Discovering E-Action Rules from Incomplete Information Systems

Li-Shiang Tsay and Zbigniew W. Raś

***Abstract*— Action rules, interventions, and E-action rules are examples of knowledge discovery tools for reclassification of objects and for defining actionability as a partially objective concept. However, most of these tools can only deal with incompleteness represented as null values. The purpose of knowledge discovery systems is to extract knowledge which is interesting and often interestingness is linked with actionability. In this paper, we present a new algorithm, DEAR_3, to discover actionability knowledge from an incomplete information system.** [1]

I. Introduction

E-action rules [5], [6], are a key tool used for extracting higher level actionable information from large volumes of data. It can be applied in many real-life applications as a powerful solution to a reclassification problem. The basic principle of reclassification is a process of learning a function that maps a class of objects into another class by changing values of some of the classification attributes describing that class. The classification attributes are divided into stable and flexible. Saying another words, reclassification is a process of showing what changes in values in some of the flexible attributes for a given class of objects are needed in order to shift them from one decision class to another more desired one.

E-action rule is a rule of the form $[(\omega) \wedge (\alpha \rightarrow \beta)] \Rightarrow [\phi \rightarrow \psi]$, where $\omega, (\alpha \rightarrow \beta)$, and $[\phi \rightarrow \psi]$ are events. It states that when the fixed condition ω is satisfied and the changeable behavior $(\alpha \rightarrow \beta)$ occurs in a database tuples so does the expectation$[\phi \rightarrow \psi]$. Support and confidence are used to measure the importance of each rule to avoid generating irrelevant, spurious, and insignificant rules. Any E-action rule forms workable strategy that can be used in a business decision making process to increase profit, reduce cost, etc. Each E-action rule can be constructed by comparing pairs of previously discovered classification rules from a given decision system. The concept of E-action rule was introduced in [9] to enhance action rules [4] and extended action rules [5],[7], and [8] to extract actionable knowledge from databases containing nominal data. Several efficient algorithms for mining E-action rules have been developed [7], [8], [6], and [9]. In all these papers, mining for action rules from a complete data is well understood and investigated on both the algorithmic and conceptual level.

[1]Manuscript received Dec. 28, 2005.

Li-Shiang Tsay is with the Department of Computer Science, Hampton University, Hampton, VA 23668, USA (phone: 757-727-5659; e-mail: li-shiang.tsay@hamptonu.edu).

Zbigniew W. Raś is with the Computer Science Department, University of North Carolina, Charlotte, NC 28223, USA (phone: 704-678-8574; e-mail: ras@uncc.edu).

However, many real-world databases contain incomplete data and current methods for action rules construction in such cases are so far inadequate. The source of incompleteness can vary and can be linked with recording error, missing values, or uncertainty in classifying information.

In this paper, we present a new algorithm DEAR_3 for discovering action rules. It consists of several basic steps: process of discovering classification rules, analyzing them, and process of action rules construction. The development of workable strategies for implementing them is naturally linked with the last step. We also propose a novel method *CID* for generating classification rules from an incomplete information system. The definition of an incomplete information system that allows to have non-singleton subsets as values of attributes is given. After forming classification rules, there are two possible options to construct E-action rules: one based on action forest [9] and the other one based on action tree [6]. They both speed up the process of E-action rule construction. They differ in the splitting criterion applied at every node of a generated tree. In this paper, action-tree algorithm is utilized. We focus on mining E-action rules, since they are more meaningful, simpler, easier to interpret, understand, and to apply than classical action rules [9].

II. Incomplete Information Systems and E-Action Rules

By an incomplete information system (IS), introduced in [2], we mean $IS = (U, A, V)$, where:

- U is a finite set of objects,
- A is a finite set of attributes, and
- $V = \bigcup\{V_a : a \in A\}$ is a finite set of values of attributes. The set V_a is a domain of attribute a. Additionally we assume that for each attribute $a \in A$ and $x \in U$, $a(x) = \{(a_i, p_i) : i \in J_{a(x)} \wedge (\forall i \in J_{a(x)})[a_i \in V_a] \wedge \sum p_i = 1\}$.

In this paper, for the purpose of clarity, objects are interpreted as customers. Attributes are interpreted as features such as, offers made by a bank, characteristic conditions etc.

