Monotone Systems approach in Inductive Learning

Peeter Roosmann, Leo Võhandu, Rein Kuusik, Tarvo Treier, and Grete Lind

***Abstract*—**In this paper we present a new approach for machine learning (ML) task solution based on Monotone Systems Theory, an inductive learning algorithm named by the authors as MONSIL (MONotone Systems in Inductive Learning). It has some advantages compared with several ML algorithms as rules overlapping, it can use several pruning techniques etc. The algorithm MONSIL usually produces more rules than other ML algorithms and it means that it would be more work-consuming than others, but as our experiments show, MONSIL works quite effectively.

In the paper we define also main terms of monotone systems theory, describe how to create monotone system to the data table and describe main advantages of the monotone systems approach. We also prove that concept description found by MONSIL is complete and consistent, explain algorithm's main steps on examples and discuss results of experiments comparing MONSIL effectiveness with well-known ID3.

***Keywords*—**Inductive learning, Machine learning, Machine learning algorithm, Monotone systems theory.

I. Introduction

In the domain of inductive learning there are many different algorithms in use (AQ, CN2, CART, ID3 and Assistant with their derivatives etc). Such a variety of algorithms shows that in the domain of inductive learning there are different problems which are hard to solve by one specific algorithm.

In the inductive learning environment we have problem how to manage large example sets with an unknown complicated structure. From our experience we can say that ***monotone systems*** algorithms have been very efficient in ordering large data tables and for finding regularities from these [1]-[4]. We try to use techniques that have been successful in data analysis in the inductive learning field.

This paper describes very shortly the underlying theory and describes also how to use monotone systems theory in inductive learning. A new learning algorithm is presented and discussed. In the end of paper we make a comparison of the effectiveness of our algorithm with ID3 on the well-known King-Rook-King task by D. Michie [5].

Manuscript received December 29, 2008. This work was supported in part by the Estonian Information Technology Foundation under Grant 08-03-00-36.

P. Roosmann was with the Department of Informatics, Tallinn University of Technology, Estonia. He is now with the Bank of Estonia (e-mail: peeter.roosmann@epbe.ee).

L. Võhandu is with the Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia (e-mail: leov@cc.ttu.ee).

R. Kuusik is with the Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia (e-mail: kuusik@cc.ttu.ee).

T. Treier is with the Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia (e-mail: tarvo.treier@gmail.com).

G. Lind is with the Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia (corresponding author to provide phone: +372 620 2306; fax: +372 620 2301; e-mail: grete@staff.ttu.ee).

II. Definitions

We mainly originate from the notions of the article [6].

The set of ***objects*** $X=\{x_1,\ldots,x_N\}$ can be described with ***attributes*** $t_1,\ldots, t_M$ so that every object x_i can be described as a tuple

$$x_i=(t_1(x_i),\ldots,t_M(x_i))=(x_{i1},\ldots,x_{iM}).$$

For each attribute t_j there exists a finite ***set of values*** A_j ($1\le j\le M$). So the ***attribute value*** x_{ij} of the object x_i belongs to the set A_j

$$x_{ij}=t_j(x_i)\in A_j.$$

Classes $C_1,\ldots,C_K$ are subsets of X such that

$$C_1\cup\ldots\cup C_K;\ \forall i\forall j,\ i\neq j : C_i\cap C_j=\varnothing.$$

The ***class value*** of the object $x\in X$ is c_j if $x_i\in Cj$. Let us denote the set of class values as

$$C=\{c_1,\ldots,c_K\}.$$

A ***learning example*** e_i is a tuple created from an object x_i and its class value

$$e_i=(x_i,c)=((t_1(x_i),\ldots,t_M(x_i)),c)=((x_{i1},\ldots x_{iM}),c).$$

Let us denote the ***set of examples*** E as

$$E=\{e_1,\ldots,e_n\}.$$

Let us denote the ***set of examples*** of class C_j as

$$E_j=\{e|e=(x,c_j),\ x\in C_j,\ C_j\subseteq X\}.$$

The ***positive example*** e_{j+} ***of the class*** C_j is an example which belongs to the set E_j, i.e.

$$e_{j+}\in E_j\subseteq E.$$

The ***negative example*** e_{j-} ***of the class*** C_j is an example which does not belong to the set E_j i.e.

$$e_{j-}\notin E_j\subseteq E.$$

The ***majority class*** of set E is the class with the largest number of examples.

Function d which does map according to every element of the set X a class value c_j is called a ***concept***

$$d:X\rightarrow C;\ d(x_i)=c_j \Leftrightarrow x_i\in C_j.$$

In inductive learning the learning system should on the base of the learning examples find a concept description D which

maps a class value for any object of the set X (not only for objects from the example set) $D:X\rightarrow C$. Consequently the inductive learning system should (in ideal case) find on the base of learning examples such concept description D which maps to every object the same class value as a concept d

$$\forall x\in X,\ D(x)=d(x).$$

The ***(concept) description*** is the set of classification rules $D=\{r_1,\ldots,r_S\}$.

