Action rules discovery: System DEAR2, method and experiments

Article *in* Journal of Experimental & Theoretical Artificial Intelligence · January 2005

DOI: 10.1080/09528130512331315855 · Source: DBLP

2 authors:

Li-Shiang Tsay
North Carolina Agricultural and Technical State University
38 PUBLICATIONS **426** CITATIONS
SEE PROFILE

Zbigniew W Ras
University of North Carolina at Charlotte
356 PUBLICATIONS **2,931** CITATIONS
SEE PROFILE

Journal of Experimental & Theoretical Artificial Intelligence,
Vol. 17, No. 1–2, January–June 2005, 119–128

Action rules discovery: system *DEAR2*, method and experiments

LI-SHIANG TSAY* and ZBIGNIEW W. RAŚ

Department of Computer Science, University of North Carolina,
9201 University City Blvd, Charlotte, NC 28223, USA

(*Received 1 March 2004; in final form 1 August 2004*)

Subjective measures, used to model interestingness of rules, are user-driven, domain-dependent, and include unexpectedness, novelty and actionability (Adomavicius and Tuzhilin 1997, Liu *et al.* 1997, Silberschatz and Tuzhilin 1995). Liu *et al.* (1997) define a rule as actionable, if a user can do an action to his/her advantage based on that rule. Their notion of actionability is too vague and leaves the door open to a number of different interpretations. Raś and Wieczorkowska (2000) assume that actionability has to be expressed in terms of attributes that are present in the database. They have introduced a new class of rules (called action rules) that are constructed from certain pairs of association rules extracted from that database. A conceptually similar definition of an action rule was proposed independently by Geffner and Wainer (1998). Action rules have been investigated further in Raś and Gupta (2002), Raś and Tsay (2003), Raś *et al.* (2005) and Tzacheva and Raś (2004).

In order to construct action rules, it is required that attributes in a database are divided into two groups: stable and flexible. Flexible attributes are used in a decision rule as a tool for making hints to a user what changes within some of their values are needed to reclassify a group of objects from one decision class into another one. Two strategies for generating action rules are presented. The first one, implemented as system *DEAR*, generates action rules from certain pairs of association rules. The second one, implemented as system *DEAR2*, is based on a tree structure that partitions the set of rules, having the same decision value, into equivalence classes each labelled by values of stable attributes (two rules belong to the same equivalence class, if values of their stable attributes are not conflicting each other). Now, instead of comparing all pairs of rules, only pairs of rules belonging to some of these equivalence classes are compared to construct action rules. This strategy significantly reduces the number of steps needed to generate action rules in comparison to *DEAR* system.

Keywords: Decision system; Knowledge discovery; Actionability; Interestingness

*Corresponding author. Email: ltsay@uncc.edu

1. Introduction

There are two aspects of interestingness of rules that have been studied in data mining literature: objective and subjective measures (see Liu *et al.* 1997, Adomavicius and Tuzhilin 1997 and Silberschatz and Tuzhilin 1995, 1996). Objective measures are data-driven and domain-independent. Generally, they evaluate the rules based on the quality and similarity between them. Subjective measures—including unexpectedness, novelty and actionability—are user-driven and domain-dependent.

A rule is actionable if a user can do an action to his/her advantage based on that rule (Liu *et al.* 1997). This definition, in spite of its importance, is quite vague and it leaves an open door to a number of different interpretations of actionability. For instance, we may formulate actions that involve attributes outside the database schema. In order to narrow it down, a new class of rules (called action rules) constructed from certain pairs of association rules, extracted from a given database, has been proposed in Raś and Wieczorkowska (2002). Independently, another formal definition of an action rule was proposed in Geffner and Wainer (1998). These rules have been investigated further in Raś and Tsay (2003) Raś *et al.* (2005), and Tzacheva and Raś (2003, 2004).

