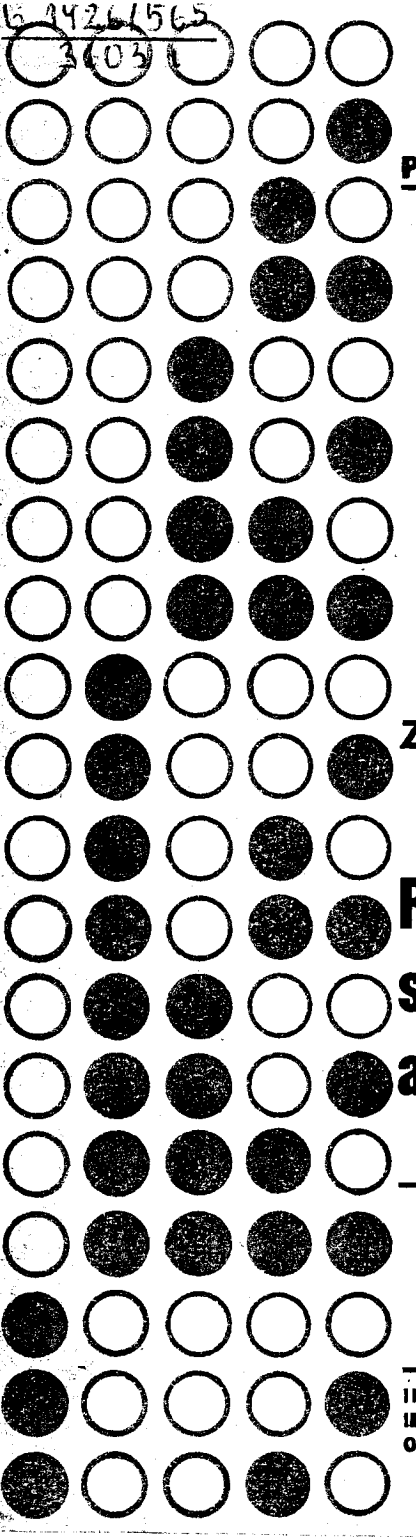

PRACE IPI PAN • ICS PAS REPORTS

Zdzisław Pawlak

Rough sets and some problems of artificial intelligence

565

July 1985

WARSZAWA

INSTYTUT PODSTAW INFORMATYKI POLSKIEJ AKADEMII NAUK
INSTITUTE OF COMPUTER SCIENCE POLISH ACADEMY OF SCIENCES
00-901 WARSAW, P.O. Box 22, POLAND

Zdzisław Pawlak

ROUGH SETS AND SOME PROBLEMS OF ARTIFICIAL INTELLIGENCE

565

Warsaw, July 1985

1. Introduction

We propose in this article a new, unified approach to some problems of artificial intelligence, such as machine learning, inductive inference, expert systems, decision-making theory, etc.

We advocate the use of the rough sets concept (Pawlak, 1982) as the mathematical basis for these areas.

The suggested approach enables a precise, mathematical formulation of fundamental concepts of these areas, yields new theoretical results and offers simple, effective algorithms.

Several experimental applications (in medicine, industrial process control and others) of the ideas presented here confirm theoretically obtained results.

The relevant literature concerning topics discussed in this article is enclosed at the end of the paper.

2. Basic notions

2.1. Information systems

In many problems of AI, such as machine learning, expert systems, etc., we are given a set of objects (states, phases) and each object under consideration is characterized in terms of some features. Moreover the set of objects is classified into disjoint family of classes and we want to characterize each class in terms of features of objects belonging to that class.

We shall investigate the problem in detail, and in order to do so we exploit the concept of information system (Pawlak, 1981) and rough classification (Pawlak, 1984).

By an information system we mean a collection of data concerning some objects, states, processes etc. For example a data file concerning patients suffering from a certain disease is an information system. Each patient in the file is characterised by means some symptoms like, for example blood preasure, temperature etc. Each symptom may assume a certain value from a fixed set of values associated with the symptom. For example the symptom "blood preasure" may assume one of the following values: normal, above normal, below normal. Thus each object in an information system is characterised by a set of preassumed attributes (features) values.

Formal definition of an information system is given below. By an information system we mean the 4-tuple

$$S = (U, Q, V, \varrho) ,$$

where

U - is a finite set of objects,

Q - is a finite set of attributes,

$V = \bigcup_{q \in Q} V_q$ and V_q - is the domain of the attribute q,

$\varrho: U \times Q \to V$ - is a total function such that $\varrho(x,q) \in V_q$ for every $q \in Q$, $x \in U$, called the information function.

The function $\varrho_x: Q \to V$ such that $\varrho_x(q) = \varrho(x,q)$ for every $x \in U$, $q \in Q$ will be called information (data, knowledge, description) about x in S.

Any pair (q,v), $q \in Q$, $v \in V_q$ is called a descriptor in S.

Any function φ from Q to V such that $\varphi(q) \in V_q$ will be called an information in S.

Thus an information system may be considered as a finite table in which columns are labelled by attributes, rows are labelled by objects and the entry in the q-th column and x-th row has the value $\varrho(x,q)$.

Each row in the table represents an information (about some object in S).

An example of information system is shown in Tab. 1.

Example 2.1.1.

U	p	q	r
x_1	1	0	2
x_2	0	1	1
x_3	2	0	0
x_4	1	1	0
x_5	1	0	2
x_6	2	0	0
x_7	0	1	1
x_8	1	1	0
x_9	1	0	2
x_{10}	0	1	1

Tab. 1.

Let $S = (U,Q,V,\varrho)$ be an information system, and $P \subseteq Q$. An information system $S' = (U,P,V',\varrho')$ $(S' = (X,Q,V',\varrho'))$ such that $\varrho' = \varrho/U \times P$ $(\varrho' = \varrho/X \times Q)$ and V' is the domain of ϱ' will be called a P-restriction (X-restriction) of S, and will be denoted $S/P(S/X)$.

2.2. Indiscernibility relation

Let $S = (U,Q,V,\varrho)$ be an information system and let $P \subseteq Q$, $x,y \in U$.

By $\widetilde{P}$ we mean a binary relation on U (called an indiscernibility relation) - defined as follows:

We say that x and y are indiscernible by the set of attributes P in S $(x \widetilde{P} y)$ iff $\varrho_x(q) = \varrho_y(q)$ for every $q \in P$.

One can easily check that $\widetilde{P}$ is an equivalence relation in U for every $P \subseteq Q$.

The equivalence classes of the relation $\widetilde{P}$ are called P-elementary sets in S. Q-elementary sets are also called atoms of S.

Thus every $P \subseteq Q$ defines a classification (partition) of U - denoted P^*, and the equivalence classes of the relation $\widetilde{P}$ are classes (blocks) of the classification P^*.

Certainly $P^* = U/\widetilde{P}$. We shall use the notation $U/\widetilde{P}$ when speaking about relations, and P^* when speaking about classifications.