A ***classification rule*** (***decision rule***) is an implication where a condition part is a complex and a conclusion part is a class name:

$$r_j = \text{“}Com_j => c_k\text{”}$$

or

$$r_j = \text{“if } Com_j \text{ then } c_k\text{”}$$

or

$$r_j=(Com_j,c_k).$$

Complex Com_j is a tuple of ***selectors*** Sel_{jk} $(k=1,\ldots,M)$

$$Com_j=(Sel_{j1},\ldots,Sel_{jM}).$$

Selector Sel_j is a subset of the set of values of the attribute t_j

$$Sel_j\subseteq A_j.$$

Description D ***maps a class value*** c_k ***for an object*** x_i if it contains a classification rule r_j which maps a class value c_k for the object x_i

$$\exists r_j\in D,\ r_j(x_i)=c_k \Rightarrow D(x_i)=c_k.$$

Rule $r_j=(Com_j,c_k)$ ***maps a class value*** c_k ***for an object*** x_i if its complex Com_j covers the object x_i

$$r_j=(Com_j,c_k),\ cover(Com_j,x_i) \Rightarrow r_j(x_i)=c_k.$$

Complex Com_j ***covers an object*** x_i if all its selectors Sel_{jk} cover this object

$$\forall k,\ 1\le k\le M,\ cover(Sel_{jk},x_i) \Rightarrow cover(Com_j,x_i).$$

Selector Sel_{jk} ***covers an object*** x_i if the value of the attribute t_k of the object x_i is in the set Sel_{jk}

$$\forall j,\ 1\le k\le M,\ x_{ik}\in Sel_{jk} \Rightarrow cover(Sel_{jk},x_i).$$

Description D is ***consistent*** on the set $X'\subseteq X$ if all its rules map the same class value for any object $x\in X'$

$$\forall r_i,r_j\in D,\ x\in X',\ X'\subseteq X,\ cover(Com_i,x),\ cover(Com_j,x) \Rightarrow r_i(x)=r_j(x).$$

Description D is ***complete*** on the set $X'\subseteq X$ if for each object $x\in X'$ there exists at least one rule so that its complex covers this object

$$\forall x\in X',\ X'\subseteq X,\ \exists r_j\in D,\ cover(Com_j,x\}$$

The inductive learning algorithms have to allow us to find descriptions that are at the same time both consistent and complete.

III. Monotone Systems Based Inductive Learning Algorithms

A lot of initial data and complicated inner structures of objects are common difficulties for data analysis, data mining, graph theory etc. This is true also for learning from examples. In the well-ordering of large and confused data sets the use of the algorithms of theory of monotone systems has been very successful and we try to use these ideas in the inductive learning field as well.

In [7] L. Võhandu described a possibility for quick data processing methodology, which uses the idea of building weakly monotone systems on data tables, using empirical frequencies of discrete attributes' values. The main mechanism of using algorithms of the theory of monotone systems is so called frequency transformation. It means that frequencies of every attribute value are calculated and values in the data table (corresponding to the objects) are replaced with their corresponding frequencies over the data table. Such a transformation enables one to estimate the degree of systematic conformity of objects on the set of learning examples. Both, objects with high conformity (for finding general rules) and objects with low conformity (for discovering exceptions and noise) are substantial in inductive learning.

Some of the attributes are more important in learning from examples. The estimation of importance of attributes is easy to do using the frequency transformation.

The frequency transformation and conformity measure guarantees a simple way to estimate the typicality of objects. The theory of monotone systems allows one to work with the tables of frequencies instead of real data tables. The computing with much smaller tables of frequencies is a lot quicker than with full data tables, although all the important information for learning is easily accessible.

A. Main concepts of theory of monotone system

We present the main concepts of the theory of monotone systems by J. Mullat [8].

Let a finite set $X=\{x_1,\ldots, x_N\}$ and a function π_X on it which maps to each element $x\in X$ a certain nonnegative number (weight) $\pi_X(x)$, be given. This function π_X is called a ***weight function*** if it is defined on any subset $X'\subset X$; the number $\pi_{X'}(x)$ is called a ***weight of element*** x on X'.

A set X with a weight function π_X is called a ***system*** and is denoted by $S=(X,\pi_X)$.

The system $S'=(X',\pi_{X'})$ where $X'\subset X$ is called a ***subsystem*** of the system $S=(X,\pi_X)$.

The system $S=(X,\pi_X)$ is called ***monotone*** (more exactly weakly monotone) if in the case of any x we have

$$\forall X'\subseteq X;\ \forall x\in X\backslash\{y\},\ (y\in X):\ \pi_{X'\backslash\{y\}}(x)\le\pi_{X'}(x).$$

Function Q that is mapping to every subset $X'\subset X$ of a monotone system S a nonnegative number $Q(X')=\min \pi_{X'}(x)$ is called an ***objective function***.

The subsystem $S'=(W,\ \pi_W)$ of the monotone system $S=(X,\pi_X)$ with the property that the objective function obtains a maximal value $Q(W)=\max Q(X')=\max\min \pi_{X'}(x)$ on it is called a ***kernel*** of the monotone system S: respectively the value Q(W) is called a measure of the kernel quality. For a kernel the following relation is valid $\forall X'\subseteq X$: $Q(X')\le Q(W)$.

To use the method of monotone systems in practice we have to fulfill two conditions:

1) There has to exist a weight function $\pi_X(x)$ which will give a measure of influence for every element $x\in X$;
2) There have to be rules to recompute the weights of the elements of the system $S=(X,\pi_X)$ in case there is a change

in the weight of one element $x \in X$.

These two conditions give a lot of freedom to choose the weight functions and create formal rules of weight changes in the system $S=(X,\pi_X)$. The only constraint we have to keep in mind is that the rules and the weight function have to be compatible in the sense that after eliminating all elements x from the system S the final weights of $x \in X$ must be equal to zero.

Kernels of a monotone system built on the empirical system create a nice picture of the inner structure of the given empirical system. We just mention that in general case one orders all objects of the system $S=(X,\pi_X)$ by eliminating in N steps all objects using the weight minimality condition. On every step we eliminate the object with minimal weight value and recalculate weight of all other remaining objects.

B. Algorithm MONSA

To find all kernels of a monotone system a very effective polynomial time algorithm named MONSA (**MON**otone **S**ystems **A**lgorithm) was created [2], see also [3], [9].

In order to describe that algorithm we use an additional concept of an intersection.

