Action Rule Extraction From A Decision Table : ARED

Seunghyun Im[1] and Zbigniew Ras[2,3]

[1] University of Pittsburgh at Johnstown, Department of Computer Science
Johnstown, PA. 15904, USA
[2] University of North Carolina, Department of Computer Science
Charlotte, NC, 28223, USA
[3] Polish Academy of Sciences, Institute of Computer Science, 01-237 Warsaw, Poland
e-mail: sim@pitt.edu, ras@uncc.edu

Abstract. In this paper, we present an algorithm that discovers action rules from a decision table. Action rules describe possible transitions of objects from one state to another with respect to a distinguished attribute. The previous research on action rule discovery required the extraction of classification rules before constructing any action rule. The new proposed algorithm does not require pre-existing classification rules, and it uses a bottom up approach to generate action rules having minimal attribute involvement.

1 Introduction

In this paper, we present an algorithm that discovers action rules. An action rule is a rule extracted from a database that describes a possible transition of objects from one state to another with respect to a distinguished attribute called a decision attribute [14]. Values of some attributes, used to describe objects stored in a database, can be changed. This change can be influenced and controlled by users. For example, let us assume that a number of customers have closed their bank accounts. We construct a description of this group of customers. Then, we search for another description of a new group of customers who keep the bank accounts active. If these descriptions have a form of rules, they can be seen as actionable rules. For instance, by comparing those two descriptions, we may identify the reason for closing their accounts, and formulate an action, which if undertaken by the bank, may prevent other customers from closing their accounts. In this case, an action rule may say that, if the bank lowers the interest rate by 2 percent on credit cards for certain group of customers, it is almost guaranteed that they do not close their accounts. A similar definition, but with different notation, of an action rule was given earlier in [4]. Also, interventions introduced in [5] are conceptually similar to action rules. Action rules introduced in [14] has been further investigated in [17][13][12][18][15].

Paper [7] was probably the first attempt towards formally introducing the problem of mining action rules without pre-existing classification rules. Authors explicitly formulated it as a search problem in a support-confidence-cost framework. The proposed algorithm is similar to Apriori [1]. Their definition of an action rule allows changes on stable attributes. Changing the value of an attribute, either stable or flexible, is linked

with a cost [18]. In order to rule out action rules with undesired changes on stable attributes, authors have assigned very high cost to such changes. However, that way, the cost of action rules discovery is getting unnecessarily increased. Also, they did not take into account the dependencies between attribute values which are naturally linked with the cost of rules used either to accept or reject a rule.

Algorithm *ARED*, presented in this paper, is based on Pawlak′s model of an information system S [10] and its goal is to identify certain relationships between granules defined by the indiscernibility relation on objects in S. Some of these relationships uniquely define action rules for S.

The rest of this paper is organized as follows. Section 2 describes the background and objectives of our work. The details of the algorithm *ARED* are presented in Section 3. Experimental results are shown in Section 4. Section 5 discusses possible future work and concludes the paper.

2 Backgrounds and Objectives

Action rules are extracted from an information system. By an information system S, we mean $S = (X, A, V)$ where X is a nonempty finite set of objects, A is a nonempty finite set of attributes, and $V = \bigcup\{V_a : a \in A\}$ is set of their values. For example, Table 1 presents an information system S with $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$, $A = \{a, b, c, d\}$, and $V = \{a_1, a_2, b_1, b_2, b_3, c_1, c_2, d_1, d_2, d_3\}$.

An information system $S = (X, A, V)$ is called a decision system (or decision table), if $A = A_{St} \cup A_{Fl} \cup \{d\}$, where d is a distinguished attribute set called the decision. Attributes in A_{St} are called *stable* and attributes in A_{Fl} are called *flexible*. They jointly form the set of conditional attributes. “Date of birth” is an example of a stable attribute. “Interest rate” for each customer account is an example of a flexible attribute.

In earlier works in [14][17][13][12][15], action rules are constructed from classification rules. This means that we use pre-existing classification rules or generate them using a rule discovery algorithm, such as LERS [6] or ERID [2], then, construct action rules either from certain pairs of the rules or from a single classification rule. For instance, algorithm $ARAS$ [15] generates sets of terms (built from values of attributes) around classification rules and constructs action rules directly from them. In this study, we propose a different approach to achieve the following objectives:

1. Extract action rules directly from an information system without using pre-existing conditional rules.
2. Extract all distinct action rules that have minimal attribute involvement.