Example 2.2.1.

Some elementary sets in the information system presented

in Tab. 1. are shown below:

i) p-elementary sets

$X_1 = \{x_1, x_4, x_5, x_8, x_9\}$

$X_2 = \{x_2, x_7, x_{10}\}$

$X_3 = \{x_3, x_6\}$

ii) $\{p,r\}$-elementary sets

$Y_1 = \{x_1, x_5, x_9\}$

$Y_2 = \{x_2, x_7, x_{10}\}$

$Y_3 = \{x_3, x_6\}$

$Y_4 = \{x_4, x_8\}$

iii) atoms

$Z_1 = \{x_1, x_5, x_9\}$

$Z_2 = \{x_2, x_7, x_{10}\}$

$Z_3 = \{x_3, x_6\}$

$Z_4 = \{x_4, x_8\}$

If $\widetilde{P}$ and $\widetilde{R}$ are equivalence relations, then $\widetilde{T} = \widetilde{P} \cap \widetilde{R}$ is called an intersection of $\widetilde{P}$ and $\widetilde{R}$, and is defined as follows:

$x \widetilde{T} y$ iff $x \widetilde{P} y$ and $x \widetilde{R} y$

It can easily be seen that

$$\widetilde{P} = \bigcap_{q \in P} \widetilde{q} \quad \text{for every} \quad P \subseteq Q.$$

$P \subseteq Q$ in every S.

An information system S is <u>selective</u> iff all atoms in S are one-element sets, i.e. $\widetilde{Q}$ is an identity relation.

2.3. Representation of an Information System

Let $S = (U,Q,V,\varrho)$ be an information system and let $P \subseteq Q$.

A <u>P-representation</u> of S is an information system

$$S_P = (U/\widetilde{P}, P, V_P, \varrho_P),$$

where $U/\widetilde{P}$ is the family of all equivalence classes of the relation $\widetilde{P}$, $V_P = \bigcup_{q \in P} V_q$, and

$$\varrho_P : U/\widetilde{P} \times P \to V_P$$

is the information function such that

$$\varrho_P([x]_{\widetilde{P}}, q) = \varrho(x,q)$$

for every $x \in U$, $q \in P$. ($[x]_{\widetilde{P}}$ - denotes an equivalence class of the relation $\widetilde{P}$ containing the object x).

Thus in a P-representation of S objects are P-elementary sets, and the information function ϱ_P is an extension of the function ϱ for P-elementary sets. Of course, every P-representation of any information system $S = (U,Q,V,\varrho)$, $P \subseteq Q$, is selective.

<u>Example 2.3.1.</u>

Examples of representations of the information system shown in Tab. 1. are given below:

$U/\widetilde{P}$	p
X_1	1
X_2	0
X_3	2

Tab. 2

$U/\{\widetilde{p,r}\}$	p	r
Y_1	1	2
Y_2	0	1
Y_3	2	0
Y_4	1	0

Tab. 3

$U/\{\widetilde{p,q,r}\}$	p	q	r
Z_1	1	0	2
Z_2	0	1	1
Z_3	2	0	0
Z_4	1	1	0

Tab. 4

■

2.4. Approximation of Sets in an Information System

Let $S = (U,Q,V,\varrho)$ be an information system, $X \subseteq U$ and $P \subseteq Q$.

By the P-lower (P-upper) approximation of $X \subseteq U$ in S, we mean the sets $\underline{P}X$ ($\overline{P}X$) defined as follows:

$$\underline{P}X = \{x \in U: [x]_{\widetilde{P}} \subseteq X\}$$

$$\overline{P}X = \{x \in U: [x]_{\widetilde{P}} \cap X \neq \emptyset\}$$

The set

$$Bn_P(X) = \overline{P}X - \underline{P}X$$

is referred to as the P-boundary of X in S.

It is easy to check that each information system $S = (U,Q,V,\varrho)$ and each subset of attributes $P \subseteq Q$ define a topological space $T_S = (U, Def_P(S))$, where $Def_P(S)$ is the family of all P-definable sets in S, and the lower and upper approximations are interior and closure in the topological space T_S. Hence the approximations have the following properties:

1) $\underline{P}X \subseteq X \subseteq \overline{P}X$
2) $\underline{P}\emptyset = \overline{P}\emptyset = \emptyset$; $\underline{P}U = \overline{P}U = U$
3) $\underline{P}(X \cup Y) \supseteq \underline{P}X \cup \underline{P}Y$
4) $\overline{P}(X \cup Y) = \overline{P}X \cup \overline{P}Y$

5) $\underline{P}(X \cap Y) = \underline{P}X \cap \underline{P}Y$
6) $\overline{P}(X \cap Y) \subseteq \overline{P}X \cap \overline{P}Y$
7) $\underline{P}(-X) = -\overline{P}(X)$
8) $\overline{P}(-X) = -\underline{P}(X)$

Moreover for the topological space T_S we have:

9) $\underline{P}\underline{P}X = \overline{P}\underline{P}X$
10) $\overline{P}\overline{P}X = \underline{P}\overline{P}X$

$\underline{P}X$ is called the P-positive region of X in S;

Bn_PX is called the P-doubtful region of X in S;

$U - \overline{P}X$ is called the P-negative region of X in S.

Example of approximations in information system given in Tab. 1 are shown below:

Let $X = \{x_1,x_2,x_3,x_6\}$

and $Q = \{p,q,r\}$.

$\underline{p}X = X_3 = \{x_3,x_6\}$

$\overline{p}X = X_1 \cup X_2 \cup X_3 = U$

$\underline{Q}X = Z_3 = \{x_3,x_6\}$

$\overline{Q}X = Z_1 \cup Z_2 \cup Z_3 = \{x_1,x_2,x_3,x_5,x_6,x_7,x_9,x_{10}\}$.

2.5. Accuracy of approximation

With every subset $X \subseteq U$ we associate a number $\mu_P(X)$

called the accuracy of approximation of X by P in S, or, in short, the accuracy of X, where P and S are defined as follows:

$$\mu_P(X) = \frac{\text{card } \underline{P}X}{\text{card } \overline{P}X}$$

$\mu_P(X)$ can be expressed as:

$$\mu_P(X) = \frac{\underline{\mu}_P(X)}{\bar{\mu}_P(X)}$$

where

$$\underline{\mu}_P(X) = \frac{\text{card } \underline{P}X}{\text{card } U}$$

and

$$\bar{\mu}_P(X) = \frac{\text{card } \overline{P}X}{\text{card } U}$$

Of course, if $X \subseteq Y$, then

$$\mu_P(X) \leq \mu_P(Y).$$

Moreover we have

$$\mu_P(-X) = \frac{1 - \bar{\mu}_P(X)}{1 - \underline{\mu}_P(X)}$$

Because of properties 3) and 6) (section 2.4.) we are unable to express the accuracy of the union and the intersection of sets X,Y in terms of the accuracies of X and Y.