To give an example justifying the need of action rules, let us assume that a number of customers have stopped buying products at one of the grocery stores. To find the cause of their decision, possibly the smallest and the simplest set of rules describing all these customers is extracted from the customer database. For instance, let us assume that [*Nationality*, *European*] $\wedge$ [*Milk Products*, *Kefir*] $\longrightarrow$ [*Profit*, *Excellent*] is such a rule. Assume also that from the same database, a rule [*Nationality*, *European*] $\longrightarrow$ [*Profit*, *Average*] representing the remaining customers has been extracted. At the same time, we know that the grocery store stopped ordering *kefir* about a month ago. Now, by comparing these two rules, we can easily find out that the grocery store manager should start ordering kefir again if he does not want to loose more European customers. Action rule, constructed from these two rules, is represented by the expression: [*Nationality*, *European*] $\wedge$ [*Milk Products*, $\longrightarrow$ *Kefir*] $\longrightarrow$ [*Profit*, *Average* $\longrightarrow$ *Excellent*]. It should be read as: *If Europeans will buy Kefir, then they should shift from the average group of customers to the excellent one*. Ordering kefir by the grocery store manager is an example of its implementation.

Formal definition of an action rule and the algorithm to construct them from association rules was proposed in Raś and Wieczorkowska (2000). This algorithm was implemented as one of the modules in *DEAR* (Raś and Tsay 2003) system. System *DEAR2* presented in this paper significantly improves its previous version. Now, we outline this new strategy. Its first step is to partition the rules, defining values of the decision attribute, into a number of equivalence classes (two rules belong to the same equivalence class if they define the same decision value). In its second step, for each decision value, the algorithm builds dynamically a tree structure partitioning all rules having that decision value into a number of new equivalence subclasses defined by values of stable attributes (two rules belong to the same equivalence subclass, if values of their stable attributes do not contradict each other). In its final step, instead of comparing all pairs of rules, only pairs of rules belonging to some of these equivalence subclasses have to be compared in order to construct action rules. This strategy significantly reduces the number of steps needed to generate action rules in comparison to the strategy *DEAR* (Raś and Tsay 2003).

2. Information system and action rules

An information system is used for representing knowledge. Its definition, presented here, is due to Pawlak (1991).

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

- U is a non-empty, finite set of objects, and
- A is a non-empty, finite set of attributes, i.e. $a : U \longrightarrow V_a$ is a function for any $a \in A$, where V_a is called the domain of a.

Objects, for instance, can be interpreted as customers. Attributes can be interpreted as features, offers made by a grocery store, characteristic conditions, etc.

In this paper, we only consider the special case of information systems called decision tables (Pawlak 1991). 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 conditions and flexible conditions. For simplicity, 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 a bank). 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. The elements of A_1 are called stable conditions, whereas the elements of A_2 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.

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 $A_1 \cup A_2$, subtables $(U, B \cup \{d\})$ of S where B is a d-reduct (Pawlak 1991) in S are used for rules extraction. By $L(r)$, we mean all attributes listed in the IF part of a rule r. For example, if $r = [(a, 2) * (b, S) \longrightarrow (d, H)]$ is a rule then $L(r) = \{a, b\}$. By $d(r)$ we denote the decision value of that rule. In our example, $d(r) = H$. If r_1, r_2 are rules and $B \subseteq A_1 \cup A_2$ is a set of attributes, then the equation $r_1/B = r_2/B$ means that the

Table 1. Decision system S.

	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

conditional parts of rules r_1, r_2 restricted to attributes B are the same. Now, if $r_1 = [(b, S) * (c, 2) \longrightarrow (d, H)]$, then $r_1/\{b\} = r/\{b\}$.

The list of certain optimal rules extracted from S, represented by table 1, is given below:

$(a, 0) \longrightarrow (d, L)$, $(c, 0) \longrightarrow (d, L)$
$(b, R) \longrightarrow (d, L)$, $(c, 1) \longrightarrow (d, L)$
$(b, P) \longrightarrow (d, L)$, $(a, 2) * (b, S) \longrightarrow (d, H)$, $(b, S) * (c, 2) \longrightarrow (d, H)$

Any certain rule is optimal in S, if by dropping an attribute value listed in its conditional part we get a rule that is no longer certain one. Similar definition of optimality can be used for association rules if we assume that the assumption about their certainty is replaced by a threshold value for their minimal confidence.