A user's knowledge about objects is often not precise. Our incomplete information system allows to use a weighted set of values of attributes as a value of an attribute. For instance, the set (brown, 60%),(black, 40%) can be seen as a value of the attribute eye color. If the incompleteness corresponds to the lack of information about the value of a particular attribute (null value), then a set of all equally weighted attribute values of that attribute can be used as its value.

TABLE I
AN INCOMPLETE DECISION SYSTEM (IS)

U	a	b	c	d
x_1	$(0, 0.33), (2, 0.67)$	$(1, 1)$		$(L, 1)$
x_2	$(0, 0.25), (1, 0.75)$		$(1, 0.33), (2, 0.67)$	$(L, 1)$
x_3		$(1, 0.5), (2, 0.5)$	$(1, 1)$	$(L, 1)$
x_4	$(1, 1)$	$(3, 1)$	$1, 0.5), (3, 0.5)$	$(H, 1)$
x_5	$(3, 0.67), (2, 0.33)$		$(1, 1)$	$(H, 1)$
x_6	$(2, 1)$	$(3, 0.9), (1, 0.1)$	$(2, 1)$	$(L, 1)$
x_7		$(1, 0.33), (4, 0.67)$	$1, 0.25), (2, 0.75)$	$(H, 1)$
x_8	$(1, 1)$	$(1, 1)$	$(2, 1)$	$(H, 1)$
x_9	$(2, 0.8), (1, 0.2)$	$(3, 1)$		$(L, 1)$

We only consider a special type of an information system, called a decision table [3]. A decision table consists of a set of objects where each object is described by a set of attributes. The set of attributes is partitioned into conditions and decisions. Additionally, we assume that the set of conditions is partitioned into stable conditions and flexible conditions. For simplicity reason, we assume that there is only one decision attribute. *Date of birth* is an example of a stable attribute. The *interest rate* on any customer account is an example of a flexible attribute as the bank can adjust rates. We adopt the following definition of a decision table:

By a decision table we mean any information system of the form $IS = \{U, A_{St} \cup A_{Fl} \cup \{d\}, V\}$, where:

- U is a set of objects,
- $A_{St} = \{a_{St}[i] : 1 \leq i \leq I\}$ is a set of stable attributes, and $V_i^{St} = \{a_{St}[i, j] : 1 \leq j \leq J(i)\}$ is the domain of attribute $a_{St}[i]$.
- $A_{Fl} = \{a_{Fl}[i] : 1 \leq i \leq L\}$ is a set of flexible attributes, and $V_i^{Fl} = \{a_{Fl}[i, j] : 1 \leq j \leq L(i)\}$ is the domain of attribute $a_{Fl}[i]$.
- d is a decision attribute where $V_d = \{d[m] : 1 \leq m \leq M\}$.

For simplicity reason, the term $a(i_1, j_1) \cdot a(i_2, j_2) \cdot ... \cdot a(i_r, j_r)$ is denoted by $[a(i_k, j_k)]_{k \in \{1,2,...,r\}}$, where all i_1, i_2, ... , i_r are distinct values and $j_p \leq J(i_p), 1 \leq p \leq r$.

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

By $L(r)$ we mean all attributes listed in the IF part of a rule r. For example, if $r_1 = [(a_1, 2) \wedge (a_2, 1) \wedge (a_3, 4) \longrightarrow (d, 8)]$ is a rule then $L(r_1) = \{a_1, a_2, a_3\}$.

By $d(r_1)$ we denote the decision value of that rule. In our example $d(r_1) = 8$. 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 $r_2 = [(a_2, 1) * (a_3, 4) \rightarrow (d, 1)]$, then $r_1/\{a_2, a_3\}$ = $r_2/\{a_2, a_3\}$.

