ON LEARNING - A ROUGH SET APPROACH

Z. Pawlak
Institute of Computer Science
Polish Academy of Sciences
P.O. Box 22
00-901 Warsaw, PKiN, Poland

1. Introduction

We propose in this article a new approach to learning and inductive inference. We advocate the use of the rough set 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 learning algorithms.

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

2. Information System

In this section we introduce basic concepts needed to define precisely the idea of learning from examples, delivered by a teacher, an expert, environment etc.

2.1. Definition of Information System

We shall start our considerations from the notion of an information system.

By an information system we understand 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 domain of 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 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 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 $\tilde{P}$ 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 ($x\tilde{P}y$) iff $\varrho_x(q) = \varrho_y(q)$ for every $q \in P$.

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

The equivalence classes of the relation $\tilde{P}$ 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 $\tilde{P}$ are classes (blocks) of the classification P^*.

Certainly $P^* = U/\tilde{P}$. We shall use the notation $U/\tilde{P}$ 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 $\tilde{P}$ and $\tilde{R}$ are equivalence relations, then $\tilde{T} = \tilde{P} \cap \tilde{R}$ is called an intersection of $\tilde{P}$ and $\tilde{R}$, and is defined as follows:

$$x \tilde{T} y \text{ iff } x \tilde{P} y \text{ and } x \tilde{R} y.$$

It can easily be seen that

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

Any finite union of P-elementary sets will be called a P-definable set in S. An empty set is P-definable for every $P \subseteq Q$ in every S.

An information system S is selective iff all atoms in S are one-element sets, i.e. $\tilde{Q}$ 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 P-representation of S is an information system

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

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

$$\varrho_P : U/\tilde{P} \to V_P$$

is the information function such that

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

for every $x \in U$, $q \in P$. ($[x]_P$ - denotes an equivalence class of the relation 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 for P-elementary sets. Of course, every P-representation of any information system $S = (U, Q, V, \varrho)$, $P \subseteq Q$, is selective.

Example 2.3.1

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

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

Tab. 2

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

Tab. 3

$U/\{\tilde{p,q,r}\}$	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:

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

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

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) $\underline{P}\emptyset = \overline{P}\emptyset = \emptyset; \quad \underline{P}U = \overline{P}U = U$

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) $\overline{P}(X \cap Y) \subseteq \overline{P}X \setminus \overline{P}Y$

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

8) $\overline{P}(-X) = -P(X)$.

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;
$Bn_P X$ 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:

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

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 $\underline{P}X = \emptyset$ and $\overline{P}X \neq U$.

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

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 $\mathfrak{X} = \{X_1, X_2, \ldots, X_n\}$, where $X_i \subseteq U$ $i \geq 2$, 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 $i, j \leq n$, $i \neq j$ 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.

$Bn_P(\mathfrak{X}) = \sum_{i=1}^{n} Bn_P X_i$ 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

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

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

$\beta_P(\mathfrak{X})$ 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:

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

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

Obviously $\beta_P(\mathfrak{X}) \leq \gamma_P(\mathfrak{X})$ and $\beta_P(\mathfrak{X}) = \gamma_P(\mathfrak{X})$ 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, $P \underset{S}{\Rightarrow} R$ (or in short $P \to R$) 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) $P \underset{S}{\Rightarrow} R$

2) $\widetilde{P \cup R} = \tilde{P}$

3) R^* is P-definable in S

4) $\underline{P}(R^*) = \tilde{P}(R^*)$

5) $\gamma_P(R^*) = \beta_P(R^*) = 1$.

Fact 2.8.2.

1) If $R \subseteq P$, then $P \underset{S}{\rightarrow} R$.

2) If $P \underset{S}{\rightarrow} R$ and $P' \supseteq P$, then $P' \underset{S}{\rightarrow} R$.

3) If $P \underset{S}{\rightarrow} R$ and $R' \subseteq R$, then $P \underset{S}{\rightarrow} R'$.

4) If $P \underset{S}{\rightarrow} R$, then $P \xrightarrow[S_{PUR}]{} R$.

