \bigcup \{Sup(r_{[d_2 \rightarrow d_1]}) : (d_2 \in V_d) \wedge (d_2 \neq d_1)\}$  and  $Sup(R_{[\rightarrow d_1]}) = \bigcup \{Sup(r_{[\rightarrow d_1]}) : r \in R(d_1)\}$ , where  $R(d_1)$  is the set of all classification rules extracted from  $S$  which are defining  $d_1$ . So,  $Sup(R_S) = \bigcup \{Sup(R_{[\rightarrow d_1]}) : d_1 \in V_d\}$  contains all objects in  $S$  which potentially can be reclassified.

Assume now that  $U(d_1) = \{x \in U : d(x) \neq d_1\}$ . Objects in the set  $B(d_1) = [U(d_1) - Sup(R_{[\rightarrow d_1]})]$  can not be reclassified to the class  $d_1$  and they are called  $d_1$ -resistant.

Let  $B(-d_1) = \bigcap \{B(d_i) : (d_i \in V_d) \wedge (d_i \neq d_1)\}$ . Clearly  $B(-d_1)$  represents the set of  $d_1$ -objects which can not be reclassified. They are called  $d_1$ -stable. Similarly, the set  $B_d = \bigcup \{B(-d_i) : d_i \in V_d\}$  represents objects in  $U$  which can not be reclassified to any decision class. All these objects are called  $d$ -stable. In order to show how to find them, the notion of a confidence of an action rule is needed.

Let  $r_{[d_2 \rightarrow d_1]}$ ,  $r'_{[d_2 \rightarrow d_3]}$  are two action rules extracted from  $S$ . We say that these rules are p-equivalent ( $\simeq$ ), if the condition given below holds for every  $b_i \in A_1 \cup A_2$ :

if  $r/b_i$ ,  $r'/b_i$  are both defined, then  $r/b_i = r'/b_i$ .

Now, we explain how to calculate the confidence of  $r_{[d_2 \rightarrow d_1]}$ . Let us take  $d_2$ -object  $x \in \text{Sup}(r_{[d_2 \rightarrow d_1]})$ . We say that  $x$  positively supports  $r_{[d_2 \rightarrow d_1]}$  if there is no classification rule  $r'$  extracted from  $S$  and describing  $d_3 \in V_d$ ,  $d_3 \neq d_1$ , which is p-equivalent to  $r$ , such that  $x \in \text{Sup}(r'_{[d_2 \rightarrow d_3]})$ . The corresponding subset of  $\text{Sup}(r_{[d_2 \rightarrow d_1]})$  is denoted by  $\text{Sup}^+(r_{[d_2 \rightarrow d_1]})$ . Otherwise, we say that  $x$  negatively supports  $r_{[d_2 \rightarrow d_1]}$ . The corresponding subset of  $\text{Sup}(r_{[d_2 \rightarrow d_1]})$  is denoted by  $\text{Sup}^-(r_{[d_2 \rightarrow d_1]})$ .

By the confidence of  $r_{[d_2 \rightarrow d_1]}$  in  $S$  we mean:

$$\text{Conf}(r_{[d_2 \rightarrow d_1]}) = [\text{card}[\text{Sup}^+(r_{[d_2 \rightarrow d_1]})] / \text{card}[\text{Sup}(r_{[d_2 \rightarrow d_1]})]] \cdot \text{conf}(r).$$

Now, if we assume that  $\text{Sup}^+(r_{[\rightarrow d_1]}) = \bigcup \{ \text{Sup}^+(r_{[d_2 \rightarrow d_1]}) : (d_2 \in V_d) \wedge (d_2 \neq d_1) \}$ , then by the confidence of  $r_{[\rightarrow d_1]}$  in  $S$  we mean:

$$\text{Conf}(r_{[\rightarrow d_1]}) = [\text{card}[\text{Sup}^+(r_{[\rightarrow d_1]})] / \text{card}[\text{Sup}(r_{[\rightarrow d_1]})]] \cdot \text{conf}(r).$$

It can be easily shown that the definition of support and confidence of action rules given in Section 3 is equivalent to the definition of support and confidence given in Section 2.

## 4 An example

Let us assume that the decision system  $S = (U, \{A_1 \cup A_2 \cup \{d\}\})$ , where  $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ , is represented by Table 3. A number of different methods can be used to extract rules in which the THEN part consists of the decision attribute  $d$  and the IF part consists of attributes belonging to  $A_1 \cup A_2$ . In our example, the set  $A_1 = \{a, b, c\}$  contains stable attributes and  $A_2 = \{e, f, g\}$  contains flexible attributes. System *LEERS* [5] is used to extract classification rules.