Now, let us assume that $(a, v \longrightarrow w)$ denotes the fact that the value of attribute a has been changed from v to w for a number of objects in S. Similarly, the term $(a, v \longrightarrow w)(x)$ means that the attribute value $a(x) = v$ has been changed to $a(x) = w$ for the object x.

Let $S = (U, A_1 \cup A_2 \cup \{d\})$ be a decision table and certain rules r_1, r_2 are extracted from S. Assume that B_1 is a maximal subset of A_1 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 $(b_1, b_2, \ldots, b_p)$ is a list of all attributes in $L(r_1) \cap L(r_2) \cap A_2$ on which r_1, r_2 differ in values and $r_1(b_1) = v_1$, $r_1(b_2) = v_2, \ldots, r_1(b_p) = v_p$, $r_2(b_1) = w_1$, $r_2(b_2) = w_2, \ldots, r_2(b_p) = w_p$.

By (r_1, r_2)-action rule r on $x \in U$ we mean the expression:

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

Similarly to the notation adopted for rules extracted from S, we assume that $L(r) = \{b_1, b_2, \ldots, b_p\}$. Several definitions related to the support of an action rule can be proposed. We list them below.

We say that object x *certainly supports* the action rule r in S, if there is an object $y \in U$ such that:

- $(\forall i \leq p)[[b_i \in L(r)] \longrightarrow [b_i(x) = v_i]]$,
- $(\forall i \leq p)[[b_i \in L(r)] \longrightarrow [b_i(y) = w_i]]$,
- $(\forall b \in [[A_1 \cup A_2] - L(r)])[b(x) = b(y)]$,
- $d(x) = k_1$ and $d(y) = k_2$.

The object y is seen as the outcome of the rule r applied on object x.

We say that object x *negatively supports* the action rule r in S, if there is an object $y \in U$ such that:

- $(\forall i \leq p)[[b_i \in L(r)] \longrightarrow [b_i(x) = v_i]]$,
- $(\forall i \leq p)[[b_i \in L(r)] \longrightarrow [b_i(y) = w_i]]$,
- $(\forall b \in [[A_1 \cup A_2] - L(r)])[b(x) = b(y)]$,
- $d(x) = k_1$ and $d(y) \neq k_2$.

In this case, we say that the object that is the outcome of the rule r applied on x contradicts S.

We say that object x *possibly supports* the action rule r in S, if there is an object $y \in U$ such that:

- $(\forall i \leq p)[[b_i \in L(r)] \longrightarrow [b_i(x) = v_i]]$,
- $(\forall i \leq p)[[b_i \in L(r)] \longrightarrow [b_i(y) = w_i]]$,

- $(\forall b \in [A_1 - L(r)])[b(x) = b(y)]$,
- $d(x) = k_1$ and $d(y) = k_2$.

The definition of *certain* support of an action rule is too strong because it does not leave any door for generalizations. On the other hand, the definition of *possible* support of an action rule is too weak because it is based on the assumption that non-existing objects in S support the outcome of any action rule. To overcome this problem, the last definition proposed by us requires the extraction of rules from S before any action rule can be constructed.

We say that object x *supports* the action rule r in S, if there are two rules r_1, r_2 extracted from S such that:

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

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

Now, let us denote by $NSup_S(r)$ the set of all objects in S negatively supporting r.

By the confidence of r in S, denoted by $Conf_S(r)$, we mean $Sup_S(r)Sup_S(r_1)$.

The definition of *support* of an action rule is not very pleasant because it involves other rules extracted from S. In order to construct all action rules based on S, we need to consider all pairs of rules extracted from S and pick up only those which satisfy the condition required for action rules construction. Then, from each of these pairs we construct an action rule and calculate its support and confidence. To speed up the process of action rules construction, we only concentrate on action rules that are built from certain pairs of rules extracted from S. Namely, we consider pairs of rules not only satisfying the required condition for action rules construction but also which have a small number of overlapping flexible attributes in their conditional part. This way we minimize the number of attribute values required by an action rule to be changed for any object supporting that rule.