A simple algorithm for checking whether $P \underset{S}{\rightarrow} R$ or not results from properties 1) (Fact 2.8.1) and 4) (Fact 2.8.2).

Note that 2) (Fact 2.8.1) and 4) (Fact 2.8.2) yield the property $P \underset{S}{\rightarrow} R$ iff S_{PUR}/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_{PUR} 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 $P \underset{S}{\rightarrow} R$ holds, otherwise the dependence $P \underset{S}{\rightarrow} R$ 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 $\widetilde{\{p,q,r\}}$ 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., $\widetilde{\{p,q\}} \subset \tilde{r}$, which is equivalent to $\widetilde{\{p,q\} \cup \{r\}} = \widetilde{\{p,q\}}$.

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 $\gamma_P(R^*)$ 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 $P' \subset P$, $\tilde{P}' \supset \tilde{P}$.

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

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

It can easily be shown that the following properties hold:

Fact 2.9.1.

a) If P is independent in S, then for every $p,q \in P$ neither $p \overset{}{\underset{S}{\Rightarrow}} q$ nor $q \overset{}{\underset{S}{\Rightarrow}} p$, 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 $P' \overset{}{\underset{S}{\Rightarrow}} P-P'$.

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).

3. Information Language

Our basic concept employed in this article is that of an information language, which will be used to describe learning algorithms - called here decision algorithms.

The information language consists of terms, formation and decision algorithms.

First we define the syntax of the information language and then the semantics of the language will be defined.

3.1. Syntax of an Information Language

Let us start with the definition of terms in the information language L. Terms are built up from some constants by means of Boolean operations $+$, $\cdot$, $-$. We assume that there are the following constants used to form terms: $0,1$ and Q,V are some finite sets of constants called attributes and values of attributes respectively. Moreover we assume that $V = \bigcup_{q \in Q} V_q$ and V_q is the domain of q.

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

1) Constants 0 and 1 are terms in L,

2) Any expression of the form $(q:=v)$ where $q \in Q$ and $v \in V_q$ is a term in L,

3) If t and s are terms in L, so are -t, (t+s) and (t•s) (or simple (ts)).

Example 3.1.1.

The following:

- ((q:=0) + (p:=2))(r:=1)
  (colour := green)(high = 170 cm)

are terms in some information language.

The set of formulas in the information language L is the least set satisfying the conditions:

1) Constants T (for true) and F (for false) are formulas in L,

2) If t and s are terms in L, then t=s and t ⇒ s, are formulas in L,

3) If ϕ and ψ are formulas in L, then $\sim \phi$, $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \rightarrow \psi)$ and $(\phi \Leftrightarrow \psi)$ are also formulas in L.

Example 3.1.2

The following:

$\sim$(q:=0) = (p:=2)
(q:=1)(p:=0) = (r:=2)
(q:=1) ⇒ (p:=0) + (r:=1)
((q:=0) ⇒ (r:=0)) → ((q:=1) ⇒ (p:=0))
(eyes colous := hard) ⇒ (hair colour := dark)

are formulas in some information languages.

Any formula of the form t ⇒ s will be called a decision rule in L; t is referred to as a predecessor and s the successor of the decision rule respectively.

Any finite set of decision rules in L is called a decision algorithm in L.

Example 3.1.3

The following is a decision algorithm in a certain decision language:

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

With every decision algorithm $\alpha = \{t_i \Rightarrow s_i\}_m$, $1 \leq i \leq m$ in L we associate the formula $\psi_\alpha = \bigwedge_{i=1}^{m} (t_i \Rightarrow s_i)$ called the <u>decision formula</u> of α in L.

3.2. The meaning (the semantics of terms and formulas in L

Now we shall define formally the meaning of terms and formulas of the information language, in a certain information system $S = (U, Q, V, \varrho)$. Terms are intended to mean subsets of the universe U and the meaning of formulas is truth or falsity. Of course the meaning of a certain term or formula can be different in various information systems.
In order to define the meaning of terms and formulas we shall use the meaning function $\sigma_S : Ter \cup For \to \mathcal{P}(U) \cup \{T,F\}$, where Ter and For denote the set of all terms and formulas of a information language respectively.