Now, let us assume that $(a, v \rightarrow w)$ denotes the fact that the value of attribute a has been changed from v to w [4]. Similarly, the term $(a, v \rightarrow w)(x)$ 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 $IS = (U, A_{St} \cup A_{Fl} \cup \{d\})$ is a decision table and rules r_1, r_2 have been extracted from IS. The notion of E-action rule was given in [9]. Its definition is given below. We assume here that:

- B_{St} is a maximal subset of A_{St} such that $r_1/B_{St} = r_2/B_{St}$,
- $d(r_1) = k_1$, $d(r_2) = k_2$ and $k_1 \leq k_2$,
- $(\forall a \in [A_{St} \cap L(r_1) \cap L(r_2)])[a(r_1) = a(r_2)]$,
- $(\forall i \leq q)(\forall e_i \in [A_{St} \cap [L(r_2) - L(r_1)]])[e_i(r_2) = u_i]$,
- $(\forall i \leq r)(\forall c_i \in [A_{Fl} \cap [L(r_2) - L(r_1)]])[c_i(r_2) = t_i]$,
- $(\forall i \in p)(\forall b_i \in [A_{Fl} \cap L(r_1) \cap L(r_2)])[[b_i(r_1) = v_i] \& [b_i(r_2) = w_i]]$.

Let $A_{St} \cap L(r_1) \cap L(r_2) = F$ and $\prod\{a = a(r_1) : a \in F\} = G$. By (r_1, r_2)-E-action rule on $x \in U$ we mean the expression r:

$[G \wedge (e_1 = u_1) \wedge (e_2 = u_2) \wedge ... \wedge (e_q = u_q) \wedge (b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge ... \wedge (b_p, v_p \rightarrow w_p) \wedge (c_1, \rightarrow t_1) \wedge (c_2, \rightarrow t_2) \wedge ... \wedge (c_r, \rightarrow t_r)](x) \rightarrow [(d, k_1 \rightarrow k_2)](x)$

Object $x \in U$ supports (r_1, r_2)-E-action rule r in $IS = (U, A_{St} \cup A_{Fl} \cup \{d\})$, if there exists $y \in U$ and the following conditions are satisfied:

- if $L(r) = \{b_1, b_2, ..., b_p\}$, then $(\forall i \leq p)[[b_i(x) = v_i] \wedge [d(x) = k_1] \wedge [b_i(y) = w_i] \wedge [d(y) = k_2]]$
- if $A_{St} \cap L(r_2) = \{a_1, a_2, ..., a_q\}$, then $(\forall j \leq q)[[a_j(x) = u_j] \wedge [a_j(y) = u_j]]$
- object x supports rule r_1
- object y supports rule r_2

By the support of E-action rule r in IS, denoted by $Sup_{IS}(r)$, we mean the set of all objects in IS supporting r. In other words, $Sup_{IS}(r)$ contains objects in U having the property $G \wedge (e_1 = u_1) \wedge (e_2 = u_2) \wedge ... \wedge (e_p = u_p) \wedge (b_1 = v_1) \wedge (b_2 = v_2) \wedge ... \wedge (b_p = v_p) \wedge (d = k_1)$.

By the confidence of r in IS, denoted by $Conf_{IS}(r)$, we mean $[Sup_{IS}(r)/Sup_{IS}(L(r))] \cdot [Conf(r_2)]$.

III. DISCOVERING CLASSIFICATION RULES

There is a number of rule extraction algorithms such as, LERS, Rosetta, C4.5, and ERID[1]. However none of these systems, except ERID, can handle partially incomplete data as introduced in this paper. We present a new method, called *C*lassification rules discovery for an *I*ncomplete *D*ecision system(***CID***) which is conceptually similar to LERS and ERID.

CID method is based on granular computing and roulette-wheel technique to reveal interesting information from incomplete data sets. It consists of four main steps given below.

1) Replace missing values by promising values
2) Generate partition of objects (granules) and compute their support.
3) Evaluate inclusion relation
4) Determine whether to terminate the learning process or to build new terms and go back to step 2

Referring back to the incomplete decision system IS represented by Table I, let us discover rules from it using the above

Fig. 1. A roulette wheel with 4 slices. The size of each slice corresponds to the frequency of the individual value of the attribute a.