3. Discovering extended action rules

In order to simplify the process required to compute the support of an action rule, the notion of an extended action rule was proposed in Raś and Tsay (2003). Its informal definition is given below.

Let us assume that two rules extracted from some decision system are represented as rows in table 2. This table representation is useful to clarify the construction of an extended action rule. Here, $A(St)$ means that attribute A is stable and $A(Fl)$ means that A is flexible.

In a classical representation, these two rules have the following form:

$$r_1 = [a_1 * b_1 * c_1 * e_1 \longrightarrow d_1], \;\; r_2 = [a_1 * b_2 * g_2 * h_2 \longrightarrow d_2]$$

Assume now that object x supports rule r_1, which means that x is assigned to class d_1. In order to reclassify x to the class d_2, we need to change its value of attribute B from b_1 to b_2. Additionally, we require that $G(x) = g_2$ and that the value of

Table 2. Two rules extracted from decision system.

A (St)	B (Fl)	C (St)	E (Fl)	G (St)	H (Fl)	D ($Decision$)
a_1	b_1	c_1	e_1			d_1
a_1	b_2			g_2	h_2	d_2

attribute H for object x has to be changed from its current value to h_2. This is the meaning of the extended (r_1, r_2)-action rule given below:

$$[(B, b_1 \longrightarrow b_2) \wedge (G = g_2) \wedge (H, \longrightarrow h_2)](x) \longrightarrow (D, d_1 \longrightarrow d_2)(x)$$

To give example of an extended action rule, let us assume that $S = (U, A_1 \cup A_2 \cup \{d\})$ is a decision system represented as table 1, $A_2 = \{b\}$, $A_1 = \{a, c\}$. It can be checked that rules $r_1 = [(b, P) \longrightarrow (d, L)]$, $r_2 = [(a, 2) * (b, S) \longrightarrow (d, H)]$, $r_3 = [(b, S) * (c, 2) \longrightarrow (d, H)]$ can be extracted from S. Clearly, both objects x_5, x_6 support rule r_1. The resulting (r_1, r_2)-action rule, which can be applied to reclassify x, is given below:

$$[(b, P \longrightarrow S)](x) \longrightarrow [(d, L \longrightarrow H)](x)$$

Now, the extended (r_1, r_2)-action rule that also can be applied to reclassify x is given:

$$[(a = 2) * (b, P \longrightarrow S)](x) \longrightarrow [(d, L \longrightarrow H)](x)$$

Clearly objects x_5, x_6 support both action rules.

We are almost ready to present our new algorithm, implemented as the main part of *DEAR2* system, for discovering action rules and extended action rules. Initially, we partition the set of rules discovered from an information system $S = (U, A_1 \cup A_2 \cup \{d\})$—where A_1 is the set of stable attributes, A_2 is the set of flexible attributes and, $V_d = \{d_1, d_2, \ldots, d_k\}$ is the set of decision values—into a minimal number of subsets, each of which contains rules defining the same decision value. Saying another words, the set of rules R discovered from S is partitioned into $\{R_i\}_{i:1 \leq i \leq k}$, where $R_i = \{r \in R\colon d(r) = d_i\}$ for any $i = 1, 2, \ldots, k$. Equivalently, objects from U are partitioned into subsets $d^{-1}(\{d_i\})$, $1 \leq i \leq k$.