We are interested in reclassifying  $d_2$ -objects either to class  $d_1$  or  $d_3$ . Four certain classification rules describing  $d_1$ ,  $d_3$  can be discovered by *LEERS* in the decision system  $S$ . They are given below:

$$r1 = [b_1 \wedge c_1 \wedge f_2 \wedge g_1] \rightarrow d_1, r2 = [a_2 \wedge b_1 \wedge e_2 \wedge f_2] \rightarrow d_3, r3 = e_1 \rightarrow d_1, r4 = [b_1 \wedge g_2] \rightarrow d_3.$$

It can be shown that  $R_{[d, \rightarrow d_1]} = \{r1, r3\}$  and  $R_{[d, \rightarrow d_3]} = \{r2, r4\}$ . Action rule schemas associated with  $r1$ ,  $r2$ ,  $r3$ ,  $r4$  and the reclassification task either  $(d, d_2 \rightarrow d_1)$  or  $(d, d_2 \rightarrow d_3)$  are:

$$\begin{aligned} r1_{[d_2 \rightarrow d_1]} &= [b_1 \wedge c_1 \wedge (f, \rightarrow f_2) \wedge (g, \rightarrow g_1)] \rightarrow (d, d_2 \rightarrow d_1), \\ r2_{[d_2 \rightarrow d_3]} &= [a_2 \wedge b_1 \wedge (e, \rightarrow e_2) \wedge (f, \rightarrow f_2)] \rightarrow (d, d_2 \rightarrow d_3), \\ r3_{[d_2 \rightarrow d_1]} &= [(e, \rightarrow e_1)] \rightarrow (d, d_2 \rightarrow d_1), \\ r4_{[d_2 \rightarrow d_3]} &= [b_1 \wedge (g, \rightarrow g_2)] \rightarrow (d, d_2 \rightarrow d_3). \end{aligned}$$

**Table 3.** Decision System

$U$	$a$	$b$	$c$	$e$	$f$	$g$	$d$
$x_1$	$a_1$	$b_1$	$c_1$	$e_1$	$f_2$	$g_1$	$d_1$
$x_2$	$a_2$	$b_1$	$c_2$	$e_2$	$f_2$	$g_2$	$d_3$
$x_3$	$a_3$	$b_1$	$c_1$	$e_2$	$f_2$	$g_3$	$d_2$
$x_4$	$a_1$	$b_1$	$c_2$	$e_2$	$f_2$	$g_1$	$d_2$
$x_5$	$a_1$	$b_2$	$c_1$	$e_3$	$f_2$	$g_1$	$d_2$
$x_6$	$a_2$	$b_1$	$c_1$	$e_2$	$f_3$	$g_1$	$d_2$
$x_7$	$a_2$	$b_3$	$c_2$	$e_2$	$f_2$	$g_2$	$d_2$
$x_8$	$a_2$	$b_1$	$c_1$	$e_3$	$f_2$	$g_3$	$d_2$

We can also show that  $U_{[r1,d2]} = Sup(r1_{[d2 \rightarrow d1]}) = \{x_3, x_6, x_8\}$ ,  $U_{[r2,d2]} = Sup(r2_{[d2 \rightarrow d3]}) = \{x_6, x_8\}$ ,  $U_{[r3,d2]} = Sup(r3_{[d2 \rightarrow d1]}) = \{x_3, x_4, x_5, x_6, x_7, x_8\}$ ,  $U_{[r4,d2]} = Sup(r4_{[d2 \rightarrow d3]}) = \{x_3, x_4, x_6, x_8\}$ .

Following  $AR(r1, d_2)$  algorithm we get:  $[b_1 \wedge c_1 \wedge a_1]^* = \{x_1\} \not\subseteq U_{[r1,d2]}$ ,  $[b_1 \wedge c_1 \wedge a_2]^* = \{x_6, x_8\} \subseteq U_{[r1,d2]}$ ,  $[b_1 \wedge c_1 \wedge f_3]^* = \{x_6\} \subseteq U_{[r1,d2]}$ ,  $[b_1 \wedge c_1 \wedge g_2]^* = \{x_2, x_7\} \not\subseteq U_{[r1,d2]}$ ,  $[b_1 \wedge c_1 \wedge g_3]^* = \{x_3, x_8\} \subseteq U_{[r1,d2]}$ . It will generate two action rules:  $[b_1 \wedge c_1 \wedge (f, f_3 \rightarrow f_2) \wedge (g, \rightarrow g_1)] \rightarrow (d, d_2 \rightarrow d_1)$ ,  $[b_1 \wedge c_1 \wedge (f, \rightarrow f_2) \wedge (g, g_3 \rightarrow g_1)] \rightarrow (d, d_2 \rightarrow d_1)$ .

In a similar way we construct action rules from the remaining three action rule schemas.

