Action Rules Discovery without pre-existing classification rules

Zbigniew W. Raś[1,2] and Agnieszka Dardzińska[3,1]

[1] Univ. of North Carolina, Dept. of Computer Science, Charlotte, NC, 28223, USA
[2] Polish Academy of Sciences, Institute of Computer Science, 01-237 Warsaw, Poland
[3] Bialystok Technical Univ., Dept. of Computer Science, 15-351 Bialystok, Poland
e-mail: ras@uncc.edu, adardzin@uncc.edu

Abstract. Action rules describe possible transitions of objects from one state to another with respect to a distinguished attribute. Previous research on action rule discovery usually requires the extraction of classification rules before constructing any action rule. In this paper, we present a new algorithm that discovers action rules directly from a decision system. It is a bottom-up strategy which has some similarity to systems *ERID* and *LERS*. Finally, it is shown how to manipulate the music score using action rules.

1 Introduction

An action rule is a rule extracted from a decision system that describes a possible transition of objects from one state to another with respect to a distinguished attribute called a decision attribute [13]. We assume that attributes used to describe objects in a decision system are partitioned into stable and flexible. Values of flexible attributes can be changed. This change can be influenced and controlled by users. Action rules mining initially was based on comparing profiles of two groups of targeted objects - those that are desirable and those that are undesirable [13]. An action rule is formed as a term $[(\omega) \wedge (\alpha \rightarrow \beta)] \Rightarrow (\phi \rightarrow \psi)$, where ω is a conjunction of fixed condition features shared by both groups, $(\alpha \rightarrow \beta)$ represents proposed changes in values of flexible features, and $(\phi \rightarrow \psi)$ is a desired effect of the action. The discovered knowledge provides an insight of how relationships should be managed so the undesirable objects can be changed to desirable. For example, in society, one would like to find a way to improve his or her salary from a low-income to a high-income. Another example in business area is when an owner would like to improve his or her company's profits by going from a high-cost, low-income business to a low-cost, high-income business.

Action rules introduced in [13] has been further investigated in [15][12][14]. Paper [5] 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 [16]. 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 [6], is based on Pawlak′s model of an information system S [9]. The goal is to identify certain relationships between granules defined by the indiscernibility relation on its objects. Some of these relationships uniquely define action rules for S.

This paper presents a new strategy for discovering action rules directly from the decision system. Action rules are built from atomic expressions following a strategy similar to *ERID* [2].

2 Background and Objectives

In this section we introduce the notion of an information system, a decision system, stable attribute, flexible attribute, and give some examples.

By an information system [9] we mean a triple $S = (X, A, V)$, where:

1. X is a nonempty, finite set of objects
2. A is a nonempty, 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
3. $V = \bigcup\{V_a : a \in A\}$.

For example, Table 1 shows an information system S with a set of objects $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$, a set of attributes $A = \{a, b, c, d\}$, and a set of their values $V = \{a_1, a_2, b_1, b_2, b_3, c_1, c_2, d_1, d_2, d_3\}$.

	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	b_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	c_2	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

An information system $S = (X, A, V)$ is called a decision system, if $A = A_{St} \cup A_{Fl} \cup \{d\}$, where d is a distinguished attribute 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 [13][15][12][14], 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* [4] or *ERID* [2], then, construct action rules either from certain pairs of these rules or from a single classification rule. For instance, algorithm $ARAS$ [14] 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 a decision system without using pre-existing classification rules.
2. Extract action rules that have minimal attribute involvement.

To meet these two goals, we introduce the notion of atomic action terms and show how to build action rules from them.

3 Action Rules

In this section we give a definition of action terms, action rules, and we propose their interpretation which we call standard.

Let $S = (X, A \cup \{d\}, V)$ be a decision system, where $V = \bigcup\{V_a : a \in A\}$. First, we introduce the notion of an action term.

By an *atomic action term* we mean an expression $(a, a_1 \rightarrow a_2)$, where a is an attribute and $a_1, a_2 \in V_a$. If $a_1 = a_2$, then a is called stable on a_1.