2.6. Non-definable Sets

Let $S = (U,Q,V,\varrho)$ be an information system and let $P \subseteq Q$, $X \subseteq U$.

Note that X is P-definable in S iff $\underline{P}X = \overline{P}X$.

We shall classify non-definable sets into the following classes:

a) X is roughly P-definable in S, iff $\underline{P}X \neq \emptyset$ and $\overline{P}X \neq U$.

b) X is internally P-non-definable in S, iff $\underline{P}X = \emptyset$.

c) X is externally P-non-definable in S, iff $\overline{P}X = U$

d) X is totally P-non-definable in S, iff $\underline{P}X = \emptyset$ and $\overline{P}X = U$.

Let us remark that if X is definable, roughly definable, or totally non-definable, so is $-X$; if X is internally (externally) non-definable, then $-X$ is externally (internally) non-definable.

2.7. Approximation of families of sets

Let $S = (U,Q,V,\varrho)$ be an information system, $P \subseteq Q$,

and let $\mathfrak{X} = \{X_1, X_2, \ldots, X_n\}$, $X_i \subseteq U$ $i \geqslant 2$, be a family of subsets of U.

By the P-lower (P-upper) approximation of $\mathfrak{X}$ in S, denoted $\underline{P}\mathfrak{X}$ ($\overline{P}\mathfrak{X}$), we mean sets

$$\underline{P}\mathfrak{X} = \{\underline{P}X_1, \underline{P}X_2, \ldots, \underline{P}X_n\}$$

and

$$\overline{P}\mathfrak{X} = \{\overline{P}X_1, \overline{P}X_2, \ldots, \overline{P}X_n\},$$

respectively.

If $\mathfrak{X}$ is a classification (a partition) of U, i.e. $X_i \cap X_j = \emptyset$ for every $i,j \leqslant n$, $i \neq j$ and $\bigcup_{i=1}^{n} X_i = U$, then X_i are called classes (blocks) of $\mathfrak{X}$.

If every class of $\mathfrak{X}$ is P-definable then the classification $\mathfrak{X}$ will be called P-definable.

$Pos_P(\mathfrak{X}) = \bigcup_{i=1}^{n} \underline{P}X_i$ will be called the P-positive region of the classification $\mathfrak{X}$ in S;

Since $U = \bigcup_{i=1}^{n} \overline{P}X_i$, there is no P-negative region for any P of the classification $\mathfrak{X}$ in S.

$Bn_P(\mathfrak{X}) = \bigcup_{i=1}^{n} Bn_P X_i$ will be called the P-doubtful region of the classification $\mathfrak{X}$ in S;

If $\mathfrak{X} = \{X_1, X_2, \ldots, X_n\}$ is a classification of U, then

$$\alpha_P(\mathfrak{X}) = \frac{\sum_{i=1}^{n} \operatorname{card}(\underline{P}X_i)}{\sum_{i=1}^{n} \operatorname{card}(\overline{P}X_i)}$$

will be called the accuracy of the approximation of $\mathfrak{X}$ by P in S, or simply the accuracy of $\mathfrak{X}$.

$\alpha_P(\mathfrak{X})$ expresses the ratio of all positive decisions to all possible decisions, when objects are classified by the set of attributes P.

We can also introduce another coefficient called quality of approximation of the classification $\mathfrak{X} = \{X_1, X_2, \ldots, X_n\}$ by the set P of attributes, defined as follows:

$$\gamma_P(\mathfrak{X}) = \frac{\sum_{i=1}^{n} \operatorname{card} \underline{P}X_i}{\operatorname{card}(U)}$$

Quality $\gamma_P(\mathfrak{X})$ expresses the ratio of all P-correctly classified objects to all objects in the system.

Obviously $\alpha_P(\mathfrak{X}) \leqslant \gamma_P(\mathfrak{X})$ and $\alpha_P(\mathfrak{X}) = \gamma_P(\mathfrak{X})$ iff $\mathfrak{X}$ is P-definable.

2.8. Dependence of attributes

Let $S = (U,Q,V,\varrho)$ be an information system and let $P,R \subseteq Q$, be subsets of attributes.

We say that set of attributes R depends on the set of attributes P in S, $P \Rightarrow_S R$ (or in short $P \Rightarrow R$) iff $\widetilde{P} \subseteq \widetilde{R}$.

One can show by simple computation the following properties:

Fact 2.8.1.

The following conditions are equivalent:

1) $P \Rightarrow_S R$

2) $\widetilde{P} \subseteq \widetilde{R}$

3) $\widetilde{P \cup R} = \widetilde{P}$

4) R^* is P-definable in S

5) $\underline{P}(R^*) = \overline{P}(R^*)$

6) $\gamma_P(R^*) = \gamma_P(R^*) = 1$

Fact 2.8.2.

1) If $P \Rightarrow_S R$ and $P' \supseteq P$, then $P' \Rightarrow_S R$,

2) If $P \Rightarrow_S R$ and $R' \subseteq R$, then $P \Rightarrow_S R'$,

3) If $P \Rightarrow_S R$, then $P \Rightarrow_{S_{P \cup R}} R$.

A simple algorithm for checking whether $P \Rightarrow_S R$ or not results from properties 3) and 5).

Note that 3) and 5) yield the property $P \Rightarrow_S R$ iff $S_{P \cup R}/P$ is selective.

This is to say that if we remove all duplicate rows,

and all columns labelled by attributes not belonging to $P \cup R$, then we obtain $P \cup R$ representation of S, which is of course selective. Having done this we check whether removing from system $S_{P \cup R}$ all columns labelled by attributes from R yields a selective system, i.e., a system with no duplicate rows. If this is the case, then $P \Rightarrow_S R$ holds, otherwise the dependence $P \Rightarrow_S R$ is not valid.

Example 2.8.1.

For example in order to check whether $\{p,q\} \Rightarrow r$ in the information system given in Tab. 1, we first compute $\{p,q,r\}$ - representation of that system, which is the following system:

$U/\widetilde{\{p,q,r\}}$	p	q	r
Z_1	1	0	2
Z_2	0	1	1
Z_3	2	0	0
Z_4	1	1	0

Tab. 6.

Removing now the column labelled by the attribute r we obtain the following table:

p	q
1	0
0	1
2	0
1	1

Tab. 7.

For the sake of simplicity we omit in the table the column containing objects. Because all rows are different (the system is selective) the dependence $\{p,q\} \to r$ holds, i.e., $\{\widetilde{p,q}\} \subset \widetilde{r}$, which is equivalent to $\{p,q\} \cup \{r\} = \{\widetilde{p,q}\}$.

Table 8 below presents the dependence function:

p	q	r
1	0	2
0	1	1
2	0	0
1	1	0

Tab. 8.

We say that R <u>roughly depends</u> on P in S, iff $0 < \gamma_P(R^*) < 1$. Then we write $P \overset{k}{\Rightarrow} R$, where $k = \gamma_P(R^*)$.