TABLE II
AN INCOMPLETE DECISION SYSTEM (IS)

U	a	b	c	d
x_1	$(0, 0.33), (2, 0.67)$	$(1, 1)$	$(1, 0.25), (2, 0.75)$	$(L, 1)$
x_2	$(0, 0.25), (1, 0.75)$	$(1, 0.5), (3, 0.5)$	$(1, 0.33), (2, 0.67)$	$(L, 1)$
x_3	$(1, 0.25), (2, 0.75)$	$(1, 0.5), (2, 0.5)$	$(1, 1)$	$(L, 1)$
x_4	$(1, 1)$	$(3, 1)$	$(1, 0.5), (3, 0.5)$	$(H, 1)$
x_5	$(3, 0.67), (2, 0.33)$	$(3, 0.75), (4, 0.25)$	$(1, 1)$	$(H, 1)$
x_6	$(2, 1)$	$(3, 0.9), (1, 0.1)$	$(2, 1)$	$(L, 1)$
x_7	$(1, 0.5), (2, 0.5)$	$(1, 0.33), (4, 0.67)$	$(1, 0.25), (2, 0.75)$	$(H, 1)$
x_8	$(1, 1)$	$(1, 1)$	$(2, 1)$	$(H, 1)$
x_9	$(2, 0.8), (1, 0.2)$	$(3, 1)$	$(1, 0.5), (2, 0.5)$	$(L, 1)$

algorithm. We assume here that the minimum support is 1 and the minimum confidence is 65%.

Step 1: **Replace missing values by promising values**. Replacing null values by randomly picking up two promising values using a roulette-wheel technique. Suppose that we have four distinct values for an attribute a such as 0, 1, 2, and 3, and their corresponding frequencies are 0.58, 2.95, 2.8, and 0.67 respectively. The frequency corresponding to v_i is defined as the weighted number of objects in IS having property (a, v_i). Each value v_i of the attribute a has a slice of the roulette wheel allocated in proportion to their frequency (see figure 1). How many times we spin the wheel is determined by the number of distinct attribute values of an attribute. Therefore, in our case, we spin the wheel four times and keep a tally of the result. Let us assume that for object x_3, values 0, 1, 2, and 3 are chosen 0, 1, 3, and 0 times, respectively. Then, we chose attribute values 1 and 2 to replace a null value in the attribute a for object x_3. The weight of the selected value 1 is $0.25 : 1/(1 + 3)$, and the weight of the selected value 2 is $0.75 : 3/(1 + 3)$. For every null value, we follow the same process to construct two-element set of weighted attribute values and use it for replacing that missing value. After imputing all missing values, our incomplete decision system IS is represented as Table II.

Step 2: **Generate partition of objects (granules) and compute their support** Partition the set of objects into weighted granules with respect to each value of each attribute and calculate the support of each granule. If a granule $CLASS_{IS}a[i]$ is below minimum support, then it is marked as negative. When a granule has negative mark, then it is not considered in later steps.

Weighted granules for the value of a decision attribute:

$CLASS_{IS}(d, L) = \{(x_1, 1), (x_2, 1), (x_3, 1), (x_6, 1), (x_9, 1)\}$,

$CLASS_{IS}(d, H) = \{(x_4, 1), (x_5, 1), (x_7, 1), (x_8, 1)\}$.

Weighted granules for the value of classification attributes:

$CLASS_{IS}(a, 0) = \{(x_1, 0.33), (x_2, 0.25)\}$- marked **negative**

$CLASS_{IS}(a, 1) = \{(x_2, 0.75), (x_4, 1), (x_8, 1), (x_9, 0.2), (x_3, 0.25), (x_7, 0.5)\}$

$CLASS_{IS}(a, 2) = \{(x_1, 0.67), (x_3, 0.75), (x_5, 0.33), (x_6, 1), (x_7, 0.5), (x_9, 0.8)\}$

$CLASS_{IS}(a, 3) = \{(x_5, 0.67)\}$ - marked **negative**

$CLASS_{IS}(b, 1) = \{(x_1, 1), (x_2, 0.5), (x_3, 0.5), (x_6, 0.5), (x_7, 0.33), (x_8, 1)\}$

$CLASS_{IS}(b, 2) = \{(x_3, 0.5)\}$- marked **negative**

$CLASS_{IS}(b, 3) = \{(x_2, 0.5), (x_4, 1), (x_5, 0.75), (x_6, 0.9), (x_9, 1)\}$

$CLASS_{IS}(b, 4) = \{(x_7, 0.67)\}$ - marked **negative**

$CLASSIS(c, 1) = \{(x_1, 0.25), (x_2, 0.33), (x_3, 1), (x_4, 0.5), (x_5, 1), (x_7, 0.25), (x_9, 0.5)\}$