By a set of *action terms* we mean a smallest set such that:

1. If t is an atomic action term, then t is an action term.
2. If t_1, t_2 are action terms, then $t_1 \star t_2$ is an action term.
3. If t is an action term containing $(a, a_1 \rightarrow a_2)$, $(b, b_1 \rightarrow b_2)$ as its sub-terms, then $a \neq b$.

By the domain of an action term t, denoted by $Dom(t)$, we mean the set of all attribute names listed in t.

By an *action rule* we mean an expression $r = [t_1 \Rightarrow t_2]$, where t_1 is an action term and t_2 is an atomic action term. Additionally, we assume that $Dom(t_2) = \{d\}$ and $Dom(t_1) \subseteq A$. The domain of action rule r is defined as $Dom(t_1) \cup Dom(t_2)$.

Now, let us give an example of action rules assuming that the decision system S is represented by Table 1, a is stable and b, c are flexible attributes. Expressions $(a, a_2 \rightarrow a_2)$, $(b, b_1 \rightarrow b_3)$, $(c, c_2 \rightarrow c_2)$, $(d, d_1 \rightarrow d_2)$ are examples of atomic action terms. Expression $(b, b_1 \rightarrow b_3)$ means that the value of attribute b is changed from b_1 to b_3. Expression $(c, c_2 \rightarrow c_2)$ means that the value c_2 of attribute c remains unchanged. Expression r = $[[(a, a_2 \rightarrow a_2) \star (b, b_1 \rightarrow b_3)] \Rightarrow (d, d_1 \rightarrow d_2)]$ is an example of an action rule. The rule says that if value a_2 remains unchanged and value b will change from b_1 to b_3, then it is expected that the value d will change from d_1 to d_2. Clearly, $Dom(r) = \{a, b, d\}$.

Standard interpretation N_S of action terms in $S = (X, A, V)$ is defined as follow:

1. If $(a, a_1 \rightarrow a_2)$ is an atomic action term, then
   $N_S((a, a_1 \rightarrow a_2)) = [\{x \in X : a(x) = a_1\}, \{x \in X : a(x) = a_2\}]$.
2. If $t_1 = (a, a_1 \rightarrow a_2) \star t$ and $N_S(t) = [Y_1, Y_2]$, then
   $N_S(t_1) = [Y_1 \cap \{x \in X : a(x) = a_1\}, Y_2 \cap \{x \in X : a(x) = a_2\}]$.

Now, let us define $[Y_1, Y_2] \cap [Z_1, Z_2]$ as $[Y_1 \cap Z_1, Y_2 \cap Z_2]$ and assume that $N_S(t_1) = [Y_1, Y_2]$ and $N_S(t_2) = [Z_1, Z_2]$. Then, $N_S(t_1 \star t_2) = N_S(t_1) \cap N_S(t_2)$.

Let $r = [t_1 \rightarrow t_2]$ be an action rule, where $N_S(t_1) = [Y_1, Y_2]$, $N_S(t_2) = [Z_1, Z_2]$. Support and confidence of r are defined as follow:

1. $sup(r) = card(Y_1 \cap Z_1)$.
2. $conf(r) = [\frac{card(Y_1 \cap Z_1)}{card(Y_1)}] \cdot [\frac{card(Y_2 \cap Z_2)}{card(Y_2)}]$.

The definition of a confidence should be interpreted as an optimistic confidence. It requires that $card(Y_1) \neq 0$, $card(Y_2) \neq 0$, $card(Y_1 \cap Z_1) \neq 0$, and $card(Y_2 \cap Z_2) \neq 0$. Otherwise, the confidence of action rule is zero.