The meaning function σ_S for terms in defined as follows (we omit the subscript S if S is understood):

1) $\sigma(0) = \emptyset$; $\sigma(1) = U$
2) $\sigma(q:=v) = \{x \in U : \varrho(x,q) = v\}$
3) $\sigma(-t) = U - \sigma(t)$
   $\sigma(t+s) = \sigma(t) \cup \sigma(s)$
   $\sigma(ts) = \sigma(t) \cap \sigma(s)$.

<u>Example 3.2.1</u>

The meaning of the term $(q:=0)(p:=1)$ in the information system shown in Table 1 is the subset $\{x_1, x_5, x_9\}$.

The meaning of formulas is defined thus:

1) $\sigma(T) = T$; $\sigma(F) = F$,
2) $\sigma(t=s) = \begin{cases} T, & \text{if } \sigma(t) = \sigma(s) \\ F, & \text{otherwise} \end{cases}$
3) $\sigma(t \Rightarrow s) = \begin{cases} T, & \text{if } \sigma(t) \subseteq \sigma(s) \\ F, & \text{otherwise} \end{cases}$

4) $\sigma(\sim\phi) = \begin{cases} T, & \text{if } \sigma(\phi) = F \\ F, & \text{if } \sigma(\phi) = T \end{cases}$

5) $\sigma(\phi \vee \psi) = \sigma(\phi) \vee \sigma(\psi)$

6) $\sigma(\phi \wedge \psi) = \sigma(\phi) \wedge \sigma(\psi)$

7) $\sigma(\phi \rightarrow \psi) = \sigma(\sim\phi) \vee \sigma(\psi)$

8) $\sigma(\phi \Leftrightarrow \psi) = \sigma(\phi \rightarrow \psi) \wedge \sigma(\phi \leftarrow \psi)$.

If $\sigma_S(\phi) = T$ we say that ϕ is <u>true</u> in S; if $\sigma_S(\phi) = F$ then ϕ is said to be <u>false</u> in S. If ϕ is true in S we shall write $\models_S \phi$ or simply $\models \phi$ when S is known.

If $\models_S (t=s)$ we say that terms t and s are <u>equivalent in</u> S; if $\models_S (t \Rightarrow s)$ we say that term t <u>implies</u> term s in S. If $\models_S (\phi \rightarrow \psi)$ we say that formulas ϕ and ψ are <u>equivalent</u> in S and if $\models_S (\phi \rightarrow \gamma)$ we say that formula ϕ <u>implies</u> formula γ in S.

<u>Example 3.2.2</u>

The formula (q:=0) = (p;=1) is false in the information system shown in Table 1 and the formula (p:=1) ⇒ (r:=2) is true in the system.

For the transformation of terms we shall use the axioms of Boolean algebra and the following specific axiom

$$(q:=v) = - \sum_{u \neq v, u \in V_q} (q:=u).$$

For the transformation of formulas we shall employ the axioms of propositional calculus.

A term t in L is P-<u>elementary</u> $(P \subseteq Q)$ if $t = \prod_{q \in Q} (q:=v_q)$.

A term in L is in P-<u>normal form</u> $(P \subseteq Q)$ if $t = \Sigma s$, where all s are P-elementary.

<u>Example 3.2.3</u>

The terms

(p:=1)(q:=0)
(p:=2)(q:=1)
(p:=0)(q:=1)

are $\{p,q\}$-elementary in the information system shown in Table 1, and terms

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

are $\{p,q,r\}$-elementary in the information system.

The following terms

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

are in $\{p,q\}$ and $\{p,q,r\}$ normal form respectively in the information system.

Let $S = (U,Q,V,\rho)$ be an information system, $P \subseteq Q$ subset of attributes, and L_P - an information language with the set of attributes P.

Fact 3.2.1

For every term t in L_P there exists the term s in L_P in P-normal form, such that $\models_S t=s$; s is referred to as the P-normal form of t in L_P.

Example 3.2.4

The $\{p,q\}$-normal form of the term $(p:=1)$ in the information system shown in Table 1 is the term $(p:=1)(q:=0) + (p:=1)(q:=1)$.