$CLASSIS(c, 2) = \{(x_1, 0.75), (x_6, 1), (x_7, 0.75), (x_8, 1), (x_9, 0.5)\}$

$CLASS_{IS}(c, 3) = \{(x_4, 0.5)\}$ - marked **negative**

$CLASS_{IS}(c, 4) = \{(x_2, 0.67)\}$ - marked **negative**

Step 3: **Evaluate inclusion relation**. The relationship between each classification granule, $CLASS_{IS}a[i]$, and a decision granule, $CLASS_{IS}d[m]$ is checked in this step. If the support of $a[i] \rightarrow d[m]$ is below the threshold, it is marked as negative. Otherwise, it is marked as positive. When a granule is marked as negative it means that the corresponding relationship $CLASS_{IS}a[i] \subseteq CLASS_{IS}d[m]$ does not hold and it is not considered any further. The positive mark means that the rule $a[i] \rightarrow d[m]$ is approved and the corresponding relationship $CLASS_{IS)}a[i] \subseteq CLASS_{IS}d[m]$ holds. The computation of support and confidence is illustrated in the following example.

$CLASS_{IS}[(a, 1) * (d, L)] = \{(x_2, (0.75 \times 1)), (x_3, (0.25 \times 1)), (x_9, (0.2 \times 1))\}$

$(a, 1) \rightarrow (d, L)$; (sup = $1.2 \geq 1$; conf = $1.2/3.7 < 0.65$)

$CLASS_{IS}[(a, 2) * (d, L)] = \{(x_1, 0.67), (x_3, 0.75), (x_6, 1), (x_9, 0.8)\}$

$(a, 2) \rightarrow (d, L)$; ($sup = 3.22 \geq 1$; conf = $0.80 \geq 0.65$) - marked **positive**

$CLASS_{IS}[(b, 1) * (d, L)] = \{(x_1, 1), (x_2, 0.5), (x_3, 0.5), (x_6, 0.5)\}$

$(b, 1) \rightarrow (d, L)$; (sup = $2.5 \geq 1$; conf = $0.649 < 0.65$)

$CLASS_{IS}[(b, 3) * (d, L)] = \{(x_2, 0.5)(x_6, 0.9), (x_9, 1)\}$

$(b, 3) \rightarrow (d, L)$; (sup = $2.4 \geq 1$; conf: $0.516 < 0.65$)

$CLASS_{IS}[(c, 1) * (d, L)] = \{(x_1, 0.25), (x_2, 0.33), (x_3, 1), (x_9, 0.5)\}$

$(c, 1) \rightarrow (d, L)$; (sup = $2.08 \geq 1$; conf = $0.543 < 0.65$)

$CLASS_{IS}[(c, 2) * (d, L)] = \{(x_1, 0.75), (x_6, 1), (x_9, 0.5)\}$

$(c, 2) \rightarrow (d, L)$; (sup = $2.25 \geq 1$; conf = $0.563 < 0.65$)

$CLASS_{IS}[(a, 1) * (d, H)] = \{(x_4, 1), (x_7, 0.5), (x_8, 1)\}$

$(a, 1) \rightarrow (d, H)$;(sup= $2.5 \geq 1$; conf = $0.68 \geq 0.65$) - marked **positive**

$CLASS_{IS}[(a, 2) * (d, H)] = \{(x_5, 0.33), (x_7, 0.5)\}$

$(a, 2) \rightarrow (d, H)$; (sup = $0.83 < 1$)- marked **negative**

$CLASS_{IS}[(b, 1) * (d, H)] = \{(x_7, 0.33), (x_8, 1)\}$

$(b, 1) \rightarrow (d, H)$; (sup = $1.33 \geq 1$; conf = $0.35 < 0.65$)

$CLASS_{IS}[(b, 3) * (d, H)] = \{(x_4, 1), (x_5, 0.75)\}$

$(b, 3) \rightarrow (d, H)$; (sup = $1.75 \geq 1$; conf = $0.42 < 0.65$)

$CLASS_{IS}[(c, 1) * (d, H)] = \{(x_1, 0.75), (x_6, 1), (x_9, 0.5)\}$

$(c, 1) \rightarrow (d, H)$; (sup = $2.08 \geq 1$; conf = $0.543 < 0.65$)

$CLASS_{IS}[(c, 2) * (d, H)] = \{(x_1, 0.75), (x_6, 1), (x_9, 0.5)\}$

$(c, 2) \rightarrow (d, H)$; (sup = $2.25 \geq 1$; conf = $0.563 < 0.65$)

Step 4: **Determine whether to terminate the learning process or build new terms one attribute value longer than current terms**. We build new terms from two unmarked terms only if their left-hand side attribute values differ only on the last value and the right hand side attribute values are the same. Then, we repeat steps 2 and 3. If all terms are marked, the algorithm is terminated.