The ***intersection*** H of the set X is any set of attribute values which belong simultaneously to all objects $x_i \in X$

$$H=\{x_{ij} | \forall k \forall i,\ k \neq i,\ x_{ij}=x_{kj}\}.$$

The intersection H describes on the set X a subset X_H

$$X_H=\{x_i | \forall j,\ x_{ij} \in H\},\ X_H \subseteq X,$$

In [2] it is proved that the subset $X_H \subseteq X$ extracted by the algorithm MONSA is a kernel $S'=(X_H,\ \pi_{X_H})$ of the system $S(X,\pi_X)$.

In order to use the monotone system method for inductive learning we use a very simple weight function connected to the frequencies of system element attribute values. To build a monotone system on the set of discrete data X we first calculate the frequency table F: $F=\|f_{jk}\|$, $j=1,\ldots,M$; $k=1,\ldots,|A_j|$, where $f_{jk}=|Z_{jk}|$, $Z_{jk}=\{x_i | x_{ij}=k\}$.

After that we define the weight of the element x_{ij} on the set X' in the system $S=(X',\pi_{X'})$ as $\pi_{X'}(x_{ij})=f_{jk}$.

The system with a weight function $\pi_{X'}(x_{ij})=f_{jk}$ is weakly monotone since

$$\forall i, \forall j, \forall k: |Z_{jk} \setminus \{x_i\}| \leq |Z_{jk}|$$

i.e. in general, the elimination of an object can only lessen the frequencies and weights of other objects in the system $S(X,\pi_X)$.

It is easy to see that the concept of intersection is similar to the concept of complex, so it would be useful for learning from examples.

The algorithm for learning from examples built up on the algorithm MONSA follows:

```
Initialize description D={}
FOR j=1 TO K (K - number of classes)
   Let complex Com=(A1,…,AM) i.e. ∀Selj=Aj
   Find Ej={ei|ei∈Cj}
   Call procedure MONSA(Com,Cj,Ej)
NEXT j

PROCEDURE MONSA(Com, Class, X)
   Find the frequency table F on the set X for
       all attributes tj where Selj=Aj
   Find the minimal value fmin=fj'k' from the
       frequency table F
   Find the maximal value fmax=fj''k'' from the
       frequency table F
   IF fmin=fmax THEN
      Add the rule r=(Com,Class) into the
          description D
   ELSE
      WHILE fmax>fmin>0 DO
         Let Selj''=k''
         Call procedure MONSA(Com,Class,X)
         Let fj''k''=0
      END WHILE
   END IF
END PROCEDURE
```

Description produced via such a simple algorithm is complete, i.e. every example is covered by at least one classification rule. All intersections are created only by the set of examples E_j of the class C_j. So, it is possible, that there exists another set E_i ($i \neq j$), such that it has an identical intersection with the one extracted from E_j. Consequently, we cannot declare that the derived description is consistent. There can exist an example which belongs to both classes C_j and C_i.

In the next section we describe a more powerful algorithm, which generates complete and consistent concept descriptions.

C. New inductive learning algorithm MONSIL

In the algorithm presented in the section III.B the learning examples are arranged into classes and all intersections (complexes) are derived for every subset. Algorithm MONSIL (MONotone Systems in Inductive Learning) presented in this section works in a reverse direction, i.e. an intersection (complex) is derived first using frequency tables and after that one checks whether all examples covered by it belong to the same class. If it is true, then the classification rule is added into the concept description.

Algorithm MONSA presents a very comfortable mechanism for extracting kernels of the data table. Using the frequency table intersections are made. If the maximum value of the weight function is equal to the minimum value of the weight function, then a kernel can be extracted [2].

In the algorithm MONSIL we first build the frequency tables in the same way, but the conditions for extracting rules are different. If a complex Com which covers only examples of one class C_j is found, then a classification rule $r=(Com,c_j)$ is added into the concept description D.

Unlike in the algorithm MONSA the selectors (elements of intersections) in MONSIL are chosen according to the minimal values of the elements of frequency tables. There is no contradiction with the theory of monotone systems, because the condition of extracting a kernel remains the same - the maximal value of the weight function has to be equal to the

minimum value of the weight function. Only the order of extracting a kernel will be different.

In the following algorithm we use:
Com for complex;
Sel for selector.

The algorithm MONSIL:

```
Initialize the complex Com={& Sel_k }
Initialize the description D={}
Find the frequency table F on set of learning
    examples E
Call procedure MONSIL(Com,F)
```

The main procedure of the algorithm follows:

```
PROCEDURE MONSIL(Com,F)
   IF ∃j,∀i, cover(Com,x_i): x_i ∈C_j THEN (i.e.
       if all examples covered by a complex
       belong to the same class)
      Add a rule r=(Com,c_j) to the description
         D
   ELSE
      Find the frequency table F'=||f'_ij|| on
          the set of learning examples E
      If f_ij=0 then let f'_ij=0
      Find the minimal value f'_min=f'_j'k' (over
          all f'_jk>0)
      Find a new complex Com' so that Sel_j'=k'
          and the other selectors Sel_j (j≠j')
          are the same as in the complex Com
      WHILE f'_min>0 DO
         Call procedure MONSIL(Com',F')
         Let f'_j'k'=0
         Find the new minimal value f'_min from
             the frequency table F' and a new
             complex Com'
      END WHILE
   END IF
END PROCEDURE
```

Theorem 1. Concept description found by the algorithm MONSIL is complete - every example is covered by at least one rule.

Proof: Minimal values from the frequency table are searched by the algorithm as long as there are frequencies greater than zero. The value of the element of the frequency table is put to zero iff there exists a rule that corresponds to this subset of examples.

Theorem 2: Concept description found by the algorithm MONSIL is consistent - all its rules map the true class value for any example.

Proof: A rule will be added into the description only if examples covered by its complex belong to the same class. This condition of algorithm excludes all rules, that cover also examples belonging to some other class(es).