Subset $X \subseteq U$ is said to be P-definable in L $(P \subseteq Q)$ if there exists a term t in L_P such that $\sigma_S(t) = X$; the term t is called the P-description of X in L.

Example 3.2.5

The $\{p,q\}$-description of the subset $\{x_1,x_5,x_9\}$ of objects in the information system shown in Table 1 is the term $(p:=1)(q:=0)$.

If set $X \subseteq U$ is not P-definable in L, then the terms t and s such that $\sigma_S(t) = \underline{P}X$ and $\sigma_S(s) = \overline{P}X$ are called the P-lower and P-upper descriptions of X in L respectively.

This is to mean that some subsets of objects can be described by a given subset of attributes not exactly but with some approximation only.

3.3. Decision Rules

Our basic concept is that of a decision rule. We shall discuss this concept in same details in this section.

Let $t \Rightarrow s$ be a decision rule in L, and let P,R be two subsets of attributes $(P,R \subseteq Q)$, which occurs in t and s respectively. We shall call then $t \Rightarrow s$ a (P,R)-decision rule. If P and R are single element sets, for the sake of simplicity, we shall use the expression (p,r)-decision rule.

Let $S = (U,Q,V,\varrho)$ be an information system and $t \Rightarrow s$ a (P,R)-decision rule in L.

We say that a (P,R)-decision rule is R-deterministic in S if $\sigma_S(s) \in R^*$, i.e. $\sigma_S(s)$ is a description of some equivalence class of the equivalence relation $\tilde{R}$; otherwise the decision rule is R-nondeterministic.

We say that a (P,R)-decision rule $t \Rightarrow s$ is in PUR-normal form if t and s are in PUR-normal form.

Fact 3.3.1

A (P,R)-decision rule $t \Rightarrow s$ is true in S iff all non-empty PUR-elementary terms occuring in PUR-normal form of t occur also in the PUR-normal form of s.

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

Example 3.3.1

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) ⇒ (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) ⇒ (p:=1) is true because the normal form of the rule has the form:

(p:=1)(r:=0) ⇒ (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.

3.4. Decision Algorithms

Now we shall discuss the most important concept of our approach - the decision algorithm.

A decision algorithm α in L is said to be correct in S if $\models_S \psi_\alpha$.

Example 3.4.1

It is easy to see that the decision algorithm

(p:=1)(q:=0) ⇒ (r:=2)
(p:=0) ⇒ (r:=1)
(p:=2) + (p:=1)(q:=1) ⇒ (r:=0)

is correct in the information system shown in Table 4 whereas the decision algorithm

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

is not correct in the information system.

A decision algorithm $\mathcal{A}$ in L is P-deterministic in S ($P \subseteq Q$) if all its decision rules are P-deterministic in S; otherwise the algorithm is P-nondeterministic.

Example 3.4.2

The algorithm

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

is r-deterministic in the information system shown in Table 4, whereas the algorithm

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

is r-nondeterministic in this system.

If P and R are the sets of all attributes occuring in the predecessors and successors of the decision rules in an decision algorithm $\mathcal{A}$ then $\mathcal{A}$ will be called the (P,R)-decision algorithm.

A (P,R)-decision algorithm is total in S if for every equivalence class X of the equivalence relation $\tilde{R}$, there exists a decision rule $t_i \Rightarrow s_i$ in $\mathcal{A}$ such that $\|s_i\|_S \supseteq X_j$ otherwise the decision algorithm is partial in S.

Example 3.4.3

The algorithm

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

is total in the information system shown in Table 4, whereas the algorithm

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

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

is partial in the information system.

In order to transform decision algorithms we assume the following rules.

First we allow to replace decision rules in the algorithm by its equivalent counterparts.

Next replacement of decision rules according to the following property is allowed:

Fact 3.4.1

1) $\models_S \bigwedge_{i=1}^{m} (t_i \Rightarrow s) \rightarrow (\sum_{i=1}^{m} t_i \Rightarrow s)$

2) $\models_S ((t \Rightarrow s) \wedge ((t \Rightarrow s) \rightarrow (p \Rightarrow r))) \rightarrow (p \Rightarrow r)$

Let us notice that property 2) can be regarded as a "modus ponens" for decision rules.