To meet these two goals, we first generate action rules using two attributes, and iteratively apply the technique to generate more specific action rules.

3 Algorithm ARED

We describe the algorithm, *ARED* (Action Rule Extraction from Decision table), by working through an example using the decision Table S shown in Table 1. We assume

that the decision attribute is d, stable attributes $A_{St} = \{a\}$, and the flexible attributes $A_{Fl} = \{b, c\}$. The minimum support (λ_1) and confidence (λ_2) are given as 1 and 0.85. For simplicity reason, we will consider decision tables with only one decision in this paper.

	a	b	c	d
x_1	a_1	b_1	c_1	d_1
x_2	a_2	b_1	c_2	d_1
x_3	a_2		c_2	d_1
x_4	a_2	b_1	c_1	d_1
x_5	a_2	b_3	c_2	d_1
x_6	a_1	b_1		d_2
x_7	a_1	b_2	c_2	d_1
x_8	a_1	b_2	c_1	d_3

Table 1 : Decision Table S

The first step is to find the pessimistic interpretation in S of all attribute values in V, as shown in Table 2 (for conditional attributes) and Table 3 (for decision attribute). The resulting sets are called granules. In S, $Dom(a) = \{a_1, a_2\}$, $Dom(b) = \{b_1, b_2, b_3\}$, $Dom(c) = \{c_1, c_2\}$, and $Dom(d) = \{d_1, d_2, d_3\}$. The granule a_1^* associated with attribute value a_1 in S (see Table 1) is the set of objects having property a_1 (e.g. objects $\{x_1, x_6, x_7, x_8\}$). The set of granules associated with an attribute a in S is defined as $\{v^* : v \in V_a\}$.

$a_1{}^* = \{x_1, x_6, x_7, x_8\}$
$a_2{}^* = \{x_2, x_3, x_4, x_5\}$
$b_1{}^* = \{x_1, x_2, x_4, x_6\}$
$b_2{}^* = \{x_7, x_8\}$
$b_3{}^* = \{x_5\}$
$c_1{}^* = \{x_1, x_4, x_8\}$
$c_2{}^* = \{x_2, x_3, x_5, x_7\}$

Table 2. Granules associated with attributes in a, b, c

$d_1{}^* = \{x_1, x_2, x_3, x_4, x_5, x_7\}$
$d_2{}^* = \{x_6\}$
$d_3{}^* = \{x_8\}$

Table 3. Granules associated with attribute d

Next, we define two sets, τ and δ, to examine possible *property transitions* between objects in S. Let T be a set of proper conjuncts built from elements in $\cup\{V_i, i \neq d, i \in A\}$. By proper conjunct, we mean a conjunct which contains maximum one element from each V_i.

– $\tau = T \cdot d_1$, where $d_1 \in V_d$, and $(\forall \rho_1 \in T \cdot d_1)(sup(\rho_1) \geq \lambda_1)$.
– $\delta = T \cdot d_2$, where $d_2 \in V_d$, and $(\forall \rho_2 \in T \cdot d_2)(sup(\rho_2) \geq \lambda_1)$.

By $T \cdot d_i$, we mean $\{t \cdot d_i : t \in T\}$, i=1,2. The support of ρ_i, $sup(\rho_i)$, is the number of objects in S supporting all attribute values listed in ρ_i, $i = 1, 2$. They can be easily calculated by intersecting two granules listed in Table 2 and Table 3. For example, $(a_1 \cdot d_2)^* = \{x_1, x_6, x_7, x_8\} \cap \{x_6\} = \{x_6\}$. So, $sup((a_1 \cdot d_2)^*) = 1$.

Each set contains only terms built from one decision value and at least one value of conditional attribute. Therefore, these sets represent (1) a relationship between conditional attributes and the decision attribute, and (2) property of a set of objects. If the property transition from τ to δ is valid, τ and δ are interpreted as the *condition* and the *decision* of an *action rule*.

As mentioned, *ARED* attempts to discover the shortest action rules in terms of the number of attributes, then iteratively generates longer action rules. Therefore, we first construct τ containing two elements (which is the shortest form of τ) by combining one attribute from Table 2 and the other from Table 3. This process aims to find the meaning in S of all terms in a conjunctive form built from values in V [16]. The concatenation functor used to build these terms is interpreted as the set-theoretical intersection.