D. The first example

In order to demonstrate MONSIL in action the classic J. R. Quinlan's example set is used [10]. There are five examples of class "-" and three examples of class "+". All examples have three attributes (see Table I):

A_1="Height" with values "short", "tall";
A_2="Hair" with values "dark", "red", "blond";
A_3="Eyes" with values "blue", "brown".

Table I. The example set (from Quinlan)

	Height	Hair	Eyes	Class
x_1	short	blond	blue	+
x_2	tall	blond	brown	-
x_3	tall	red	blue	+
x_4	short	dark	blue	-
x_5	tall	dark	blue	-
x_6	tall	blond	blue	+
x_7	tall	dark	brown	-
x_8	short	blond	brown	-

The concept description generated by ID3 (Quinlan [10]) contains four classification rules:

r_1=((Hair="dark")=>(Class="-"));
r_2=((Hair="red")=>(Class="+"));
r_3=((Hair="blond"&Eyes="blue")=>(Class="+"));
r_4=((Hair="blond"&Eyes="brown")=>(Class="-")).

We describe now how MONSIL works on the example set. We collect all attribute values into one table (Table II) and count the corresponding frequencies of those values (Table III).

Table II. The attribute values in the frequency table

	A_1	A_2	A_3	C
v_1	short	dark	blue	-
v_2	tall	red	brown	+
v_3	*	blond	*	*

Table III. The frequency table F_1

	A_1	A_2	A_3	C
v_1	3	3	5	5
v_2	5	1	3	3
v_3	*	4	*	*

Step 1: We find the minimal value $f_{min}=f_{22}=1$ from the frequency table F_1. Since there is only one example (x_3) covered by the complex (Hair="red") then the rule r_1=((Hair="red")=>(Class="+")) will be added into the description. The value f_{22} in the frequency table F_1 will be put to zero.

Step 2: The new minimal value $f_{min}=f_{11}=3$ will be found. If there are many equal minimal values we take the first one. For the three examples x_1, x_4, x_8 covered by the complex (Height="short") the new frequency table is F_2 (see Table IV) and from here for $f_{min}=f_{21}=1$ a new rule r_2=((Height="short"& Hair="dark") => (Class="-")) will be created. In the frequency

table F_2 the value f_{21} will be put to zero.

Table IV. The frequency table F_2 (Height="short")

	A_2	A_3	C
v_1	1	2	2
v_2	0	1	1
v_3	2	*	*

Step 3: A new $f_{min}=f_{32}=1$ will be found. Consequently: r_3=((Height="short"&Eyes="brown")=>(Class="-")). In the frequency table F_2 the value f_{32} will be put to zero.

Step 4: The new $f_{min}=f_{23}=2$ will be found. For the examples x_1, x_8 covered by the complex (Height="short"& Hair="blond") a new frequency table F_3 (see Table V) will be created. Here "1\0" means that in the frequency table F_2 the value f_{32} is already zero, consequently the corresponding complex (Height="short"&Hair="blond"&Eyes="brown") is unnecessary – it would be a specification of already found rule r_3. The minimal value $f_{min}=f_{31}=1$ of the table F_3 will create the rule r_4=((Height="short"&Hair="blond"&Eyes="blue")=> (Class="+")). In the frequency table F_2 the value f_{23} will be put to zero.

Table V. The frequency table F_3 (Height="short"&Hair="blond")

	A_3	C
v_1	1	1
v_2	1\0	1
v_3	*	*

Step 5: For the new $f_{min}=f_{31}=2$ (examples x_1, x_4) in F_2 the frequency table F_4 (see Table VI) will be created. "Bringing zeros down" from F_2 prevents finding complexes (Height="short"&Eyes="blue"&Hair="dark") which is a specification of already found rule r_2 and (Height="short"& Eyes="blue"&Hair="blond") which is a repetition of r_4. In F_4 there is no $f_{min}\neq 0$ therefore in the frequency table F_2 the value f_{31} will be put to zero and there is no new $f_{min}\neq 0$ neither. Now we have exhausted all complexes created from the field f_{11} and the value f_{11} in the frequency table F_1 will be put to zero.

Table VI. The frequency table F_4 (Height="short"&Eyes="blue")

	A_2	C
v_1	1\0	1
v_2	0	1
v_3	1\0	*

Step 6: The new minimal value in F_1 is $f_{min}=f_{21}=3$, the new complex is (Hair="dark"), the new examples are x_4, x_5, x_7 and the corresponding frequency table is F_5 (see Table VII). Here $f_{41}=f_{min}=3$ (f_{41} means the number of examples in the class "-"), consequently r_5=((Hair="dark")=>(Class="-")). The value f_{21} in the frequency table F_1 will be put to zero.

Table VII. The frequency table F_5 (Hair="dark")

	A_1	A_3	C
v_1	1\0	2	3
v_2	2	1	0
v_3	*	*	*

Step 7: We will choose a new minimal nonzero in the table F_1 $f_{min}=f_{32}=3$. The new complex is (Eyes="brown"), the new examples are x_2, x_7, x_8 and the corresponding frequency table is F_6 (see Table VIII). Here $f_{41}=f_{min}=3$, consequently r_6=((Eyes="brown")=>(Class="-")). The value f_{32} in the frequency table F_1 will be put to zero.

Table VIII. The frequency table F_6 (Eyes="brown")

	A_1	A_2	C
v_1	1\0	1\0	3
v_2	2	0	0
v_3	*	2	*

Step 8: The new minimal value in F_1 is $f_{min}=f_{23}=4$, the new complex is (Hair="blond") and the new examples for learning are x_1, x_2, x_6, x_8, the corresponding frequency table is F_7 (see Table IX).