Let us take the system S represented by table 1 as an example of a decision system. We assume that a, c are its stable attributes and b, d are flexible. The set R of certain rules extracted from S is equal to:

$$\{(a, 0) \longrightarrow (d, L), (c, 0) \longrightarrow (d, L), (b, R) \longrightarrow (d, L), (c, 1) \longrightarrow (d, L)$$

$$(b, P) \longrightarrow (d, L), (a, 2) * (b, S) \longrightarrow (d, H), (b, S) * (c, 2) \longrightarrow (d, H)\}$$

We partition this set into two subsets $R_1 = \{[(a, 0) \longrightarrow (d, L)], [(c, 0) \longrightarrow (d, L)], [(b, R) \longrightarrow (d, L)], [(c, 1) \longrightarrow (d, L)], [(b, P) \longrightarrow (d, L)]\}$ and $R_2 = \{[(a,2)*(b,S) \longrightarrow (d,H)], [(b, S) * (c, 2) \longrightarrow (d, H)]\}$.

Now, our goal is to reclassify some objects from the class $d^{-1}(\{L\})$ into the class $d^{-1}(\{H\})$.

The set of rules R_1 is represented as table 3. Each row represents one rule. The first column in each row shows objects in S supporting the corresponding rule. The set of rules R_2 is represented as table 4 in figure 2. In the general case, assumed earlier,

Table 3. Set of rules R_1 with objects supporting them.

	a	*b*	*c*	*d*
$\{x_1,x_2,x_3,x_4\}$	0			*L*
$\{x_2,x_4\}$		*R*		*L*
$\{x_1,x_3\}$			0	*L*
$\{x_2,x_4\}$			1	*L*
$\{x_5,x_6\}$		P		*L*

Table 4. Time needed to extract rules and action rules.

Data set	Rules	Action rules *DEAR*	Action rules *DEAR2*
Breast cancer	20 sec	27 min 51 sec	3 sec
Cleveland	1 min 09 sec	Over 8 hrs	54 min 20 sec
Hepatitis	54 sec	Over 8 hrs	51 min 53 sec

the number of different decision classes and the same the number of tables representing them is equal to k.

The first step of the algorithm is to build d_i-tree and d_j-tree. From the set of rules R extracted from S, all rules with the decision value d_i are selected and presented in a table format similar to table 3 in figure 1. Similarly, we select all rules with decision value d_j and present them in a table format similar to table 4 in figure 2.

By d_i-tree we mean a tree $T(d_i) = (N_i, E_i)$, such that:

- each interior node is labelled by a stable attribute from A_1;
- each edge is labelled either by a question mark or by an attribute value of the attribute that labels the initial node of that edge;
- along a path, all nodes (except a leaf) are labelled with different stable attributes;
- all edges leaving a node are labelled with different attribute values (including the question mark) of the stable attribute that labels that node; and
- each leaf represents a set of rules that do not contradict on stable attributes and also define decision value d_i. The path from the root to that leaf gives the description of objects supported by these rules.

Now, taking (d, L) from our example as the value d_i, we show how to construct (d, L)-tree for the set of rules R_1. The construction of (d, L)-tree starts with a table associated with the root of that tree (table 3 in figure 1). It represents the set of rules R_1 defining L and listed with objects from S supporting them. We use stable attribute c to split table 3 into three sub-tables associated with values $\{0, 1, ?\}$ of attribute c. The question mark denotes here an unknown value.

Following the path labelled by value $c = ?$, we get table $T1$. Following the path labelled by value $c = 1$, we get table $T2$. Following the path labelled by value $c = 0$, we get table $T3$. Following the path labelled by value $[c = ?][a = 0]$, we get table $T4$. Finally, by following the path having the label $[c = ?][a = ?]$, we get table $T5$.