Coming back to the example of S given in Table 1, we can find many action rules associated with S. Let us take $r = [[(a, a_2 \rightarrow a_2) \star (b, b_1 \rightarrow b_2)] \Rightarrow (d, d_1 \rightarrow d_2)]$ as an example of the action rule. Then,

$N_S((a, a_2 \rightarrow a_2)) = [\{x_2, x_3, x_4, x_5\}, \{x_2, x_3, x_4, x_5\}]$,
$N_S((b, b_1 \rightarrow b_2)) = [\{x_1, x_2, x_4, x_6\}, \{x_3, x_7, x_8\}]$,
$N_S((d, d_1 \rightarrow d_2)) = [\{x_1, x_2, x_3, x_4, x_5, x_7\}, \{x_6\}]$,
$N_S((a, a_2 \rightarrow a_2) \star (b, b_1 \rightarrow b_2)) = [\{x_2, x_4\}, \{x_3\}]$.

Clearly, $sup(r) = 2$ and $conf(r) = 1 \cdot 0 = 0$.

Assume that $L([Y, Z]) = Y$ and $R([Y, Z]) = Z$. The new algorithm *ARD* for constructing action rules is similar to *ERID* [2] and *LERS* [4]. So, to present this algorithm, it is sufficient to outline the strategy for assigning marks to atomic action terms and show how terms of length greater than one are built. Only positive marks yield action rules. Action terms of length k are built from unmarked action terms of length $k-1$ and unmarked atomic action terms of length one. Marking strategy for terms of any length is the same as for action terms of length one.

Now, let us assume that $S = (X, A \cup \{d\}, V)$ is a decision system and λ_1, λ_1 denote minimum support and confidence, respectively. Each $a \in A$ uniquely defines the set $C_S(a) = \{N_S(t_a) : t_a$ *is an atomic action term built from elements in* $V_a\}$. By t_d we mean an atomic action term built from elements in V_d.

Marking strategy for atomic action terms

For each $N_S(t_a) \in C_S(a)$ **do**

if $L(N_S(t_a)) = \emptyset$ or $R(N_S(t_a)) = \emptyset$ or $L(N_S(t_a \star t_d)) = \emptyset$ or $R(N_S(t_a \star t_d)) = \emptyset$, **then** t_a is **marked negative**.

if $L(N_S(t_a)) = R(N_S(t_a))$ **then** t_a **stays unmarked**

if $card(L(N_S(t_a \star t_d)) < \lambda_1$ **then** t_a is **marked negative**

if $card(L(N_S(t_a \star t_d)) \geq \lambda_1$ and $conf(t_a \rightarrow t_d) < \lambda_2$ **then** t_a **stays unmarked**

if $card(L(N_S(t_a \star t_d)) \geq \lambda_1$ and $conf(t_a \rightarrow t_d) \geq \lambda_2$ **then** t_a is **marked positive** and the action rule $[t_a \rightarrow t_d]$ is printed.

Now, to clarify *ARD* (Action Rules Discovery) strategy for constructing action rules, we go back to our example with S defined by Table 1 and with $A_{St} = \{b\}$, $A_{Fl} = \{a, c, d\}$. We are interested in action rules which may reclassify objects from the decision class d_1 to d_2. Additionally, we assume that $\lambda_1 = 2$, $\lambda_2 = 1/4$.

All atomic action terms for S are listed below:
For Decision Attribute in S:

$N_S(t_{12}) = [\{x_1, x_2, x_3, x_4, x_5, x_7\}, \{x_6\}]$

For Classification Attributes in S:

$t_1 = (b, b_1 \rightarrow b_1)$, $t_2 = (b, b_2 \rightarrow b_2)$, $t_3 = (b, b_3 \rightarrow b_3)$, $t_4 = (a, a_1 \rightarrow a_2)$,
$t_5 = (a, a_1 \rightarrow a_1)$, $t_6 = (a, a_2 \rightarrow a_2)$, $t_7 = (a, a_2 \rightarrow a_1)$, $t_8 = (c, c_1 \rightarrow c_2)$,
$t_9 = (c, c_2 \rightarrow c_1)$, $t_{10} = (c, c_1 \rightarrow c_1)$, $t_{11} = (c, c_2 \rightarrow c_2)$, $t_{12} = (d, d_1 \rightarrow d_2)$.