Table IX. The frequency table F_7 (Hair=" blond")

	A_1	A_3	C
v_1	2\0	2	2
v_2	2	2\0	2
v_3	*	*	*

Step 9: The minimal value in F_7 is $f_{min}=f_{12}=2$, the corresponding complex is (Hair="blond" & Height="tall"), the frequency table is F_8 (see Table X) and the new $f_{min}=f_{31}=1$ in F_8. Consequently: r_7 = ((Hair="blond" & Height="tall" & Eyes="blue")=>(Class="+")). The value f_{31} in the frequency table F_8 and after that the value f_{12} in the frequency table F_7 will be put to zero.

Table X. The frequency table F_8 (Hair=" blond"&Height="tall")

	A_3	C
v_1	1	1
v_2	1\0	1
v_3	*	*

Step 10: The new minimal value in F_7 is $f_{min}=f_{31}=2$ and the new frequency table is F_9 (see Table XI). Here $f_{42}=f_{min}=2$ and consequently r_8 = ((Hair="blond" & Eyes="blue")=> (Class="+")). The value f_{31} in the frequency table F_7 will be put to zero. In F_7 there is no $f_{min}\neq 0$, so f_{23} in F_1 will be put to zero.

Table XI. The frequency table F_9 (Hair=" blond"&Eyes="blue")

	A_1	C
v_1	1\0	0
v_2	1\0	2
v_3	*	*

Step 11: The new minimal value in the frequency table F_1 is $f_{min}=f_{12}=5$, the corresponding complex is (Height="tall"), the examples are x_2, x_3, x_5, x_6, x_7 and the new frequency table is F_{10} (see Table XII).

Table XII. The frequency table F_{10} (Height="tall")

	A_2	A_3	C
v_1	2\0	3	3
v_2	1\0	2\0	2
v_3	2\0	*	*

Step 12: The minimal value of F_{10} is $f_{min}=f_{31}=3$. The new complex (Height="tall"& Eyes="blue") and the corresponding frequency table F_{11} (see Table XIII) give no new rule.

Table XIII. The frequency table F_{11} (Height="tall"&Eyes="blue")

	A_2	C
v_1	1\0	1
v_2	1\0	2
v_3	1\0	*

In the table F_{10} no new $f_{min}\neq 0$ can be found, therefore in F_1 the value f_{12} will be put to zero.

Step 13: The complex (Eyes="blue") will be generated on the base of $f_{min}=f_{31}=5$ in F_1. The corresponding frequency table is F_{12} (see Table XIV). The examples belong to both of the classes and the new minimal value can not be found. Consequently the frequency f_{31} in F_1 can be put to zero.

Table XIV. The frequency table F_{12} (Eyes="blue")

	A_1	A_2	C
v_1	2\0	2\0	2
v_2	3\0	1\0	3
v_3	*	2\0	*

Now the concept description is ready because in the frequency table F_1 there is no minimal value different from zero.

So altogether eight rules were generated by the algorithm MONSIL:

r_1=((Hair="red")=>(Class="+"));

r_2=((Height="short"&Hair="dark")=>Class="-");

r_3=((Height="short"&Eyes="brown")=>(Class="-"));

r_4=((Height="short"&Hair="blond"&Eyes="blue")=>(Class="+"));

r_5=((Hair="dark")=>(Class="-"));

r_6=((Eyes="brown")=>(Class="-"));

r_7=((Height="tall"&Hair="blond"&Eyes="blue")=>(Class="+"));

r_8=((Hair="blond"&Eyes="blue")=>(Class="+")).

Now we apply the second procedure for lessening a number of rules. If there are two rules r_i and r_j in the description so that: $r_i\neq r_j$: $Com_i\subseteq Com_j$ then the rule r_j can be excluded from the description. If any example is covered by such a rule r_j then it is also covered by the rule r_i.

So for our example the rules r_2, r_3, r_4 and r_7 can be excluded from our description:

rule r_1 covers x_3 of Class "+";

rule r_5 covers x_4, x_5 and x_7 of Class "-";

rule r_6 covers x_2, x_7 and x_8 of Class "-";

rule r_8 covers x_1 and x_6 of Class "+".

The resulting description is complete and consistent.

It is easy to check that rules r_1, r_5, r_8 correspond to the ID3 rules, but the rule r_6=((Eyes="brown")=>(Class="-")) is simpler than ID3 rule ((Hair="blond"&Eyes="brown")=>(Class="-")). It covers x_7 which is already covered by rule r_5 (overlapping). ID3 covers objects only once.

E. The second example

If we use initial data table with different order of attributes (columns): A_2 (Hair) – A_3 (Eyes) – A_1 (Height), then the search tree traversed by the algorithm is different (stricken-through text shows the places where zeros "brought down" from upper frequency table prevent entering corresponding (redundant) branches of the search tree):

Hair="red" (r1)
Hair="dark" (r2)
Eyes="brown" (r3)
Height="short"
~~&Hair="dark"~~
~~&Eyes="brown"~~
&Hair="blond"
~~&Eyes="brown"~~
&Eyes="blue" (r4)
&Eyes="blue"
~~&Hair="dark"~~
~~&Hair="blond"~~
Hair="blond"
~~&Eyes="brown"~~
~~&Height="short"~~
&Eyes="blue" (r5)
&Height="tall"
~~&Eyes="blue"~~
~~&Eyes="brown"~~
Eyes="blue"
~~&Hair="dark"~~
~~&Hair="red"~~
~~&Hair="blond"~~
~~&Height="short"~~
&Height="tall"
~~&Hair="dark"~~

~~&Hair="red"~~
~~&Hair="blond"~~
Height="tall"
~~&Hair="dark"~~
~~&Hair="red"~~
~~&Hair="blond"~~
~~&Eyes="blue"~~
~~&Eyes="brown"~~

As a result we get five rules:
r_1=((Hair="red")=>(Class="+"));
r_2=((Hair="dark")=>(Class="-"));
r_3=((Eyes="brown")=>(Class="-"));
r_4=((Height="short"&Hair="blond"&Eyes="blue")=>(Class="+"));
r_5=((Hair="blond"&Eyes="blue")=>(Class="+")).