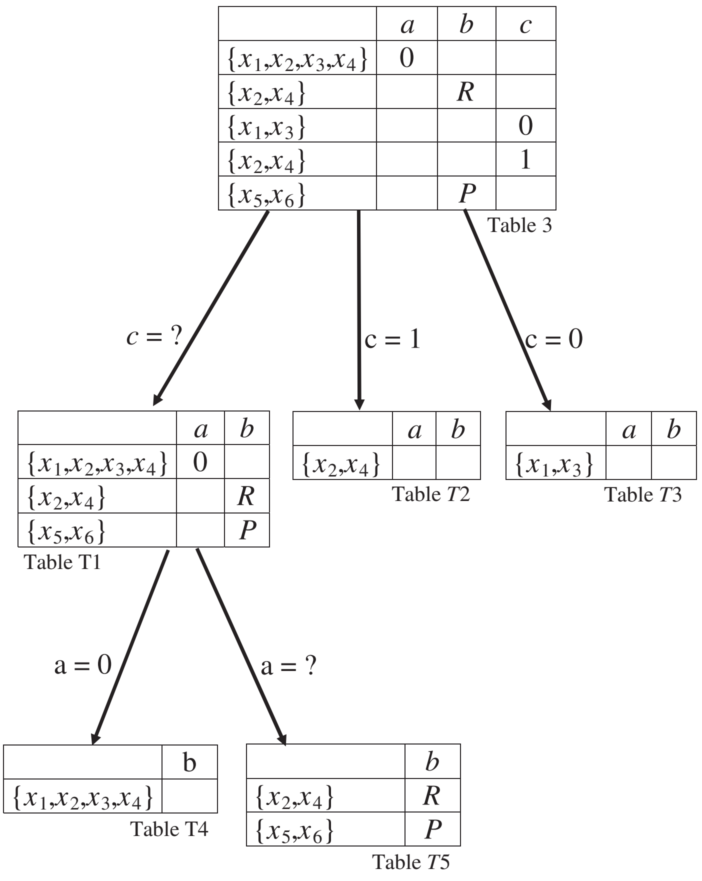


Figure 1. (d, L)-tree.

Now, let us define (d, H)-tree using table 4 as its root (see figure 2). Following the path labelled by value $[c = ?]$, we get the table $T6$. When we follow the path labelled by value $[c = 2]$, we get table $T7$. Both tables can be easily constructed. Now, it can be checked that only pairs of rules belonging to tables $\{[T5, T7], [T5, T6], [T2, T6], [T3, T6], [T4, T7]\}$ can be used for action rules construction.

For each pair of tables, we use the same algorithm as in Raś and Tsay (2003) to construct extended action rules. This new algorithm (called *DEAR2*) was implemented and tested on many datasets using PC with 1.8 GHz CPU. The time complexity of this algorithm was significantly lower than the time complexity of the algorithm *DEAR* presented in Raś and Tsay (2003). Both algorithms extract rules describing values of the decision attribute before any action rule is constructed. Table 4 shows the time needed by systems *DEAR* and *DEAR2* to extract rules and next action rules from three datasets: *breast cancer*, *Cleveland*, *hepatitis*. These three *UCI* datasets are available at http://www.sgi.com/tech/mlc/db/. The first one has 191 records described by 10 attributes. Only *Age* was taken by us as the stable attribute. The second one has 303 records described by 15 attributes. Only two attributes *age* and *sex* have been taken as stable. The last one has 155 records described by 19 attributes. Again, only two attributes *age* and *sex* have been taken by us as stable.

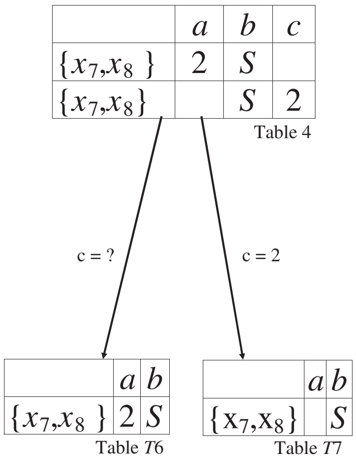
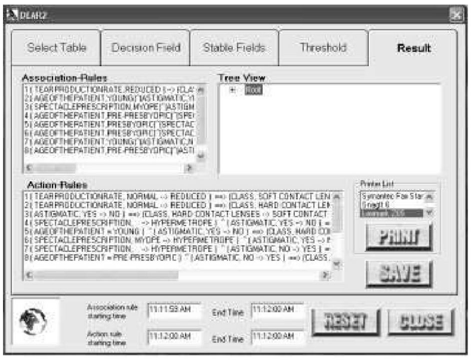