If $P \overset{k}{\Rightarrow} R$, then the dependence $P \to R$ holds for some objects only, namely for all $x \in Pos_P(R^*)$, i.e., the objects belonging to P-possitive region of the classification R^*. The number $\gamma_P(R^*)$ indicates the percentage of objects for which the dependence $P \to R$ holds. In other words $P \overset{k}{\Rightarrow} R$ iff $P \to R$ in $S/Pos_P(R^*)$.

We say that R is <u>totally independent</u> on P in S iff $\gamma_P(R^*) = 0$.

The meaning of this definition is obvious.

2.9. <u>Reduction of attributes</u>

Let $S = (U,Q,V,\varrho)$ be an information system and let $P \subseteq Q$.

a) $P \subseteq Q$ is <u>independent</u> in S iff for every $P' \subset P$, $\widetilde{P'} \supset \widetilde{P}$.

b) $P \subseteq Q$ is <u>dependent</u> in S iff there exist $P' \subset P$, such that $\widetilde{P'} = \widetilde{P}$.

c) $P \subseteq Q$ is <u>reduct</u> of Q in S iff P is the greatest independent set in Q.

It can easily be shown that the following properties hold:

<u>Fact 2.9.1</u>.

a) If P is independent in S, then for every $p,q \in P$ neither $p \Rightarrow q$ nor $q \Rightarrow p$, i.e., all attributes from P

are pairwise independent.

b) If P is dependent in S, then there exists $P' \subset P$, independent in S, such that $P' \Rightarrow P-P'$;

Note that an information system may have more than one reduct. For example in the information system shown in Tab. 1 there are three reducts: $\{p,q\}$, $\{p,r\}$, and $\{q,r\}$.

More properties of reducts can be found in Pawlak (1981).

2.10. The Language of Information System (The Description Language)

With each information system $S = (U,Q,V,\varrho)$ we associate a language L_S called a description language of S.

We recall after Pawlak (1981) the basic definitions concerning the description language needed in this paper.

First we define the set of terms T_S of the language L_S. It is the least set satisfying the conditions:

a) Constants 0,1 and all descriptors (i.e., pairs (q,v), $q \in P$, $v \in V_q$) are terms of L_S. (We shall also write down descriptors in the following form: $q:=v$).

b) If t and s are terms in L_S, then so are $\sim t$, $t+s$, ts.

Now we shall define the set of formulas F_S of L_S.

The set of formulas is the least set satisfying the conditions:

a) Constants T,R are formulas in L_S; if t,s are terms in L_S, then $t = s$ and $t \Rightarrow s$ are formulas in L_S.

b) If φ, ψ are formulas in L_S, then $\sim\varphi$, $\varphi \vee \psi$ and $\varphi \wedge \psi$ are formulas in L_S.

Now we shall define the semantics of L_S.

The semantics of terms is a function σ_S (or σ - when S is understood), which associates the subset of objects to terms, i.e., $\sigma_S: T_S \to \wp(U)$, defined as follows:

a) $\sigma(0) = \emptyset$, $\sigma(1) = U$

b) $\sigma(q,v) = \{x \in U: \varrho_x(q) = v\}$

c) $\sigma(\sim t) = U - \sigma(t)$

d) $\sigma(t + s) = \sigma(t) \cup \sigma(s)$

e) $\sigma(t \cdot s) = \sigma(t) \cap \sigma(s)$

The semantics of formulas is a function σ_S^* which assigns to each formula its truth value T or F (truth or falsity), i.e., $\sigma_S^*: F_S \to \{T,F\}$.

If S is known we shall omit the subscript S and write σ^* instead of σ_S^*.

σ^* is defined in the following manner:

a) $\sigma^*(T) = T$, $\sigma^*(F) = F$

b) $\sigma^*(t = s) = \begin{cases} T, & \text{if } \sigma(t) = \sigma(s) \\ F, & \text{otherwise} \end{cases}$

c) $\sigma^*(t \Rightarrow s) = \begin{cases} T, & \text{if } \sigma(t) \subseteq \sigma(s) \\ F, & \text{otherwise} \end{cases}$

d) $\sigma^*(-\varphi) = \begin{cases} T, & \text{if } \sigma^*(\varphi) = F \\ F, & \text{if } \sigma^*(\varphi) = T \end{cases}$

e) $\sigma^*(\varphi \vee \psi) = \sigma^*(\varphi) \vee \sigma^*(\psi)$

f) $\sigma^*(\varphi \wedge \psi) = \sigma^*(\varphi) \wedge \sigma^*(\psi)$

If $\sigma^*(\varphi) = T$ we say that φ is <u>true</u> in S, and if $\sigma^*(\varphi) = F$, then φ is said to be <u>false</u> in S.

If φ is true in S we shall write $\vdash_S \varphi$, or $\vdash \varphi$ if S is known.

We assume substitutions of the axioms of Boolean algebra for terms and substitutions of the propositional calculus axioms for formulas; moreover we assume one specific axiom of the form:

$$(q,v) = \sum_{\substack{u \neq v \\ u \in V_q}} (q,u),$$

for every $q \in P$.

Let $S = (U,Q,V,\varrho)$ be an information system, L_S, a description language.

We say that $t \in L_S$ is <u>P-elementary</u> iff $t = \prod_{q \in P} (q,v)$, $v \in V_q$.

A term $t \in L_S$ is in <u>P-normal form</u> iff $t = \sum t_i$, where t_i are P-elementary terms.

<u>Fact 2.10.1</u>.

For every $t \in L_S$ and $P \subseteq Q$, there exists $t' \in L_S$ in P-normal form such that $\vdash t = t'$.

We say that $X \subseteq U$ is <u>P-describable</u> in L_S iff there exists a term $t \in L_{S/P}$ such that $\sigma_S(t) = X$. The term t is called the <u>description</u> of X in $L_{S/P}$.

<u>Fact 2.10.2</u>.

$X \subseteq U$ is P-describable in L_S iff X is P-definable in S.

Note that P-elementary terms are descriptions of P-elementary sets.

More details concerning the description language can be found in Marek and Pawlak (1976).

Let $S = (U,Q,V,\varrho)$ be an information system, let $P,R \subseteq Q$ and $t \in L_{S/P}$, $t' \in L_{S/R}$.

Each formula of the form $t \Rightarrow t'$ will be called a <u>(P,R)-decision (classification)</u> rule in L_S. If $t \Rightarrow t'$ is a (P,R)-decision rule we shall also write $t \Rightarrow t'$.

The term t is referred to as a predecessor and t′ successor of the rule.

If t and t′ are P-elementary and R-elementary terms in $L_{S/P}$ and $L_{S/R}$, respectively, then the rule $t \Rightarrow t'$ is called (P,R)-elementary.

If t′ is R-elementary, then the rule $t \Rightarrow t'$ will be called R-deterministic; if t′ is a union of R-elementary terms, then $t \Rightarrow t'$ is called P-non-deterministic.