One of them is redundant: r_4 is a specification of r_5. The example showed that the result of the algorithm MONSIL depends on used order of attributes' values. But after using the second procedure for lessening the number of rules we always get the same result. For our example it is four rules as in case of previous example:
rule r_1 covers x_3 of Class "+";
rule r_2 covers x_4, x_5 and x_7 of Class "-";
rule r_3 covers x_2, x_7 and x_8 of Class "-";
rule r_5 covers x_1 and x_6 of Class "+".

IV. Effectiveness of MONSIL

D. Michie has described in [5] an experiment in the Turing Institute comparing different algorithms of inductive learning. We use a similar King-Rook-King task to compare the effectiveness of the algorithms MONSIL and ID3.

D. Michie took a set of random positions with three pieces on a chess-board - white king, white rook and black king as a set of objects. The position of each piece on the board is described by two attributes. In some positions white's move is allowed by chess laws, in some positions not. So, two classes can be distinguished. The possible number of different positions is 64^3=262144.

In our experiment three similar training suites consist of five sets of positions which were randomly generated as training data examples and two test data suites consist of two sets of examples which were randomly generated for testing. All sets of positions are generated independently that means bigger sets may not consist of smaller sets and the same size sets in different suites are not the same.

We have split our test scenario into two stages. In both stages we have several training data example sets and one testing set. All rules found by both learning algorithms for every training data example set are tested separately. The speed of generating rules, the number of rules and the exactness of predicting were estimated.

In the first stage we used two training data sets from each (three) training suite, first one containing 1000 examples and second one containing 2000 examples. All (2x3) found rule sets from each training suite were tested with one set of 200 positions.

In the second stage we used the same two training data sets from each training suite, which were used already in the first stage. In addition, we used three training-data sets from each training suite, where accordingly 3000, 4000 and 8000 examples were present. The size of the testing set was 400 positions.

In our experiment the test scenario was executed two times. Only the test data suite was replaced on the second time. In both cases the same training suites were used. In the first execution, test data suite 1 was used and in the second run we used test data suite 2.

The results of the experiments with both MONSIL and ID3 base algorithm are presented in the five following tables and illustrated with four figures. Table XV holds the average speed of generating rules, Tables XVI, XVII and XVIII hold the average exactness of predictions and Table XIX holds the average number of found rules. Figure 1 shows trends of growth of both algorithms´ average execution times while the learning examples were growing. Figure 2 shows the ratio between MONSIL and ID3 algorithms´ average execution times. Figure 3 shows trends of growth of both algorithms´ average number of rules while the learning examples were growing. Figure 4 shows the ratio between MONSIL and ID3 algorithms´ average number of rules. (Both algorithms are implemented in Java. The computer used in the experiment was Pentium M 2,0GHz.)

We notice that the MONSIL algorithm is more exact in classification and not very much slower than the ID3 algorithm regardless of fact that it extracts much more rules.

Table XV. Algorithms' average execution times (s)

Learning examples	1000	2000	3000	4000	8000
ID3	0,06	0,12	0,17	0,22	0,44
MONSIL	0,16	0,28	0,34	0,54	0,69
Ratio	2,7	2,3	2,0	2,5	1,6

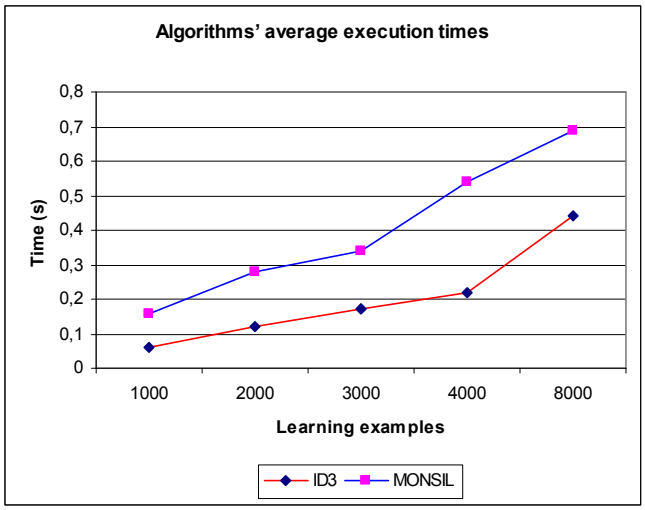

Fig. 1. Algorithm's average execution times

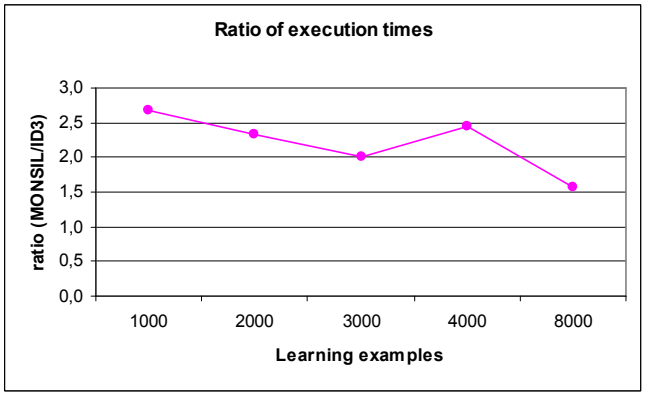

Fig. 2. Ratio of execution times

Table XVI. Algorithms' average exactness of predictions (%) on test suite 1

		Learning examples	ID3 %	MONSIL %
Test suite 1	First stage	1000	58,67	64,00
		2000	59,00	64,00
	Second stage	1000	53,75	62,00
		2000	58,58	64,75
		3000	55,75	65,75
		4000	58,83	67,75
		8000	58,08	69,75
		Average	**57,52**	**65,43**