The action rules discovery process, presented above, is called *ARAS* and it consists of two main modules. For its further clarification, we use another example which has no connection with Table 3. The first module extracts all classification rules from  $S$  following *LEERS strategy*. Assuming that  $d$  is the decision attribute and user is interested in re-classifying objects from its value  $d_1$  to  $d_2$ , we treat the rules defining  $d_1$  as seeds and build clusters around them. For instance, if  $A_1 = \{a, b, g\}$  are stable attributes,  $A_2 = \{c, e, h\}$  are flexible in  $S = (U, A_1 \cup A_2 \cup \{d\})$ , and  $r = [[a_1 \wedge b_1 \wedge c_1 \wedge e_1] \rightarrow d_1]$  is a classification rule in  $S$ , where  $V_a = \{a_1, a_2, a_3\}$ ,  $V_b = \{b_1, b_2, b_3\}$ ,  $V_c = \{c_1, c_2, c_3\}$ ,  $V_e = \{e_1, e_2, e_3\}$ ,  $V_g = \{g_1, g_2, g_3\}$ ,  $V_h = \{h_1, h_2, h_3\}$ , then we remove from  $S$  all tuples containing values  $a_2, a_3, b_2, b_3, c_1, e_1$  and we use again *LEERS* to extract rules from this subsystem. Each rule defining  $d_2$  is used jointly with  $r$  to construct an action rule. The validation step of each of the set-inclusion relations, in the second module of *ARAS*, is replaced by checking if the corresponding term was marked by *LEERS* in the first module of *ARAS*.

## 5 Conclusion

System *ARAS* differs from the tree-based strategies for action rules discovery (for instance from *DEAR* [11]) because clusters generated by its second module are formed around target classification rules. An action rule can be constructed

in *ARAS* from two classification rules only if both of them belong to the same cluster and one of them is a target classification rule. So, the complexity of the second module of *ARAS* is  $O(k \cdot n)$ , where  $n$  is the number of classification rules extracted by *LERS* and  $k$  is the number of clusters. The time complexity of the second module of *DEAR* is equal to  $O(n \cdot n)$ , where  $n$  is the same as in *ARAS*. The first module of *ARAS* is the same as the first module of *DEAR*, so their complexities are the same.

## 6 Acknowledgements

This research was partially supported by the National Science Foundation under grant IIS-0414815.

## References

1. Adomavicius, G., Tuzhilin, A. (1997) Discovery of actionable patterns in databases: the action hierarchy approach, in **Proceedings of KDD'97 Conference**, Newport Beach, CA, AAAI Press
2. Dardzinska, A., Ras, Z.W., "Extracting Rules from Incomplete Decision Systems: System ERID", in *Foundations and Novel Approaches in Data Mining*, (Eds. T.Y. Lin, S. Ohsuga, C.J. Liao, X. Hu), *Advances in Soft Computing*, Vol. 9, Springer, 2006, 143-154
3. Hilderman, R.J., Hamilton, H.J. (2001) **Knowledge Discovery and Measures of Interest**, Kluwer
4. Greco, S., Matarazzo, B., Pappalardo, N., Slowiński, R. (2005) Measuring expected effects of interventions based on decision rules, in **Journal of Experimental and Theoretical Artificial Intelligence**, Taylor Francis, Vol. 17, No. 1-2
5. Grzymala-Busse, J. (1997) A new version of the rule induction system LERS, in **Fundamenta Informaticae**, Vol. 31, No. 1, 27-39
6. Pawlak, Z., (1991) Information systems - theoretical foundations, in **Information Systems Journal**, Vol. 6, 205-218
7. Silberschatz, A., Tuzhilin, A., (1995) On subjective measures of interestingness in knowledge discovery, in **Proceedings of KDD'95 Conference**, AAAI Press
8. Raś, Z., Wieczorkowska, A. (2000) Action Rules: how to increase profit of a company, in **Principles of Data Mining and Knowledge Discovery**, LNAI, No. 1910, Springer, 587-592
9. Raś, Z.W., Tzacheva, A., Tsay, L.-S. (2005) Action rules, in **Encyclopedia of Data Warehousing and Mining**, (Ed. J. Wang), Idea Group Inc., 1-5
10. Raś, Z.W., Dardzińska, A. (2006) Action rules discovery, a new simplified strategy, in **Foundations of Intelligent Systems**, F. Esposito et al. (Eds.), LNAI, No. 4203, Springer, 445-453
11. Tsay, L.-S., Raś, Z.W. (2005) Action rules discovery system DEAR, method and experiments, in **Journal of Experimental and Theoretical Artificial Intelligence**, Taylor & Francis, Vol. 17, No. 1-2, 119-128
12. Tsay, L.-S., Raś, Z.W. (2006) Action rules discovery system DEAR3, in **Foundations of Intelligent Systems**, LNAI, No. 4203, Springer, 483-492
13. Tzacheva, A., Raś, Z.W. (2005) Action rules mining, in **International Journal of Intelligent Systems**, Wiley, Vol. 20, No. 7, 719-736