Let $S = (U,Q,V,\varrho)$ be an information system, $P,R \subseteq Q$, and $t \Rightarrow s$ a (P,R)-decision rule in L_S. A rule $t' \Rightarrow s'$ such that t′ and s′ are Q-normal forms of terms t and s, respectively, will be called Q-normal form of (P,R)-decision rule $t \Rightarrow s$, and denoted by $t \tilde{Q} s$. Of course, $t \Rightarrow s$ is true iff $t' \tilde{Q} s'$ is true.

Fact 2.10.3.

A (P,R)-decision rule $t \Rightarrow s$ is true in S iff all Q-elementary terms occuring in a Q-normal form of t occurs in a Q-normal form of s.

This property enables us to prove the validity of any decision rule in a simple syntactical way.

Example 2.10.1.

Consider the information system given in Tab.9,

U	p	r
x_1	1	0
x_2	1	1
x_3	0	2

Tab. 9.

and the (p,r)-decision rule $(p:=1) \Rightarrow (r:=0)$. In order to check whether this rule is true or not, we present it in $\{p,r\}$ - normal form as shown below:

$$(p:=1)(r:=0) + (p:=1)(r:=1) \Rightarrow (p:=1)(r:=0)$$

Because the elementary term $(p:=1)(r:=1)$ occurs only in the predecessor of the normal form decision rule, for the rule $(p:=1) \Rightarrow (r:=0)$ is false.

We can also check the validity of the formula directly from the definition of the semantics of formulas, namely:

$$\sigma(p:=1) = \{x_1, x_2\}$$

and

$$\sigma(r:=0) = \{x_1\}$$

hence the decision rule is not true.

On the other hand the decision rule $(r:=0) \Rightarrow (p:=1)$ is true because the normal form of the rule has the form:

$$(p:=1)(r:=0) \Rightarrow (p:=1)(r:=0) + (p:=1)(r:=1)$$

and the only one (p,r)-elementary term $(p:=1)(r:=0)$ in the predecessor occurs in the successor of the rule.

From the semantic definition we have that

$$\sigma(r:=0) \subset \sigma(p:=1)$$

hence the decision rule is true.

■

Any finite set of decision rules in L_S will be called a decision (classification) algorithm in L_S.

We shall also write decision algorithms as below:

$t_1 \Rightarrow t_1'$
$t_2 \Rightarrow t_2'$
$\vdots$
$t_n \Rightarrow t_n'$

A decision algorithm is deterministic if all its decision rules are deterministic; otherwise the algorithm is non-deterministic.

If all decision rules of an algorithm are (P,R)-elementary then the algorithm is said to be in (P,R)-normal form.

Let

$t_1 \Rightarrow t_1'$
$t_2 \Rightarrow t_2'$
$\vdots$
$t_n \Rightarrow t_n'$

be a decision algorithm α, then the formula

$$\Psi_\alpha = \bigwedge_{i=1}^{n} t_i \Rightarrow t_i'$$

will be called a decision (classification) formula of the algorithm α.

We say that the algorithm α is correct in S if Ψ_α is true in S.

An algorithm in (P,E)-normal form in S is maximal in S iff α is the set of all (P,R)-elementary rules $t_i \Rightarrow t_i'$ in S such that there exists a non-empty P∪R-elementary term $t_i \cdot t_i'$ in $L_S/P \cup R$.

Example 2.10.2.

In the information system given in Tab. 4 (Section 2.3)

the following is the maximal decision algorithm in $(\{p,q\},r)$-normal form:

$(p:=1)(q:=0) \Rightarrow (r:=2)$
$(p:=0)(q:=1) \Rightarrow (r:=1)$
$(p:=2)(q:=0) \Rightarrow (r:=0)$
$(p:=1)(q:=1) \Rightarrow (r:=0)$

■

Of course there is only one maximal decision algorithm in (P,R)-normal form for every $P,R \subseteq Q$ in every information system $S = (U,Q,V,\varrho)$.

Let σ be a maximal decision algorithm in (P,R)-normal form in S. One can easily show the following important property:

Fact 2.10.4

$\vdash_S \Psi_\alpha$ iff there does not exist in σ two decision rules with the same predecessor and different successors.

Thus we have two methods of checking whether $P \Rightarrow R$ or not: we can use the semantic method using property 2), section 2.8, or we can use the syntactic method, proving the validity of the corresponding formula with the aid of

Fact 2.10.4.a).

Let us notice that the maximal decision algorithm is a counterpart of the dependence function; in other words, the decision algorithm is a linguistic representation of the dependence function.

To this end let us the property of algorithms which can be used as transformation rules for classification algorithms.

Fact 2.10.5.

$$\vdash \bigwedge_{i=1}^{n} (t_i \Rightarrow t) \quad \text{iff} \quad \vdash \left(\sum_{i=1}^{n} t_i \Rightarrow t\right)$$

3. Applications

3.1. Static Learning

Machine learning from examples (see Michalski ed. (1982)) can be very easily formulated in our approach leading to new important theoretical and practical results.

In order to avoid confusion with the existing terminology we introduce new terms for machine learning: static and dynamic learning, discussed in two successive sections of this paper.

Let us first consider static learning.

Suppose we are given a finite set U of objects.

Elements of U are called <u>training examples (instances)</u> and U is called <u>training set</u>. Assume further that U is classified into disjoint classes $X_1, X_2, \ldots, X_n$ $(n \geq 2)$ by a <u>teacher (expert, environment)</u>. The classification represents the teacher's knowledge of objects from U. Furthermore let us assume that a student is able to characterize each object from U in terms of attributes from set P. Description of objects in terms of attributes from P represents the student's knowledge of objects from U.

We can say that the teacher has <u>semantic</u> knowledge and the student, <u>syntactical</u> knowledge of objects from U.

The problem we are going to discuss in this section is whether the student's knowledge can be matched with the teacher's knowledge, or, more precisely, whether the teacher's classification can be described in terms of attributes available to the student.

Thus static learning consists in describing classes $X_1, X_2, \ldots, X_n$ in terms of attributes from P, or more exactly, in finding a classification algorithm which provides the teacher's classification on the basis of properties of objects expressed in terms of attributes from P.

The problem of static learning can be formulated precisely in terms of concepts introduced in the previous sections as follows:

Let $S = (U, P, V, \varrho)$ be an information system, associated

with the student's knowledge of elements of U. Note that ϱ_x is <u>student's knowledge</u> (information) about x in S. Let us extend system S by adding a new attribute e representing the classification provided by the teacher, i.e., $e^* = \{X_1, X_2, \ldots, X_n\}$. Thus we obtain a new information system $S' = (U, Q, V', \varrho')$, where $Q = P \cup \{e\}$, $P \cap \{e\} = \emptyset$, $V' = V \cup \{1, 2, \ldots, n\}$, $\varrho'/U \times P = \varrho$, and $\varrho'_x(e) = i$ iff $x \in X_i$. $\varrho'_x(e)$, be called teacher's knowledge of x in S', is the number of the class to which x belongs according to the teacher's knowledge.