Following the first loop of *ARD* algorithm we get:

$N_S(t_1) = [\{x_1, x_2, x_4, x_6\}, \{x_1, x_2, x_4, x_6\}]$ Not Marked /$Y_1 = Y_2$/

$N_S(t_2) = [\{x_3, x_7, x_8\}, \{x_3, x_7, x_8\}]$ Marked "-" /$card(Y_2 \cap Z_2) = 0$/

$N_S(t_3) = [\{x_5\}, \{x_5\}]$ Marked "-" /$card(Y_2 \cap Z_2) = 0$/

$N_S(t_4) = [\{x_1, x_6, x_7, x_8\}, \{x_2, x_3, x_4, x_5\}]$ Marked "-" /$card(Y_2 \cap Z_2) = 0$/

$N_S(t_5) = [\{x_1, x_6, x_7, x_8\}, \{x_1, x_6, x_7, x_8\}]$ Not Marked /$Y_1 = Y_2$/

$N_S(t_6) = [\{x_2, x_3, x_4, x_5\}, \{x_2, x_3, x_4, x_5\}]$ Marked "-" /$card(Y_2 \cap Z_2) = 0$/

$N_S(t_7) = [\{x_2, x_3, x_4, x_5\}, \{x_1, x_6, x_7, x_8\}]$ Marked "+"
/rule $r_1 = [t_7 \Rightarrow t_{12}]$ has $conf = 1/2 \geq \lambda_2$, $sup = 2 \geq \lambda_1$/

$N_S(t_8) = [\{x_1, x_4, x_8\}, \{x_2, x_3, x_5, x_6, x_7\}]$ Not Marked
/rule $r_1 = [t_8 \Rightarrow t_{12}]$ has $conf = [2/3] \cdot [1/5] < \lambda_2$, $sup = 2 \geq \lambda_1$/

$N_S(t_9) = [\{x_2, x_3, x_5, x_6, x_7\}, \{x_1, x_4, x_8\}]$ Marked "-" /$card(Y_2 \cap Z_2) = 0$/

$N_S(t_{10}) = [\{x_1, x_4, x_8\}, \{x_1, x_4, x_8\}]$ Marked "-" /$card(Y_2 \cap Z_2) = 0$/

$N_S(t_{11}) = [\{x_2, x_3, x_5, x_6, x_7\}, \{x_2, x_3, x_5, x_6, x_7\}]$ Not Marked /$Y_1 = Y_2$/

Now, we build action terms of length two from unmarked action terms of length one.

$N_S(t_1 \star t_5) = [\{x_1, x_6\}, \{x_1, x_6\}]$ Not Marked /$Y_1 = Y_2$/

$N_S(t_1 \star t_8) = [\{x_1, x_4\}, \{x_2, x_6\}]$ Marked "+"
/rule $r_1 = [[t_1 \star t_8] \Rightarrow t_{12}]$ has $conf = 1/2 \geq \lambda_2$, $sup = 2 \geq \lambda_1$/

$N_S(t_1 \star t_{11}) = [\{x_2, x_6\}, \{x_2, x_6\}]$ Not Marked /$Y_1 = Y_2$/

$N_S(t_5 \star t_8) = [\{x_1, x_8\}, \{x_6, x_7\}]$ Marked "-"
/rule $r_1 = [[t_5 \star t_8] \Rightarrow t_{12}]$ has $conf = 1/2 \geq \lambda_2$, $sup = 1 < \lambda_1$/

$N_S(t_5 \star t_{11}) = [\{x_6, x_7\}, \{x_6, x_7\}]$ Not Marked /$Y_1 = Y_2$/

$N_S(t_8 \star t_{11}) = [\emptyset, \{x_2, x_3, x_5, x_6, x_7\}]$ Marked "-" /$card(Y_1) = 0$/

Finally (there are only 3 classification attributes in S), we build action terms of length three from unmarked action terms of length one and length two.