τ	δ
$(a_1 \cdot d_1)$	$(a_1 \cdot d_1)$
$(a_1 \cdot d_2)$	$(a_1 \cdot d_2)$
$(a_1 \cdot d_3)$	$(a_1 \cdot d_3)$
$(a_2 \cdot d_1)$	$(a_2 \cdot d_1)$
$(b_1 \cdot d_1)$	$(b_1 \cdot d_1)$
$(b_1 \cdot d_2)$	$(b_1 \cdot d_2)$
$(b_2 \cdot d_1)$	$(b_2 \cdot d_1)$
$(b_2 \cdot d_3)$	$(b_2 \cdot d_3)$
$(b_3 \cdot d_1)$	$(b_3 \cdot d_1)$
$(c_1 \cdot d_1)$	$(c_1 \cdot d_1)$
$(c_1 \cdot d_3)$	$(c_1 \cdot d_3)$
$(c_2 \cdot d_1)$	$(c_2 \cdot d_1)$

Table 4. 2-element τ and δ

τ	δ	sup	$conf$	$rule$
$(b_1 \cdot d_2)$	$\mapsto (b_2 \cdot d_1)$	1	1	y
$(b_1 \cdot d_2)$	$\mapsto (b_2 \cdot d_3)$	1	1	y
$(b_1 \cdot d_2)$	$\mapsto (b_3 \cdot d_1)$	1	1	y
$(b_2 \cdot d_1)$	$\mapsto (b_1 \cdot d_2)$	1	1	y
$(b_2 \cdot d_3)$	$\mapsto (b_1 \cdot d_1)$	1	1	y
$(b_2 \cdot d_3)$	$\mapsto (b_1 \cdot d_2)$	1	1	y
$(b_2 \cdot d_3)$	$\mapsto (b_3 \cdot d_1)$	1	1	y
$(b_3 \cdot d_1)$	$\mapsto (b_1 \cdot d_2)$	1	1	y
$(b_3 \cdot d_1)$	$\mapsto (b_2 \cdot d_3)$	1	1	y
$(c_1 \cdot d_3)$	$\mapsto (c_2 \cdot d_1)$	1	1	y

Table 5. Action Rules

τ	δ	sup	$conf$	$rule$
$(b_1 \cdot d_1)$	$\mapsto (b_2 \cdot d_3)$	1	0.33	n
$(c_2 \cdot d_1)$	$\mapsto (c_1 \cdot d_3)$	1	0.25	n
$(a_1 \cdot d_1)$	$(a_1 \cdot d_1)$			
$(a_1 \cdot d_2)$	$(a_1 \cdot d_2)$			
$(a_1 \cdot d_3)$	$(a_1 \cdot d_3)$			
$(a_2 \cdot d_1)$	$(a_2 \cdot d_1)$			

Table 6. Invalid Transitions

Clearly, it is required that $sup(\rho_1) \geq \lambda_1 \wedge sup(\rho_2) \geq \lambda_1$ because we need to find property transitions that involve at least λ_1 objects. Terms satisfying these criteria, related to the example in Table 1, are shown in Table 4.

Now, we construct action rules using τ to δ in Table 4 by evaluating the validity of their transitions. We use the following code to explain the evaluation method.

input : $\{\tau\}, \{\delta\}$
output : action_rule [],gen_τ[], gen_δ[]

1: for each $\tau_i = [t_1 \cdot d_1] \in \tau, \delta_j = [t_2 \cdot d_2] \in \delta$
2: if $d_1 \neq d_2$ then
3: if $\exists(a(x_i) \in t_1, a(x_j) \in t_2)$, where $a \in A_{Fl}$ and $a(x_i) \neq a(x_j)$ then
4: if $conf(\tau_i \mapsto \delta_j) \geq \lambda_2$
5: action_rule [end+1] = $\tau_i \mapsto \delta_j$
6: elseif
7: gen_τ[end+1] = τ_i, gen_δ [end+1] = δ_j
8: end if
9: else
10: gen_τ[end+1] = τ_i, gen_δ [end+1] = δ_j
11: end if
12: end if