Table XVII. Algorithms' average exactness of predictions (%) on test suite 2

		Learning examples	ID3 %	MONSIL %
Test suite 2	First	1000	55,67	62,00
		2000	62,33	67,00
	Second	1000	56,58	63,00
		2000	57,17	65,75
		3000	55,42	66,75
		4000	58,83	69,50
		8000	59,17	71,25
		Average	**57,88**	**66,46**

Table XVIII. Average exactness (%)

Test set	ID3	MONSIL
Average of test suite 1	57,52	65,43
Average of test suite 2	57,88	66,46
Total average	**57,70**	**65,95**

Table XIX. Average number of rules

Number of examples	1000	2000	3000	4000	8000
ID3	579	1126	1685	2160	4334
MONSIL*	8966	15709	22284	28488	43600
Ratio	15,5	14,0	13,2	13,2	10,1

* In the Table XIX procedure for lessening a number of rules was not used.

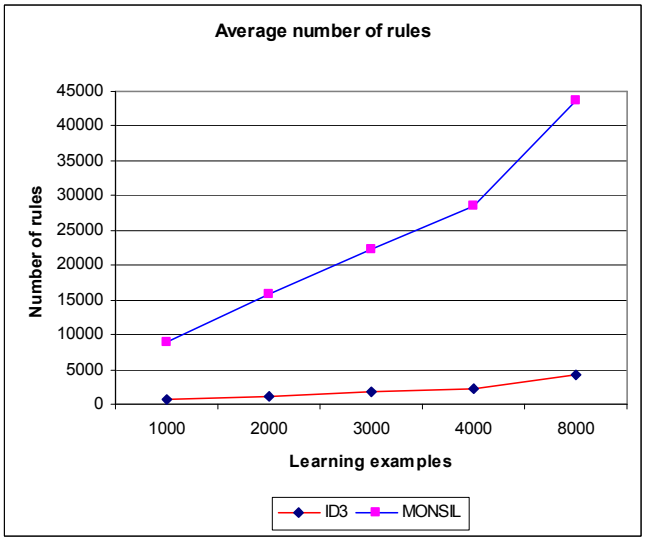

Fig. 3. Average number of rules

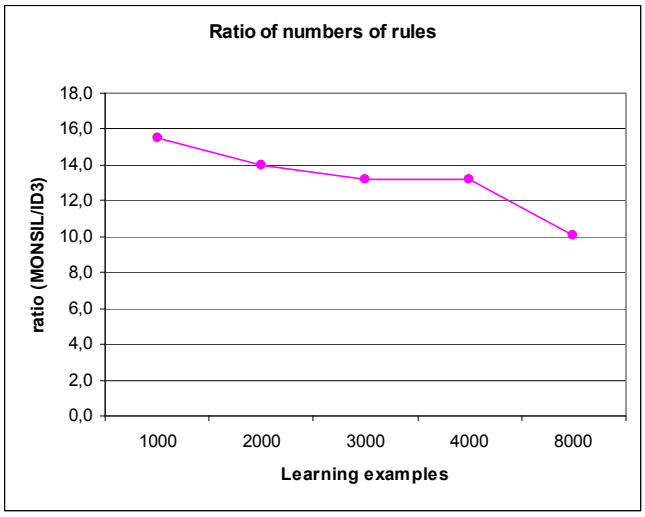

Fig. 4. Ratio of number of rules

V. Advantages and Disadvantages of MONSIL

We saw in chapter III.D that it can be happen that two or more rules cover the same object. Such a situation is called an ***overlapping***.

"Classical" algorithms do not take this situation into account. If a rule is added to the description, then corresponding examples will not be taken into consideration during generating next rules.

The use of frequency table technique allows to overcome this disadvantage. So, in case of overlapping the rule will be generated on the basis of larger set of examples than "classical" algorithms do. In the example of the section III.D the learning example x_7 is covered by complexes of the rules r_6 and r_5.

It is easy to discover the contradictory examples in MONSIL i.e. the situation where the objects with identical descriptions (complexes) belong to different classes. It is done as follows: if we have reached to such node of a search tree, where it is not possible to make the next extract (all the attributes are used already, but objects still belong to different classes), then we can say that these examples are contradictory.

In the case of large sets of examples algorithm MONSIL is quite quick (as we could see in the previous section). Most of its speed comes thanks to the circumstance that a frequency table allows to estimate the conformity of attributes in a very simple way. Using the frequency table method allows to reduce necessary memory space dramatically in real applications as well.

The other difficult problem - noise - can be taken into account by different pruning techniques. Those techniques allow to choose different parameters and values of them. In our examples and experiments presented in the paper we did not use any pruning techniques.

MONSIL extracts a lot of classification rules. The exclusion of excessive rules from the description takes some additional time proportional to the square of the rules number.

VI. Possible Modifications of MONSIL using Pre-pruning Techniques

The completeness and consistency of a description are very important in the algorithm MONSIL, but it is much more important in the practical systems of learning from examples that as many as possible objects will be classified exactly. Usually, learning examples presented to the system form only a small part from whole possible example set and there is often different level of noise in them. A consistent and complete description found for the set of examples does not have to be the best one. Giving up those strict requirements we can often find rules that guarantee exact classification on many more objects from the same expert field.

The main technique to facilitate work with noisy and large sets of examples is pruning. It can reduce the size of the decision tree or of the set of examples, making at the same time the prediction more exact. If by generating the concept description the requirements of consistency and completeness are given up, then we are dealing with pre-pruning. If changes affecting completeness and consistency are made in an already generated description, then such a technique is called post-pruning. Pre-pruning techniques are unique for different algorithms while post-pruning techniques are similar.