Only, the term $t_1 \star t_5 \star t_8$ can be built. It is an extension of $t_5 \star t_8$ which is already marked as negative. So, the algorithm *ARD* stops and two action rules are constructed: $[[(b, b_1 \rightarrow b_1) \star (c, c_1 \rightarrow c_2)] \Rightarrow (d, d_1 \rightarrow d_2)]$, $[(a, a_2 \rightarrow a_1) \Rightarrow (d, d_1 \rightarrow d_2)]$. Following the notation used in previous papers on action rules mining (see [6], [14], [13], [12]), the first of the above two action rules will be presented as $[[(b, b_1) \star (c, c_1 \rightarrow c_2)] \Rightarrow (d, d_1 \rightarrow d_2)]$.

4 Application Domain and Experiment

Music Information Retrieval (MIR) is chosen as the application area for our research. In [11], authors present the system *MIRAI* for automatic indexing of music by instruments and emotions. When *MIRAI* receives a musical waveform, it divides that waveform into segments of equal size and then its classifiers identify the most dominating musical instruments and emotions associated with each segment and finally with the musical waveform. In [7], [8] authors follow another approach and present a Basic Score Classification Database (*BSCD*) which describes associations between different scales, regions, genres, and jumps. This database is used to automatically index a piece of music by emotions. In this section, we show how to use action rules extracted from *BSCD* assuming that we need to change the emotion either from the retrieved or submitted piece of music by minimally changing its score. By a score, in MIR area, we mean a written form of a musical composition.

To introduce the problem, let's start with Figure 1 showing an example of a score of a Pentatonic Minor Scale played in the key of C on a piano. As we can see, 8 notes are played: $A\sharp, G, A\sharp, F, D\sharp, G, C$ and C. The ordered sequence of the same notes without repetitions [$A\sharp$, C, $D\sharp$, F, G] uniquely represents that score. Now, we explain the process of computing its numeric representation $[2, 3, 2, 2]$. The score is played in the key of $A\sharp$ which becomes the root. Its second note C is 2 tones up from $A\sharp$. The third note $D\sharp$ is three tones up from C. The fourth note F is two tones up from $D\sharp$, and finally G is two tones up from F. This is how the sequence of jumps $[2, 3, 2, 2]$ with root $A\sharp$ is generated.

Essentially any combination of notes $A\sharp, C, D\sharp, F, G$ can be played while still remaining within the constraints of a C Pentatonic Minor Scale on a piano. This scale is illustrated in Figure 2. Accordingly one plays the root, plays 3 tones up, then 2 tones up then 2 tones up, and then 3 tones up (*m* means *mode*). The first note, or in musical terms, the "Root" is a C note. It means that the remaining four notes are all in the key of C Pentatonic Minor Scale on a piano. However, from the score itself, we have no idea about its key or scale. We can only discern the jumps between the notes and the repeated notes.

To tackle the above problem, authors in [7] built a Basic Score Classification Database (*BSCD*) which describes associations between different scales, regions, genres,