Thus the problem of static learning reduces to the question whether the classification e^* is P-definable. In virtue of property 1) section 2.8, e^* is P-definable iff $P \to e$, i.e., the problem of whether there exists an algorithm to "learn" classification e^* by checking the properties of objects reduces to proving whether the attribute e depends on the set of attributes P in S'.

This can easily be done by methods shown in previous sections.

If the dependence $P \to e$ holds, one can formulate a (P,e)-decision algorithm $\mathcal{A}$, which represents the dependence function. In other words the algorithm can be used directly as a learning algorithm.

Because the algorithm is a set of decision (classification) rules, this means that learning a classification consists in finding classification rules.

Example 3.1.1.

Let us consider for example an information system given in Tab. 1, section 2.1, and let us assume that the attribute r in that system represents a classification r^*, provided by the teacher. We ask whether the classification can be expressed by attributes p and q.

Because the dependence $\{p,q\} \to r$ holds, the learning algorithm exists, and it has the form (see Tab. 8):

$(p:=1)(q:=0) \Rightarrow (r:=2)$
$(p:=0)(q:=1) \Rightarrow (r:=1)$
$(p:=2)(q:=0) \Rightarrow (r:=0)$
$(p:=1)(q:=1) \Rightarrow (r:=0)$

Note that we are not allowed to remove p or q because the set $\{p,q\}$ is independent in S.

It may happen, however, that the teacher classification e^* is not Q-definable. That is to mean that the learning algorithm does not exist, and it is impossible to classify objects correctly by examining their features.

In such a case it is possible to classify objects only approximately, i.e., to approximate the classification e^* by the set of attributes Q. This is to say that we are unable to classify every object correctly; only some objects (possibly zero) can be classified properly in this case.

Obviously there is no deterministic classification algorithm in the case of an approximate classification, but there is non-deterministic one.

The dependence function must be replaced by dependence relation (or dependence multifunction) for approximate classifications.

The co-efficients of accuracy and quality of the approximate classification show what part of objects can be classified correctly (quality) and what part of decisions can be correct (accuracy).

Of course both co-efficients are less than one.

The example below illustrates the above situation.

Example 3.1.2.

Let us consider an information system shown in Tab. 9.

U	p	q	r
x_1	1	0	2
x_2	0	1	1
x_3	2	0	0
x_4	1	0	2
x_5	1	0	0
x_6	0	1	1
x_7	2	0	0
x_8	1	0	0

x_9	0	1	1
x_{10}	2	0	0
x_{11}	1	0	0
x_{12}	1	0	2

Tab. 9

Let us assume that the attribute r represents the teacher knowledge, so that r^* is the teacher classification.

The representation of the system (with respect to all attributes) is shown in Tab. 10.

$U/\tilde{Q}$	p	q	r
Z_1	1	0	2
Z_2	0	1	1
Z_3	2	0	0
Z_4	1	0	0

Tab. 10

where

$$Z_1 = \{x_1, x_4, x_{12}\}$$
$$Z_2 = \{x_2, x_6, x_9\}$$
$$Z_3 = \{x_3, x_7, x_{10}\}$$
$$Z_4 = \{x_5, x_8, x_{11}\}$$

are atoms of the system.

It can easily be seen from Tab. 10 that the classification r^* is not (p,q)-definable. Hence we can approximate the classification r^* by set of attributes $\{p,q\}$.

In order to do that let us first compute classes of the classification r^* (equivalence classes of relation $\tilde{r}$), which are as follows:

$$Y_1 = \{x_1, x_4, x_{12}\}$$
$$Y_2 = \{x_2, x_6, x_9\}$$
$$Y_3 = \{x_3, x_5, x_7, x_9, x_{10}, x_{11}\}$$

$$X_1 = \{x_1, x_4, x_5, x_8, x_{11}, x_{12}\}$$
$$X_2 = \{x_2, x_6, x_9\}$$
$$X_3 = \{x_3, x_7, x_{10}\}$$

Now are the equivalence classes of relation $\{p,q\}$.

Let us set $P = \{p,q\}$. Then the following sets are the lower P-approximation of r^*:

$$\underline{P}Y_1 = \emptyset$$
$$\underline{P}Y_2 = X_2$$
$$\underline{P}Y_3 = X_3$$

and the upper P-approximation of r^* is:

$\overline{P}Y_1 = X_1$

$\overline{P}Y_2 = X_2$

$\overline{P}Y_3 = X_1 \cup X_3$

Thus the class Y_1 is internally P-non-definable, Y_2 is P-definable, and Y_3 is roughly P-definable.

The corresponding accuracy co-efficients are:

$\mu_P(Y_1) = 0$

$\mu_P(Y_2) = 1$

$\mu_P(Y_3) = 0,5.$

Thus it is impossible to learn positive instances of Y_1, but it is possible to learn negative instances of Y_1, (if $x \in Y_2 \cup Y_3$ we know that x is not in Y_1).

In other words, it is impossible to classify correctly x_1, x_4, x_{12} by observing their features expressed by p and q.

Y_2 can be learned fully, i.e., all elements of Y_2 can be classifical correctly on the basis of their features expressed by p and q.

Y_3 can be learned only roughly, i.e., only objects x_3, x_7, x_{10} can be recognized on the basis of p and q as elements of Y_3; objects x_2, x_6, x_9 can be excluded from Y_3, and $X_1 = \{x_1, x_4, x_5, x_8, x_{11}, x_{12}\}$ is the doubtful region of Y_3, i.e., it cannot be decided on the basis

of p and q whether the elements of X_1 are, or are not, in Y_3.

The non-deterministic classification algoritam is shown below:

$(p:= 0)(q:= 1) \Rightarrow (r:= 2) + (r:= 0)$

$(p:= 1)(q:= 0) \Rightarrow (r:= 1)$

$(p:= 2)(q:= 0) \Rightarrow (r:= 0)$

The dependence relation (multifunction) is shown in Tab. 11.

p, q	r
0 1	{2,0}
1 0	1
2 0	0

Tab. 11.

The accuracy and quality of learning are:

$$\beta_p(r^*) = \frac{\text{card}(\underline{P}Y_2) + \text{card}(\underline{P}Y_3)}{\text{card}(\overline{P}Y_1) + \text{card}(\overline{P}Y_2) + \text{card}(\overline{P}Y_3)} = 9/15 = 0,6,$$

$$\gamma_p(r^*) = \frac{\text{card}(\underline{P}Y_2) + \text{card}(\underline{P}Y_3)}{\text{card}(U)} = 9/12 = 0,75,$$

which means that at most 75 per cent of instances can be classified correctly and at most 60 per cent decision can be correct.