The formation of the two element terms is illustrated in the following example.

$CLASS_{IS}((a, 1) * (b, 3)) = \{(x_2, 0.375), (x_9, 0.2)\}$:- marked **negative**

TABLE III

SET OF RULES R WITH SUPPORTING OBJECTS

$Objects$	a	b	c	d
$\{x_1, x_3, x_6, x_9\}$	2			L
$\{x_4, x_7, x_8\}$	1			H
$\{x_2, x_6\}$		3	2	L
$\{x_4, x_5\}$		3	1	H

$CLASS_{IS}((a,1)*(c,1)) = \{(x_2, 0.2475), (x_4, 0.5), (x_9, 0.1)\}$;-marked **negative**

$CLASS_{IS}((a,1)*(c,2)) = \{(x_7, 0.375), (x_8, 1), (x_9, 0.1)\}$; (sup =0.665 ¿ 1)

$(a,1) \wedge (c,2) \rightarrow (d,H)$; (sup = $0.1 < 1$)-marked **negative**

$CLASS_{IS}((b,3)*(c,1)) = \{(x_2, 0.165), (x_4, 0.5), (x_5, 0.75), (x_9, 0.5)\}$;

$(b,3) \wedge (c,1) \rightarrow (d,L)$; (sup = $0.665 < 1$)- marked **negative**

$CLASS_{IS}((b,3)*(c,2)) = \{(x_6, 0.9), (x_9, 0.5)\}$;

$(b,3) \wedge (c,2) \rightarrow (d,L)$; (sup = $1.4 \geq 1$; conf = 1)- marked **positive**

$CLASS_{IS}((b,1)*(c,1)) = \{(x_1, 0.25), (x_2, 0.165), (x_3, 0.5), (x_7, 0.0825)\}$;

(sup =0.9975 ¡ 1)- marked **negative**

$CLASS_{IS}((b,1)*(c,2)) = \{(x_1, 0.75), (x_3, 0.2475), (x_6, 0.5), (x_8, 1)\}$;

$(b,1) \wedge (c,2) \rightarrow (d,H)$; (sup = $1 \geq 1$; conf = 0.4 ¡ 1)

$CLASS_{IS}((b,3)*(c,1)) = \{(x_2, 0.165), (x_4, 0.5), (x_5, 0.75), (x_9, 0.5)\}$;

$(b,3) \wedge (c,1) \rightarrow (d,H)$; (sup = $1.25 \geq 1$; conf = $0.6527 \geq 0.65$)-marked **positive**

$CLASS_{IS}((b,3)*(c,2)) = \{(x_6, 0.9), (x_9, 0.5)\}$;

$(b,3) \wedge (c,2) \rightarrow (d,H)$; (sup = 0)- marked **negative**

Since only $(b,1)*(c,2)$ is unmarked, we can not extend the term any further. It can be easily checked that the following two optimal classification rules are built:

$r_1 = [(a,2) \rightarrow (d,L)]$, $r_2 = [(a,1) \rightarrow (d,H)]$,

$r_3 = [(b,3) \wedge (c,2) \rightarrow (d,L))]$, $r_4 = [(b,3) \wedge (c,1) \rightarrow (d,H)$.

IV. DISCOVERING E-ACTION RULES

After forming classification rules, Action-Tree algorithm [6] is utilized to construct e-action rules. It is based on a divide-and-conquer approach to the re-classification problem. Assume now that our goal is to re-classify objects from the class $d^{-1}(\{d_i\})$ into a new class $d^{-1}(\{d_j\})$. In our example, we assume that $d_i = (d,L)$ and $d_j = (d,H)$.