To this end let us give an important property, which establishes relation between the concept of dependency of attributes and the decision algorithm.

Let $\mathcal{A}$ be a (P,R)-decision algorithm in L.

Fact 3.4.2

$$\models_S \Psi_{\mathcal{A}} \text{ iff } P \Rightarrow_S R.$$

Thus we have two methods of checking whether $P \Rightarrow_S R$ or not: we can use the semantic method using property 2.8.1, or we can use the syntactic method, proving the validity of the corresponding formula. Let us notice that the total decision algorithm is a counterpart of the dependence function; in other words, the decision algorithm is a linguistic representation of the dependence function.

4. Application to Learning

Machine learning from examples 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.

4.1. Static Learning

Suppose we are given a finite set U of objects. Elements of U are called training examples (instances) and U is called training set. Assume further that U is classified into disjoint classes $X_1, X_2, \ldots, X_n$ $(n>2)$ by a teacher (expert, environment). 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 semantic knowledge and the student, syntactical 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 intorudced 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 student's knowledge (information) about x in S. Let us extend system S by adding a new attribute e representating 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 $Q = P \cup \{e\}$, $P \cap \{e\} = \emptyset$, $V' = V \cup \{1, 2, \ldots, n\}$, $\varrho'/U \times P = \varrho$, $\varrho'_x(e) = i$ iff $x \in X_i$. $\varrho'_x(e)$, called teacher's knowledge on 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

2.8.1, e^* is P-definable iff $P \rightarrow e$, 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 $P \rightarrow e$ 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 4.1.1

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

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

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 4.1.2

Let us consider an information system shown in Tab. 10

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. 10

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. 11.

$U/\tilde{Q}$	p	q	r
z_1	1	0	2
z_2	0	1	1
z_3	2	0	0
z_4	1	0	0

Tab. 11

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. 11 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:

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

The equivalence classes of relation $\{\tilde{p,q}\}$ are following

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

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:

$$\bar{P}Y_1 = X_1$$
$$\bar{P}Y_2 = X_2$$
$$\bar{P}Y_3 = X_1 \cup X_2 \cup X_3$$

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 algorithm 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. 12.

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

Tab. 12.

The accuracy and quality of learning are:

$$\beta_P(r^*) = \frac{\text{card}(\underline{P}Y_2) + \text{card}(\underline{P}Y_3)}{\text{card}(\bar{P}Y_1) + \text{card}(\bar{P}Y_2) + \text{card}(\bar{P}Y_3)} = 9/18 = 0,5$$

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

which means that at most 75 per cent of instances can be classified correctly and at most 50 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. 13

□

The ideas presented in this section were applied to computer-supported medical diagnosis algorithms, resulting in a new simple method of medical data analysis.

4.2. 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 $\mathcal{A}$ 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 $\mathcal{A}$. Let t_x be a P-elementary term describing object x.

If in the classification algorithm $\mathcal{A}$ there is a classification rule $t_i \Rightarrow t_i^-$ 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

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_P and β_P, for S and S^-, respectively, in the three above mentioned situations (correct, incorrect, classification impossible).

Let us remark that adding a new object x to the set U changes the teacher classification too. The new object can match one of existing classes or it can form a completely new single element class.

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

a) Correct classification:

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

where

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

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

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

$k_P(x)$ - arity of x in S with respect to P;

The arity of x with respect to P in S, $k_P(x)$ is the maximal number of classes $X_{i_1}, X_{i_2}, \ldots, X_{i_m}$ in the classification e^* such that

$$x \in \bigcap_{j=1}^{m} \bar{P}X_{i_j} .$$

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 U $\{x\}$ 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 be-

long 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 classfication 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:

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

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:

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

It can easily be seen that in the case of correct classification the accuracy is not decreasing 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 impossible 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.

Acknowledgments. Thanks are due to dr. A. Skowron for helpful discussions and critical remarks.

References

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

Bar, A. and Reingenbaum, 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, 331-354.

Michalski, R., Carbonell, J. and Mitchell, T. (1980). Machine Learning Tioga Publishing Company, Palo Alto.

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