Note also that $\{p,q\}$ has one reduct, namely p. This means that it is not necessary to have both p and q to learn the classification r^* but it is enough to use p only. The classification algorithm can thus be simplified as follows:

$(p:= 0) \Rightarrow (r:= 2) + (r:= 0)$
$(p:= 1) \Rightarrow (r:= 1)$
$(p:= 2) \Rightarrow (r:= 0)$

and the dependence relation takes on the form:

p	r
0	{2,0}
1	1
2	0

Tab. 12.

■

The ideas presented in this section were applied to computer-supported medical diagnosis algorithms, resulting in

a new simple method of medical data analysis.

3.2. Dynamic Learning

Static learning consists in a description of objects by a student classification provided by the teacher, or in other words, in learning classification (decision) rules on the basis of training examples provided by the teacher. The classification rules learned from training examples can be assumed as the background knowledge of the student. The question arises whether the background knowledge can be used to classify correctly new objects not occurring in training examples.

Classification of new objects on the basis of background knowledge previously acquired from training examples will be called dynamic learning.

The problem of dynamic learning can also be viewed as a kind of inductive generalization (inference), but we shall not consider this problem here.

Discussion on induction can be found in Barr and Fingenbaum (1981).

Orłowska (1984) discusses inductive generalization from the rough set theory point of view, however in our approach we assume a somewhere different approach.

We shall consider in this section the problem of dynamic learning in terms of concepts introduced in previous sections.

Let $S = (U, Q, V, \varrho)$ be an information system, where U is the training set, $Q = P \cup \{e\}$, P - is the set of attributes associated with student, e - is the teacher attribute providing classification e^* of training examples, and let α be the classification algorithm resulting from the set U of training examples.

Assume that the student has to classify a new object x (not belonging to the training set U) using the classification algorithm α. Let t_x be a P-elementary term describing object x.

If in the classification algorithm α there is a classification rule $t_i = t_i'$ such that $t_i = t_x$, the student will assign object x to the set $\delta(t_i')$. (one class if the algorithm is deterministic; union of some classes if the algorithm is non-deterministic).

If there is no such rule the student is unable to classify the new object by means of algorithm α.

We assume that the teacher also classifies the new objects according to his knowledge. If both decisions, that of the student and that of the teacher, agree, the student classification is <u>correct</u> - otherwise the classification is <u>incorrect</u>.

Thus by adding a new object x, we face the following possibilities:

1) the student classification of x is correct,

2) the student classification of x is incorrect,

3) the student is unable to classify the new object x.

In order to show how the background knowledge influences the correctness of student decisions we have to investigate how the accuracy and quality of learning changes in all above mentioned three situations.

Since adding a new object x to the set U results in a new information system S', our task is to compare the co-efficients β, ϱ and β', ϱ' for S and S', respectively, in the three above mentioned situations (correct, incorrect, classification impossible).

The accuracy co-efficient for these three situations is given below:

a) Correct classification:

$$\beta_P'(e^*) = \frac{\text{card } \underline{P}(e^*) + 1}{\text{card } \overline{P}(e^*) + k_P(x)}$$

where

$\underline{P}(e^*)(\overline{P}(e^*))$ is the lower (upper) approximation of the classification $e^* = \{X_1, X_2, \ldots, X_n\}$ in S,

$$\text{card } \underline{P}(e^*) = \sum_{i=1}^{n} \text{card } \underline{P}X_i,$$

$$\text{card } \overline{P}(e^*) = \sum_{i=1}^{n} \text{card } \overline{P}x_i,$$

$k_P(x)$ - arity of x in S.

b) Incorrect classification:

$$\beta_P'(e^*) = \frac{\text{card } \underline{P}(e^*) - \text{card}[x]}{\text{card } \bar{P}(e^*) + k_P(x)}$$

where $[x]$ is the set of all objects in $U \cup \{x\}$ having the same description as x.

c) Classification impossible:

$$\beta_P'(e^*) = \frac{\text{card } \underline{P}(e^*)}{\text{card } \bar{P}(e^*) + 1}$$

The quality co-efficient has the value:

d) Correct classification:

$$\gamma_P'(e^*) = \frac{\text{card } \underline{P}(e^*) + 1}{\text{card } U + 1}$$

e) Incorrect classification:

$$\gamma_P'(e^*) = \frac{\text{card } \underline{P}(e^*) - \text{card}[x]}{\text{card } U + 1}$$

f) Classification impossible:

$$\gamma_P'(e^*) = \frac{\text{card } \underline{P}(e^*)}{\text{card } U + 1}$$

Let us discuss briefly the above formulas. Consider first the deterministic case, when the classification algorithm is deterministic. In this case both co-efficients β' and γ' are the same and have the form:

g) Correct classification:

$$\beta_P'(e^*) = \gamma_P'(e^*) = \frac{\text{card } U + 1}{\text{card } U + 1} = \frac{\text{card } U}{\text{card } U}$$

$$= \beta_P(e^*) = \gamma_P(e^*) = 1$$

h) Incorrect classification:

$$\beta_P'(e^*) = \gamma_P'(e^*) = \frac{\text{card } U - \text{card}[x]}{\text{card } U + 1}$$

i) Classification impossible:

$$\beta_P'(e^*) = \gamma_P'(e^*) = \frac{\text{card } U}{\text{card } U + 1}$$

This is to say that:

Correct classification does not change the accuracy and quality of learning.

Incorrect classification decreases the accuracy and quality of learning "essentially".

Impossibility of classification decreases the accuracy and quality of learning "slightly".

Informally this can be explained as follows:

If the training set U has all possible types of objects, adding a new object does not improve the background knowledge and this knowledge is sufficient to learn properly how to classify any new object.

If the set of attributes P is not large enough then the student may face a situation in which the new object x has the same description as another object y in the training set U, but x and y belong to two different classes according to the teacher knowledge. This is to say that these two objects are different in the teacher opinion; while the student is unable to distinguish them by checking their properties (attributes from the set P), which leads to an incorrect classification. Thus the background knowledge does not suffice to classify a new object correctly in such a case.

If the set of examples U is not large enough it may happen that the new object x has a completely new description in terms of attributes from P, and this description does not match any description of objects in the training set U. So the student is unable to classify this object by means of the classification algorithm. Also in this case the background knowledge does not suffice to classify the new object correctly.

The above discussion could be more precise if we used the concept of a sample of a set (see Pawlak (1982)), but this lies outside the scope of the article.

Let us now discuss the case when the classification algorithm is non-deterministic.

The accuracy of learning in this case is the following:

j) Correct classification:

$$\beta_P'(e^*) = \frac{\text{card } \underline{P}(e^*) + 1}{\text{card } \overline{P}(e^*) + 1} > \frac{\text{card } \underline{P}(e^*)}{\text{card } \overline{P}(e^*)} = \beta_P(e^*)$$

k) Incorrect classification:

$$\beta_P'(e^*) = \frac{\text{card } \underline{P}(e^*) - \text{card } [x]}{\text{card } \overline{P}(e^*) + k_P(x)}$$

l) Classification impossible:

$$\beta_P'(e^*) = \frac{\text{card } \underline{P}(e^*)}{\text{card } \underline{P}(e^*) + 1}$$

It can easily be seen that in the case of correct classification the accuracy increases with new experience (new objects). This means that the background knowledge can be improved by proper new examples unlike in the previous case of deterministic algorithm.