J_1	J_2	J_3	J_4	J_5	Scale	Region	Genre	Emotion	sma
2	2	3	2		Pentatonic Major	Western	Blues	melancholy	s
3	2	1	1	2	Blues Major	Western	Blues	depressive	s
3	2	2	3		Pentatonic Minor	Western	Jazz	melancholy	s
3	2	1	1	3	Blues Minor	Western	Blues	dramatic	s
3	1	3	1	3	Augmented	Western	Jazz	feel-good	s
2	2	2	2	2	Whole Tone	Western	Jazz	push-pull	s
1	2	4	1		Balinese	Balinese	ethnic	neutral	s
2	2	3	2		Chinese	Chinese	ethnic	neutral	s
2	3	2	3		Egyptian	Egyptian	ethnic	neutral	s
1	4	1	4		Iwato	Iwato	ethnic	neutral	s
1	4	2	1		Japanese	Japanese	Asian	neutral	s
2	1	4	1		Hirajoshi	Hirajoshi	ethnic	neutral	s
1	4	2	1		Kumoi	Japanese	Asian	neutral	s
2	2	3	2		Mongolian	Mongolian	ethnic	neutral	s
1	2	4	3		Pelog	Western	neutral	neutral	s
2	2	3	2		Pentatonic Majeur	Western	neutral	happy	m
2	3	2	3		Pentatonic 2	Western	neutral	neutral	m
3	2	3	2		Pentatonic 3	Western	neutral	neutral	m
2	3	2	2		Pentatonic 4	Western	neutral	neutral	m
2	2	3	3		Pentatonic Dominant	Western	neutral	neutral	m
3	2	2	3		Pentatonic Minor	Western	neutral	sonorous	m
1	3	3	2		Altered Pentatonic	Western	neutral	neutral	m
3	2	1	1	2	Blues	Western	Blues	depressive	m
4	3				Major	neutral	neutral	sonorous	a
3	4				Minor	neutral	neutral	sonorous	a
4	3	4			Major 7th Major	neutral	neutral	happy	a
4	3	3			Major 7th Minor	neutral	neutral	not happy	a
3	4	4			Minor 7th Major	neutral	neutral	happy	a
3	4	3			Minor 7th Minor	neutral	neutral	not happy	a
2	2	3	3		Major 9th	neutral	neutral	happy	a
2	1	4	3		Minor 9th	neutral	neutral	not happy	a
2	2	1	2	3	Major 11th	neutral	neutral	happy	a
2	1	2	2	3	Minor 11th	neutral	neutral	not happy	a
4	4				Augmented	neutral	neutral	happy	a
3	3	3			Diminished	neutral	neutral	not happy	a

Table 2. Basic Score Classification Database

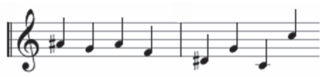

Fig. 1. Example score of a Pentatonic Minor Scale played in the key of C

Scale	sma	ii	iii	iv	v	vi	vii	viii	ix	x	xi	xii	#
Pentatonic Minor	m	3	2	2	3								4

Fig. 2. Representation of a Pentatonic Minor Scale

and jumps (see Table 2). The attribute J_i means i-th jump. When a music piece is submitted to QAS associated with $BSCD$, each note one by one, is drawn into the array of incoming signals. Assuming that the score is represented by Figure 1, QAS will generate five optional sequences:

[$A\sharp$, C, $D\sharp$, F, G], [G, $A\sharp$, C, $D\sharp$, F], [F, G, $A\sharp$, C, $D\sharp$], [$D\sharp$, F, G, $A\sharp$, C], or [C, $D\sharp$, F, G, $A\sharp$].

In the first case $A\sharp$ is the root, in the second G is the root, in the third F, in the fourth $D\sharp$, and in the fifth C is the root. Clearly, at this point, QAS has no idea which note is the root and the same which sequence out of the 5 is a representative one for the input sequence of notes $A\sharp, G, A\sharp, F, D\sharp, G, C$ and C. Table 3 gives numeric representation of these five sequences.

Root	J_1	J_2	J_3	J_4
A$^\sharp$	2	3	2	2
G	3	2	3	2
F	2	3	2	3
D$^\sharp$	2	2	3	2
C	3	2	2	3

Table 3. Possible Representative Jump Sequences for the Input Sequence

Paper [8] presents a heuristic strategy for identifying which sequence out of these five sequences is a representative one for the input score. The same, on the basis of associations between sequences of jumps and emotions which can be extracted from *BSCD*, we can identify the emotion which invokes in most of us the above input score.

What about changes to the input score so the scale associated with that score will change the way user wants. Action rules extracted from *BSCD* can be used for that purpose and they guarantee the smallest number of changes needed to achieve the goal. Example of an action rule extracted from *BSCD* is given below:

$[(J_1, 3 \rightarrow 2) \star (J_2, 2 \rightarrow 3)] \Rightarrow (Scale, PentatonicMinor \rightarrow Egyptian)$.

For instance, this rule can be applied to a music score represented by a sequence of 25 notes (Figure 3). They are
$[A\sharp, G, A\sharp, C, C, D\sharp, D, C, C, F, C, A\sharp, C, A\sharp, G, A, G, G, D\sharp, G, C, D\sharp, A\sharp, C, C]$.