In the following we present some pre-pruning techniques one can use in our algorithm MONSIL. There are six different techniques of pre-pruning based on following indicators:

1) Class frequency,
2) Rule weight,
3) Frequency threshold,
4) Deviation,
5) Rule length,
6) Common rule.

Class frequency. Class frequency P_f the indicates percentage of examples of the current majority class in the subset of examples

$$P_f = \frac{N_m}{N} * 100\%$$

where $N_m=\max |E_j|$ - number of examples in the majority class.

Class frequency threshold P'_f sets a condition, that a classification rule is only then added to the description if $P_f>P'_f$. Using the threshold of class frequencies reduces the risk of over-specification created by the noise. It is also possible that the size of description and time spent to create it on the computer diminishes, because the rules are more general and not so many fractured frequency tables have to be calculated. If pruning is not necessary, then $P'_f=100\%$.

Rule weight. Rule weight P_w indicates a percentage of examples covered by complex Com_j in whole set of examples

$$P_W = \frac{N_{Com_j}}{N} * 100\%$$

If the weight P_w for a complex is less than a threshold P'_W, then all corresponding elements of the frequency table will be

put to zero. It means also that this rule will not be added to the description D. If $P'_W=0$, then no pruning takes place.

Frequency threshold. If elements of the frequency table are less than the frequency threshold P'_S then there will be no search for new complexes. It means that after creating the frequency table the command "IF $f_{jk}<P'_S$ THEN Let $f_{jk}=0$" would be executed. If $P'_S=1$, then no pruning takes place.

Deviation. Deviation P_D indicates the number of examples that are covered by complex but do not belong to the corresponding majority class. If the deviation in a set of examples extracted by the complex Com is less than the deviation threshold P'_D, then a rule r=(Com,c) can be added into description D. If the deviation threshold is $P'_D=0$, then rules will not be pruned.

Rule length. Rule length P_L is the number of selectors Sel_j not equal to the set of values of the attribute t_j. If the rule length is less than the threshold of the minimal rule length P_{Lmin} then the rule will not be added into the description D. Using the minimal rule length threshold allows to reduce the risk of overgeneralization caused by the small size of the set of examples or by the lack of negative examples. If $P_{Lmin}=0$, then no pruning takes place.

If the rule length increases over the threshold of the maximal rule length P_{Lmax} then the corresponding rule will not be added to the description D and new frequency table will not be calculated. Using the maximal rule length threshold allows to reduce the risk of overspecification. If $P_{Lmax}=M$, then all possible rules will be generated.

Common rule. All rules which map to the majority class C_m will not be produced by generating the concept description. Instead of it a rule r=(Com,C_m) will be added, where Com=(A_1,…,A_M). If no rule exists that covers object that ought to be classified then it will be classified as an object which belongs to the majority class. So, using the common rule we assume that an unknown object belongs most probably to the majority class.

VII. Conclusion

This paper described a monotone system approach for inductive learning. A new learning algorithm that rests on the theory of monotone systems is presented and discussed. A comparison of the effectiveness of the algorithm with ID3 on the well-known King-Rook-King task by D. Michie was presented. It showed that the base algorithm MONSIL produces much more rules than ID3, but it is more exact in classification and at the same time it is not very much slower than ID3. We can say also that using several pruning techniques described in the paper we can do it more effective. The main goal of the paper was presentation of monotone systems approach for inductive learning, a base algorithm MONSA and its development - the algorithm MONSIL.

References

[1] L. Võhandu, "Fast Methods in Exploratory Data Analysis," in *Transactions of Tallinn Technical University*, No 705, 1989, pp. 3-13.

[2] R. Kuusik, "The Super-Fast Algorithm of Hierarchical Clustering and the Theory of Monotone Systems," in *Transactions of Tallinn Technical University*, 734, 1993, pp. 37-62.

[3] L. Võhandu, R. Kuusik, A. Torim, E. Aab, G. Lind, "Some Monotone Systems Algorithms for Data Mining," in *WSEAS Transactions on Information Science and Applications*, Issue 4, Vol. 3, April 2006, pp. 802-809.

[4] I. Liiv, R. Kuusik, L. Võhandu, "Analytical CRM with conformity analysis," in *WSEAS Transactions on Systems and Control*, 2(2), 2007, pp. 155-161.

[5] S. Muggleton, M. Bain, J. Hayes-Michie, D. Michie, "An Experimental Comparison of Human and Machine Learning Formalisms," in *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, NY: Morgan Kaufmann, 1989, pp. 113-118.

[6] M. Gams, N. Lavrac, "Review of Five Empirical Learning Systems within a Proposed Schemata," in *I. Bratko, N. Lavrac (Eds.), Progress in Machine Learning, Proceedings of EWSL 87: 2nd European Working Session on Learning*, Bled, Yugoslavia, May 1987. Sigma Press, Wilmslow, 1987, pp. 46-66.

[7] L. Võhandu, "Express Methods of Data Analysis," in *Transactions of Tallinn Technical University*, No. 464, 1979, pp. 21-37 (in Russian).

[8] I. Mullat, "Extremal Monotone Systems," in *Automation and Remote Control*, No 5, pp. 130-139, 1976 (in Russian).

[9] R. Kuusik, G. Lind. Algorithm MONSA for All Closed Sets Finding: basic concepts and new pruning techniques. *WSEAS Transactions on Information Science and Applications*, 5(5), May 2008, pp. 599-611.

[10] J. R. Quinlan, "Learning efficient classification procedures and their application to chess end games," in *J. G. Carbonell, R. S. Michalski, T. M. Mitchell (Eds.), Machine Learning. An Artificial Intelligence Approach*, Springer-Verlag, 1984, pp. 463-482.