First, we represent the set R of optimal rules extracted from IS as a table (see Table III). The first column of this table shows objects in IS supporting the rules from R (each row represents a rule). For instance, the first row represents the rule $[(a,2) \Rightarrow (d,L)]$. The construction of an action tree starts with the set R as Table III representing the root of the tree (T_1 in Fig. 2). The root node selection is based on a stable attribute with the smallest number of values among all stable attributes. The same strategy is used for a child node selection. After labeling the nodes of the tree by all stable attributes, the tree is split based on the value of the decision attribute. In Table I, there is only one stable attribute b and only one value involved so we use stable attribute b to expand the tree one more level defined by it. Corresponding edge is labeled by the value 3 of attribute b. All rules in the sub-table T_2 do not contain any stable attributes but do contain different decision values so it is divided into two new sub-tables. Each leaf represents a set of rules which do not contradict on stable attributes and also define decision value d_i.

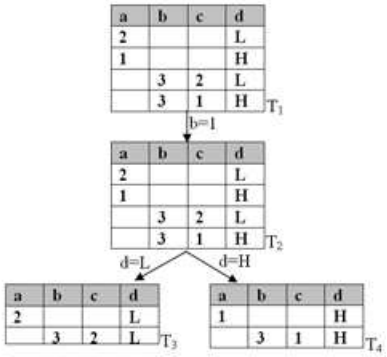

Fig. 2. Action-tree

When we follow the path labeled by value $[b = 3]$ and $[d = L]$, we get table T_3. Then, by following the path labeled by $[b = 3]$ and $[d = H]$, we get table T_4. Now, we can compare pairs of rules belonging to these two resulting tables and generate two E-action rules given below:

$[[(a, 2 \rightarrow 1) \Rightarrow (d, L \rightarrow H)]$, and

$[[(b,3) \wedge (c, 2 \rightarrow 1)] \Rightarrow (d, L \rightarrow H)]$.

After the rules are formed, we evaluate them by checking their support and confidence.

V. CONCLUSION

Most of the collected real-world data sets are incomplete. In this paper we presented a new method for extracting actionable knowledge from incomplete data. By incomplete system we mean a system which allows to have sets of weighted values of an attribute as an attribute value assigned to an object. Algorithm DEAR_3, presented here, is a significant improvement of our previous system DEAR_2 [8] for E-action rules discovery.

REFERENCES

[1] Dardzińska A, Raś Z W (2003) On Rule Discovery from Incomplete Information Systems. In Proceedings of IEEE ICDM'03 Workshop on Foundations and New Directions of Data Mining, 31–35

[2] Pawlak Z (1981) Information systems - theoretical foundations. In: Information Systems Journal, Vol. 6, 205–218

[3] Pawlak Z (1985) Rough Sets and Decision Tables. In: Lecture Notes in Computer Science 208, Springer-Verlag, 186–196

[4] Raś Z, Wieczorkowska A (2000) Action rules: how to increase profit of a company. Proceedings of PKDD'00, Lyon, France, LNCS/LNAI, No. 1910, Springer-Verlag, 587–592

[5] Raś Z W, Tsay L-S (2003) Discovering extended action-rules (System DEAR). In: Proceedings of the IIS'2003 Symposium, Zakopane, Poland, Advances in Soft Computing, Springer-Verlag, 293–300

[6] Raś Z W, Tsay L-S, A. Dardzinska (2005) Mining E-Action Rules. In: Mining Complex Data, Proceedings of 2005 IEEE ICDM Workshop in Houston, Texas, Published by Math. Dept., Saint Mary's Univ., Nova Scotia, Canada, 2005, 85-90

[7] Tsay L-S, Raś Z W, and A. Wieczorkowska (2004) Tree-based algorithm for discovering extended action-rules (System DEAR2). In: Proceedings of the IIS'2004 Symposium, Zakopane, Poland, Springer, 459-464

[8] Tsay L-S, Raś Z W (2005) Action rules discovery: System DEAR2, method and experiments. Journal of Experimental and Theoretical Artificial Intelligence, Taylor Francis, Vol. 17, No. 1-2, 119-128

[9] Tsay L-S, Raś Z W (2006) E-Action Rules. In: Foundations of Data Mining, Studies in Computational Intelligence, Springer, 2006, will appear