In line 1, we read each τ_i and δ_i listed in Table 4. Note that an information system may or may not produce two-way action rules (e.g. if b changes from b_1 to b_2 then d changes from d_1 to d_2, and vice versa). Therefore, we generate all pairs of elements from τ and δ. The condition in line 2 is clear. No action can be performed if the decision values are equal. The condition in line 3 checks if there exist different flexible attribute values between t_1 and t_2, and they are from the same domain. For example, $d_1 \neq d_2$ and $a_1 \neq a_2$ for $(a_1 \cdot d_2)$ and $(a_2 \cdot d_1)$. However, a is a stable attribute, so no action rule is constructed. If two sets do not meet this condition, we put them in two separate arrays (line 10) and use them to generate τ and δ for the next iteration. Last 4 rows in Table 6 are the sets in this category. The reasoning behind this strategy is simple. Clearly, there are at least λ_1 objects supporting $(a_1 \cdot d_2)$ and $(a_2 \cdot d_1)$. These sets may be concatenated with one or more flexible attribute values in later iterations, and produce action rules. An example of this case is $(a_1 \cdot b_1 \cdot d_1) \mapsto (a_1 \cdot b_2 \cdot d_3)$ in Table 8. The condition in line 4 checks the confidence of $\tau_i \mapsto \delta_j$. If it is greater than or equal λ_2, $\tau_i \mapsto \delta_j$ becomes an action rule. To compute the confidence, we first find the support of $ar = \tau \mapsto \delta$ which is defined as the minimum support of two sets.

$$sup(ar) = min(sup(\tau), sup(\delta))$$

$min(sup, sup)$ finds the exact number of object transitions because at least one element is different between τ and δ, and by definition of S, an object cannot support two different values of the same attribute. Note that we do not need to check the support of ar in line 4 because the support of τ and δ were checked when they were generated, and the support of the action rule is the smaller value between them. The confidence of an ar is defined as,

$$conf(ar) = \frac{sup(ar)}{sup(\tau)}$$

Two-element action rules extracted from S are shown in Table 5. For example, $(b_1 \cdot d_2) \mapsto (b_2 \cdot d_1)$ is an action rule, and it is interpreted as, "if b changes from b_1 to b_2,

then d changes from d_1 to d_2". Finally, if the confidence is less than λ_2 (line 7) τ_i and δ_j are considered in the next iteration to generate 3-element candidate sets. In Table 6, we have two transitions having confidence of 0.33 and 0.25 that are less than 1 (λ_2).

Assume that $\mid t_i \mid$, for any conjunct term t_i, denotes the set of all values of attributes listed in t_i. To find the action rules of length 3, we generate τ of length 3 from τs in Table 6. Two terms $\tau_1 = t_1 \cdot d_1$ and $\tau_2 = t_2 \cdot d_2$ are concatenated if (1) $d_1 = d_2$ (2) $\mid t_1 \cup t_2 \mid - \mid t_2 \cap t_1 \mid = \{v_1 \in V_a, v_2 \in V_b\}$, where $a \neq b$. The set $\{\delta\}$ is generated from δs in Table 6 using the same method. That means they are generated independently. The reason that we build 3-element candidate sets separately in this way is that all pairs are considered in the initial step. Therefore, any super set of the candidate set identified as an action rule will not be considered. Table 7 shows those 3-element candidate sets. Corresponding action rule and invalid transitions are show in Table 8 and 9 respectively.

τ	δ
$(a_2 \cdot b_1 \cdot d_1)$	$(a_1 \cdot c_2 \cdot d_3)$
$(a_2 \cdot c_3 \cdot d_1)$	$(a_1 \cdot b_2 \cdot d_3)$
$(a_1 \cdot b_1 \cdot d_1)$	$(b_2 \cdot c_2 \cdot d_3)$
$(a_1 \cdot c_3 \cdot d_1)$	
$(b_1 \cdot c_3 \cdot d_1)$	

Table 7. 3-element τ and δ

τ	δ	sup	$conf$	$rule$
$(a_1 \cdot b_1 \cdot d_1)$	$\mapsto (a_1 \cdot b_2 \cdot d_3)$	1	1	y
$(a_1 \cdot c_3 \cdot d_1)$	$\mapsto (a_1 \cdot c_2 \cdot d_3)$	1	1	y
$(b_1 \cdot c_3 \cdot d_1)$	$\mapsto (b_2 \cdot c_2 \cdot d_3)$	1	1	y