The case of incorrect classification by non-deterministic classification algorithm needs some more explanation. Incorrect classification means that the student is unable to assign the new object to any single class although he is able to point out several classes to which the object may belong. This is, however, according to our definition, not a proper classification. Therefore the accuracy is decreasing in this case.

The last case is obvious.

Similar discussion can be provided for the quality coefficient and is left to the reader.

To sum up, if the student background knowledge is in a certain sense complete (the classification algorithm is deterministic) it provides the highest accuracy and quality, and it is imposeible to increase the classification skills of the student by new examples. If the background knowledge is incomplete (the classification algorithm is non-deterministic) the classification skills of the student can be improved

by properly chosen new training examples.

The presented approach can also be used when the student is faced not with one new object, but a sequence of new objects - which leads to the concept of learning process.

This problem however requires some auxiliary notions and will be discussed on another occasion.

3.3. Decision tables

The concept of decision table (see Pollack, Hicks and Harrison (1971)) can be precisely formulated in terms of the proposed approach (see Pawlak, 1985a and Pawlak, 1985 b).

We shall identify a decision table with an information system $S = (U, Q, V, \varrho)$ assuming that $Q = P \cup R$ and $P \cap R = \emptyset$, where P will be called conditions attributes and R actions attributes. Attributes from R define some actions which are performed provided some conditions pointed out by conditions attributes are met.

Example 3.3.1.

Suppose VALVE is an action attribute and V_{VALVE} = on, off and let TEMPERATURE be a condition attribute with the following set of values $V_{TEMPERATURE}$ = high, normal, low. The decision table may have the form:

number	TEMPERATURE	VALVE
1	normal	off
2	high	off
3	low	on

Tab. 13 .

The decision table shows that if the temperature (of water) is normal or high the valve controlling the fuel supply should be closed, if the temperature is low, the valve should be open.

■

Note that the objects in the information system representing a decision table are situations (states) of a certain device or environment.

The properties of information systems and description languages can be interpreted in terms of decision tables. For example:

a) a decision table $S = (U, P \cup R, V, \varrho)$ is <u>deterministic</u> iff $P \varrho R$; otherwise the decision is <u>non-deterministic</u>;

b) a decision table $S = (U, P \cup R, V, \varrho)$ is <u>P-optimal</u> (<u>R-optimal</u>) iff the system $S/P(S/R)$ is reduced;

c) a decision table $S = (U, P \cup R, V, \varrho)$ is <u>selective</u> iff $\widetilde{P \cup R}$ is the identity relation.

The deterministic decision table describes uniquely actions to be performed when some conditions are satisfied; In the case of a non-deterministic table actions are not uniquely determined by the conditions. Instead a subset of actions is defined which are possible to perform under

circumstances determined by the conditions. In some applications non-determinism in decision tables can be interpreted as a kind of inconsistency, i.e., the actions determined by a decision table are ill-defined.

The properties given in the previous sections, concerning dependence of attributes, can be used here to check whether a given decision table is deterministic or non-deterministic (i.e., well-defined or ill-defined).

The notion of a reduct can be used in decision tables to optimize the table, i.e., to find the smallest set of conditions attributes or/and actions attributes.

The description language of an information system can be interpreted as a language in which decision rules and decision algorithms resulting from a given decision table are formulated. Thus the concept of a decision rule and decision algorithm formulated in section 2.10 can be directly employed to find the decision algorithm from the decision table and to transform the decision algorithm in to a practically convenient form.

It seems that this application is obvious and does not require further comments.

The proposed approach has found application in the cement kiln process and has confirmed practically the theoretical considerations (see Mrózek (1984)).

3.4. Expert Systems

Rule driven expert systems (see Barr and Feingenbaum (1981)) can also be very easily presented as a special case of the problems discussed in previous sections.

The problem of rule-driven expert systems is in a certain sense opposite to the decision tables analysis.

In the case of decision tables we are given a decision table and we look for a decision algorithm which implements the decision table.

In a rule-driven expert system we are given a decision algorithm of the form: if...then..., which means that if some conditions are satisfied, one has to perform the actions pointed out by the algorithm.

Using the concept of information system and the language of information system we can interpret the set of rules as a decision algorithm and prove its consistency (whether it is deterministic or non-deterministic) and simplify the algorithm by reducing the set of conditions and actions attributes. Both these tasks are impossible to solve within existing theories (see Barr and Feingenbaum (1981)). Because the problem is obvious in view of the properties of information systems given in section 2 we shall refrain ourselves from further comments.

Acknowledgement

The author's thanks are due to prof. J. Bańkowski and Dr A. Mrózek for helpful discussions and critical remarks.

References

Banerji, R.B. (1980). Artificial Intelligence: a Theoretical Perspective. Elsevier North Holland, New York.

Barr, A. and Feingenbaum, E. (1981). The Handbook of Artificial Intelligence, Vol. 1-3, Harris Tech. Press, Stanford, California.

Marek, W. and Pawlak, Z., (1976). Information systems: mathematical foundations. Theoretical Computer Science. No 1. 321-354.

Michalski, R., Carbonell, J. and Mitchell, T. (1983). Machine Learning Tioga Publishing Company, Palo Alto.

Mrózek, A. (1984). Information Systems and Control Algorithms Tech. Note (15 p.).

Orłowska, E. (1984). Semantical Analysis of Inductive Reasoning, Tech. Note (21 p.).

Orłowska, E. (1980) Dependences of Attributes in Pawlak's Information Systems. Fundamenta Informaticae VI.3-4, 247-256.

Orłowska, E. and Pawlak, Z. (1984). Logical Foundations of Knowledge Representation, ICS PAS Reports No 537, 1-108.

Pawlak, Z. (1984). Rough Classification. Int. J. Man-Machine Studies. 20, 469-483.

Pawlak, Z. (1981). Information Systems, Theoretical Foundations. Information Systems, 6(3), 205-218.

Pawlak, Z. (1982). Rough sets. International Journal of Information and Computer Sciences, 11(5), 341-356.

Pawlak, Z. (1985a). Decision Tables and Decision Algorithms. Bull. Polish Acad. Sci. (to appear).

Pawlak, Z. (1985b). Rough Sets and Decision Tables. Lecture Notes, Springer Verlag (to appear).

Pollack, S., Hicks, H. and Harrison, W. (1971). Decision Tables: Theory and Practice. Wiley and Sons, Inc. New York, London.

Tou, J.T. (1980). Knowledge Engineering, International Journal of Computer and Information Sciences, 9, 275-285.

Contents