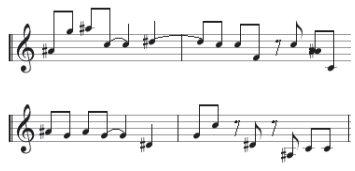

Fig. 3. Example of a Music Score

The ordered sequence of the same 25 notes without repetitions $[A\sharp, C, D, D\sharp, F, G]$ uniquely represents that score. Assume now, that the score is played in the key of G. So, $[3, 2, 2, 1, 2]$ is its numeric representation.

The classifier trained on Table 1, based on Levenshtein's distance [8], identified the sequence $[3, 2, 2, 3]$ as the closest one to $[3, 2, 2, 1, 2]$. Action rule $[(J_1, 3 \rightarrow 2) \star (J_2, 2 \rightarrow 3)] \Rightarrow (Scale, PentatonicMinor \rightarrow Egyptian)]$, extracted from Table 1, converts that score to
$[A\sharp, G, A, C, C, D\sharp, D, C, C, F, C, A, C, A\sharp, G, A, G, G, D\sharp, G, C, D\sharp, A, C, C]$.
Please notice that $A\sharp$ is changing to A only if the note C follows it in the input score.

This example shows how to use action rules to manipulate the music score. Following the same approach, we can manipulate music emotions, genre, and region.

5 Acknowledgment

This material is based in part upon work supported by the National Science Foundation under Grant Number *IIS-0414815*, the Ministry of Science and Higher Education in Poland under Grant *N N519 404734*, and Bialystok Technical University under Grant Number *S/WI/1/08*. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Also, we acknowledge the help from Rory Lewis in the application part of the paper.

6 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 was most likely less flexible than the health condition in one of the data set used in our experiment, and this may have to be considered. Future work shall address this issue as well as further analysis of the algorithm with more real world data sets.

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. S. Greco, B. Matarazzo, N. Pappalardo, R. Slowiński (2005), Measuring expected effects of interventions based on decision rules, J. Exp. Theor. Artif. Intell., Vol. 17, No. 1-2, 103-118
4. J. Grzymala-Busse (1997), A new version of the rule induction system LERS, Fundamenta Informaticae, Vol. 31, No. 1, 27-39
5. Z. He, X. Xu, S. Deng, R. Ma (2005), Mining action rules from scratch, Expert Systems with Applications, Vol. 29, No. 3, 691-699
6. S. Im, Z.W. Ras (2008), Action rule extraction from a decision table: ARED, in Foundations of Intelligent Systems, Proceedings of ISMIS'08, A. An et al. (Eds.), Toronto, Canada, LNAI, Vol. 4994, Springer, 160-168
7. R. Lewis, Z.W. Ras (2007), Rules for processing and manipulating scalar music theory, in Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE 2007), IEEE Computer Society, April 26-28, 2007, in Seoul, South Korea, 819-824
8. R. Lewis, W. Jiang, Z.W. Ras (2008), Mining scalar representations in a non-tagged music database, in Foundations of Intelligent Systems, Proceedings of ISMIS'08, A. An et al. (Eds.), Toronto, Canada, LNAI, Vol. 4994, Springer, 445-454
9. Z. Pawlak (1981) Information systems - theoretical foundations, Information Systems Journal, Vol. 6, 205-218
10. 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
11. Z.W. Ras, X. Zhang, R. Lewis (2007), MIRAI: Multi-hierarchical, FS-tree based music information retrieval system, (Invited Paper), in Proceedings of RSEISP 2007", LNAI, Vol. 4585, Springer, 80-89
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. 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
14. Z. Raś, E. Wyrzykowska, H. Wasyluk (2008), 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, 196-208
15. L.-S. Tsay, Z.W. Ras (2008), Discovering the concise set of actionable patterns, in Foundations of Intelligent Systems, Proceedings of ISMIS'08, LNAI, Vol. 4994, Springer, 169-178
16. 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