Table 8. Action Rule

τ	δ	sup	$conf$	$rule$
$(a_2 \cdot b_1 \cdot d_1)$	$\mapsto (a_1 \cdot b_2 \cdot d_3)$	1	0.50	n
$(a_2 \cdot c_3 \cdot d_1)$	$\mapsto (a_1 \cdot c_2 \cdot d_3)$	1	0.33	n

Table 9. Invalid Transition

In the next iteration, we construct τ and δ listed in Table 10 and build an action rule containing 4 elements as shown in Table 11. However, it is not included in the list of action rules because it's τ and δ are supersets of $(b_1 \cdot c_3 \cdot d_1) \mapsto (b_2 \cdot c_2 \cdot d_3)$ in Table 8, which is a more general action rule.

τ	δ
$(a_2 \cdot b_1 \cdot c_3 \cdot d_1)$	$(a_1 \cdot b_2 \cdot c_2 \cdot d_3)$

Table 10. 4-element τ and δ

τ	δ
$(a_2 \cdot b_1 \cdot c_3 \cdot d_1) \mapsto$	$(a_1 \cdot b_2 \cdot c_2 \cdot d_3)$

Table 11. Action Rule

The process stops because there are no sets to be combined.

4 Experiment

We implemented the algorithm in Matlab on a Pentium M 1.6 GHz computer running Windows XP, and tested it using a sample data set (nursery database) obtained from [8]. The data set contains information about applications for nursery schools, and used to rank them. The decision attribute has five classes (not recommend, recommend, very

recommend, priority, special priority).We partitioned the attributes into stable and flexible based on the description given by the provider. For example, the number of children in a family (1, 2, 3, more) is considered as a stable attribute, while financial standing of the family (convenient, inconvenient) or parents' occupation (usual, pretentious, great pretentious) are considered as flexible attributes.

Type	Attribute Name	Description
flexible	parents (p)	Parents' occupation
stable	has_nurs (n)	Child's nursery
stable	form (o)	Form of the family
stable	children (c)	Number of children
flexible	housing (h)	Housing conditions
flexible	finance (f)	Financial standing of the family
stable	social (s)	Social conditions
flexible	health (t)	Health conditions
flexible	decision (d)	Rank

Table 12. Experiment Result

Table 12 shows the attributes names, descriptions, and the partitions. All attributes are categorical in the data set. If a data set contains continuous data, one can use a discretization method (such as Rosetta [9]) to convert them to categorical data. Table 13 shows the parameters used in our experiment, the number of action rules generated, and the time required to complete the task.

Experiment No.	Num. of objects	Stab. attribute	Flex. attribute	Sup	Conf	Num. of action rules	Computation time (in seconds)
1	12960	4	5	5%	85%	86	2.36
2	12960	4	5	10%	85%	53	1.07
3	12960	4	5	15%	85%	12	0.88

Table 13. Experiment Result

Table 14 shows the action rules generated during experiment No. 3. The first action rule can be read as, if the housing standing of the family changes from *convenient* to *inconvenient*, then the decision changes from *not recommend* to *priority* with support value of 2022 and confidence of 0.9. The overall result shows that the change of rank (decision attribute) is strongly related to the changes in financial standing of the family and health conditions in experiment 3.

condition	decision	sup	conf
$(f)convenient \rightarrow (d)notrecom$	$\mapsto (f)inconvenient \rightarrow (d)priority$	2022	0.9
$(f)convenient \rightarrow (d)notrecom$	$\mapsto (f)inconvenient \rightarrow (d)spec\_prior$	2160	1
$(f)convenient \rightarrow (d)priority$	$\mapsto (f)inconvenient \rightarrow (d)notrecom$	2160	1
$(f)convenient \rightarrow (d)priority$	$\mapsto (f)inconvenient \rightarrow (d)spec\_prior$	2188	1
$(f)inconvenient \rightarrow (d)notrecom$	$\mapsto (f)convenient \rightarrow (d)priority$	2160	1
$(f)inconvenient \rightarrow (d)priority$	$\mapsto (f)convenient \rightarrow (d)notrecom$	2022	1
$(f)inconvenient \rightarrow (d)spec\_prior$	$\mapsto (f)convenient \rightarrow (d)notrecom$	2160	1
$(f)inconvenient \rightarrow (d)spec\_prior$	$\mapsto (f)convenient \rightarrow (d)priority$	2188	1
$(t)priority \rightarrow (d)spec\_prior$	$\mapsto (t)notrecom \rightarrow (d)notrecom$	2466	1
$(t)priority \rightarrow (d)spec\_prior$	$\mapsto (t)recommended \rightarrow (d)priority$	2412	1
$(t)recommended \rightarrow (d)priority$	$\mapsto (t)notrecom \rightarrow (d)notrecom$	2412	1
$(t)recommended \rightarrow (d)priority$	$\mapsto (t)priority \rightarrow (d)spec\_prior$	2412	1