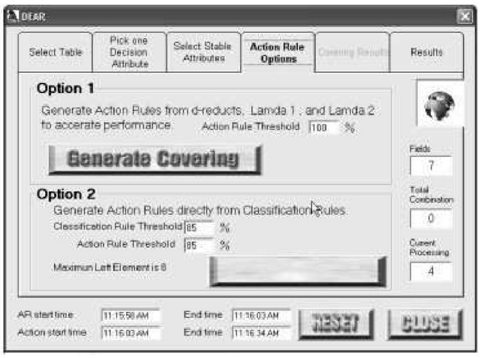
Figure 2. (d, H)-tree.



Association Rules
1{ TEARPRODUCTIONRATE,REDUCED }-> {CLA
2{ AGEOFTHEPATIENT,YOUNG}^{ASTIGMATIC,YI
3{ SPECTACLEPRESCRIPTION,MYOPE}^{ASTIGM
4{ AGEOFTHEPATIENT,PRE-PRESBYOPIC}^{SPE
5{ AGEOFTHEPATIENT,PRESBYOPIC}^{SPECTAC
6{ AGEOFTHEPATIENT,PRESBYOPIC}^{SPECTAC
7{ AGEOFTHEPATIENT,YOUNG}^{ASTIGMATIC,N
8{ AGEOFTHEPATIENT,PRE-PRESBYOPIC}^{ASTI

Action Rules
1{ TEARPRODUCTIONRATE, NORMAL -> REDUCED } ==> {CLASS, SOFT CONTACT LEN
2{ TEARPRODUCTIONRATE, NORMAL -> REDUCED } ==> {CLASS, HARD CONTACT LEN
3{ ASTIGMATIC, YES -> NO } ==> {CLASS, HARD CONTACT LENSES -> SOFT CONTACT
4{ SPECTACLEPRESCRIPTION, -> HYPERMETROPE } ^ {ASTIGMATIC, YES -> NO } =
5{ AGEOFTHEPATIENT = YOUNG } ^ {ASTIGMATIC, YES -> NO } ==> {CLASS, HARD CO
6{ SPECTACLEPRESCRIPTION, MYOPE -> HYPERMETROPE } ^ {ASTIGMATIC, YES ->
7{ SPECTACLEPRESCRIPTION, -> HYPERMETROPE } ^ {ASTIGMATIC, NO -> YES } =
8{ AGEOFTHEPATIENT = PRE-PRESBYOPIC } ^ {ASTIGMATIC, NO -> YES } ==> {CLASS,

Association rule starting time 11:11:53 AM End Time 11:12:00 AM
Action rule starting time 11:12:00 AM End Time 11:12:00 AM



Option 1
Generate Action Rules from d-reducts, Lamda 1 ; and Lamda 2 to accerate performance. Action Rule Threshold 100 %

Option 2
Generate Action Rules directly from Classification Rules.
Classification Rule Threshold 85 %
Action Rule Threshold 85 %
Maximun Left Element is 8

Fields 7
Total Combination 0
Current Processing 4

AR start time 11:15:58 AM End time 11:16:03 AM
Action start time 11:16:03 AM End time 11:16:34 AM

Figure 3. *DEAR* and *DEAR2* interface.

The interface for both systems, *DEAR* and *DEAR2*, is written in Visual Basic. The second picture presented in figure 3 shows a part of the interface used in both systems. The user has an option to generate the coverings (Pawlak 1981, 1991) for the decision attribute and next use them in the process of action rules extraction or, if he prefers, he can directly proceed to the rules extraction step. It is recommended, in *DEAR2*, to generate the coverings for a decision attribute if the information system has many attributes. By doing this we may significantly speed up the process of action rules extraction. The first picture in figure 3 shows how the results are displayed by *DEAR2* system.