Table 14. Action Rules

5 Conclusion and Future Work

We presented an algorithm that discovers action rules from a decision table. The proposed algorithm generates a complete set of shortest action rules without using pre-existing classification rules. During the experiment with several data sets, we noticed that the flexibility of attributes are not equal. For example, the social condition is most likely less flexible than the health condition in the data set used in our experiment, and this may have to considered. Future work shall address this issue as well as further analysis of the algorithm with real world data sets.

6 Acknowledgment

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

References

1. R. Agrawal, R. Srikant (1994), Fast algorithm for mining association rules, Proceeding of the Twentieth International Conference on VLDB, 487-499
2. A. Dardzińska, Z. Raś (2006), Extracting rules from incomplete decision systems, in Foundations and Novel Approaches in Data Mining, Studies in Computational Intelligence, Vol. 9, Springer, 143-154
3. D. Fensel (1998), Ontologies: a silver bullet for knowledge management and electronic commerce, Springer-Verlag
4. H. Geffner, J. Wainer (1998), Modeling action, knowledge and control, ECAI, 532-536
5. S. Greco, B. Matarazzo, N. Pappalardo, R. Slowiński, Measuring expected effects of interventions based on decision rules, J. Exp. Theor. Artif. Intell., Vol. 17, No. 1-2, 103-118
6. J. Grzymala-Busse (1997), A new version of the rule induction system LERS, Fundamenta Informaticae, Vol. 31, No. 1, 27-39
7. Z. He, X. Xu, S. Deng, R. Ma, (2005) Mining action rules from scratch, Expert Systems with Applications, Vol. 29, No. 3, 691-699

8. S. Hettich, C.L. Blake, C.J. Merz (1998), UCI Repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/MLRepository.html, University of California, Irvine, Dept. of Information and Computer Sciences
9. A. Øhrn, J. Komorowski (1997), ROSETTA: A Rough Set Toolkit for Analysis of Data
10. Z. Pawlak (1981) Information systems - theoretical foundations, Information Systems Journal, Vol. 6, 205-218
11. Y. Qiao, K. Zhong, H.-A. Wang and X. Li (2007), Developing event-condition-action rules in real-time active database, Proceedings of the 2007 ACM symposium on Applied computing, ACM, New York, 511-516
12. Z.W. Raś, A. Dardzińska (2006), Action rules discovery, a new simplified strategy, Foundations of Intelligent Systems, LNAI, No. 4203, Springer, 445-453
13. Z.W. Raś, A. Tzacheva, L.-S. Tsay, O. Gürdal (2005), Mining for interesting action rules, Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2005), Compiegne University of Technology, France, 2005, 187-193
14. Z.W. Raś, A. Wieczorkowska (2000), Action-Rules: How to increase profit of a company, in Principles of Data Mining and Knowledge Discovery, Proceedings of PKDD 2000, Lyon, France, LNAI, No. 1910, Springer, 587-592
15. Z. Raś, E. Wyrzykowska, H. Wasyluk (2007), ARAS: Action rules discovery based on agglomerative strategy, in Mining Complex Data, Post-Proceedings of 2007 ECML/PKDD Third International Workshop (MCD 2007), LNAI, Vol. 4944, Springer, 2008, 196-208
16. A. Skowron (2001), Rough sets and boolean reasoning, in Granular Computing: an Emerging Paradigm, Physica-Verlag, 95-124
17. L.-S. Tsay, Z.W. Raś (2006), Action rules discovery system DEAR3, in Foundations of Intelligent Systems, Proceedings of ISMIS 2006, Bari, Italy, LNAI, No. 4203, Springer, 483-492
18. A. Tzacheva, Z.W. Ras (2007), Constraint based action rule discovery with single classification rules, in Proceedings of the Joint Rough Sets Symposium (JRS07), LNAI, Vol. 4482, Springer, 322-329