4. Conclusion

System *DEAR2* initially generates a set of association rules from *S* (satisfying two thresholds: the first one for a minimum support and second for a minimum confidence) defining values of a decision attribute in *S*, in terms of the remaining attributes. *DEAR2* is giving preference to rules whose classification part contains small number of stable attributes in *S*. These rules are partitioned by *DEAR2* into a number of equivalence classes. Two rules belong to the same equivalence class if their classification parts are the same on stable attributes. Each equivalence class is used independently by *DEAR2* as a base for constructing action rules. The current strategy requires the generation of association rules from *S* to form a base, before the process of action rules construction starts.

In a separate paper, we propose a strategy for generating action rules directly from *S* that is similar to *LERS* (Chmielewski *et al*. 1993, Grzymala-Busse 1997) and *ERID* (Dardzinska and Raś 2003). This new strategy is initially centred on all stable attributes in *S* and does not require to generate the base formed from association rules in order to construct action rules.

Acknowledgement

This paper is in memory of Jan M. Żytkow.

References

G. Adomavicius and A. Tuzhilin, "Discovery of actionable patterns in databases: the action hierarchy approach", in *Proceedings: KDD'97 Conference*, 1997.

M.R. Chmielewski, J.W. Grzymala-Busse, N.W. Peterson and S. Than, "The rule induction system LERS – a version for personal computers", *International Journal of Approximate Reasoning*, 1993, pp. 181–212.

A. Dardzińska and Z.W. Raś, "On rule discovery from incomplete information systems", in *Proceedings: ICDM'03 Workshop on Foundations and New Directions of Data Mining*, T.Y. Lin, X. Hu, S. Ohsuga and C. Liau (Eds.), 2003, pp. 31–35.

H. Geffner and J. Wainer, "Modeling action, knowledge and control", in *Proceedings: ECAI'98, the 13th European Conference on AI*, H. Prade, (Ed.), 1998, pp. 532–536.

J. Grzymala-Busse, "A new version of the rule induction system LERS", *Fundamenta Informaticae*, 31, 1998, pp. 27–39.

B. Liu, W. Hsu and S. Chen, "Using general impressions to analyze discovered classification rules", in *Proceedings: KDD'97 Conference* 1997.

Z. Pawlak, *Rough Sets-theoretical Aspects of Reasoning about Data*, Dordrecht: Kluwer, 1991.

Z. Pawlak, "Information systems – theoretical foundations", *Information Systems Journal*, 6, 1981, pp. 205–218.

L. Polkowski and A. Skowron, "Rough sets in knowledge discovery", in *Studies in Fuzziness and Soft Computing*, Springer: Physica-Verlag, 1998.

Z. Raś and S. Gupta, "Global action rules in distributed knowledge systems", *Fundamenta Informaticae Journal*, 51, 2002, pp. 175–184.

Z. W. Raś and L.-S. Tsay, "Discovering extended action-rules (System DEAR)", in *Intelligent Information Systems 2003, Proceedings: IIS'03 Symposium*, 2003, pp. 293–300.

Z. Raś and A. Wieczorkowska, "Action rules: how to increase profit of a company", in *Principles of Data Mining and Knowledge Discovery*, Zighed, D.A. Komorowski, J. and Zytkow, J. (Eds), Lyon: Springer-Verlag, 2000, pp. 587–592.

Z.W. Raś, A. Tzacheva and L.-S. Tsay, "Action rules", in *Encyclopedia of Data Warehousing and Mining*, Wang, J. (Ed.), Idea Group Inc., 2005.

A. Silberschatz and A. Tuzhilin, "On subjective measures of interestingness in knowledge discovery", in *Proceedings: KDD'95 Conference* 1995.

A. Silberschatz and A. Tuzhilin, What makes patterns interesting in knowledge discovery systems, *IEEE Transactions on Knowledge and Data Engineering*, 5, 1996.

A. Tzacheva and Z.W. Raś, "Discovering non-standard semantics of semi-stable attributes", in *Proceedings: FLAIRS'03 Conference*, 2003, pp. 330–334.

A. Tzacheva and Z.W. Raś, "Action rules mining", *International Journal of Intelligent Systems*, Forthcoming, 2004