ISSN 0252-9742

BULLETIN

of the

European Association for Theoretical Computer Science

EATCS

Number 44 JUNE 1991

Decision Logic

Z. Pawlak

Institute of Computer Science
Warsaw Technical University
00 665 Warsaw, ul. Nowowiejska 15/19

1. Introduction

The concept of the rough set (cf. Pawlak (1982)) have inspired a variety of logical research (cf. Jian-Ming et al.(1990), Konikowska (1987), Nakamura et al.(1988), Orłowska (1984, 1985a,b, 1989), Pawlak (1987b), Rasiowa et al.(1985, 1986a,b), Rauszer (1985, 1986), Szczerba (1987), Vakarelov (1981, 1989) Wasilewska (1988, 1989) and others). Most of the above mentioned logical research has been directed to create deductive logical tools to deal with approximate (deductive) reasoning.

In contrast to the above line of research we propose in this chapter logic which is of inductive character (cf. Ajdukiewicz (1974)) and is intended as a tool for data analysis, i.e. our main concern is in discovering dependencies in data and data reduction, which is rather closer to statistical then deductive methods, however to this end we shall use deductive tools.

Let us explain these ideas more exactly. Our main goal is reasoning about knowledge concerning certain reality. We have assumed that knowledge is represented as a value-attribute table, called sometimes Information System (cf. Pawlak (1981)) or Knowledge Representation System (cf. Pawlak (1984)).

Representation of knowledge in tabular form, has great advantages in particular for its clarity. It turns out that the data table may be looked at as a set of propositions about the reality and consequently can be treated by means of logical tools, which will be developed in this paper. We offer two possibilities here, one based on normal form representation of formulas and the second employing indiscernibility to investigate whether some formulas are true or not. The latter approach, referring to indiscernibility, leads to simple algorithms for data reduction and analysis, and is fundamental to our philosophy.

In fact the data table can be viewed as a model for special logic, called here decision logic, which will be used to derive conclusions from data available in the knowledge representation system. We will be basically concerned in discovering dependencies in knowledge and also in knowledge reduction, and to this end we shall use syntactical tools available in the proposed logic.

One of the chief implications of the presented philosophy is that our main concern is the fundamental notion of the decision logic, the decision algorithm, which is a set of decision rules (implications). Because an algorithm is usually meant as a sequence (not set) of instructions (decision rules), thus the decision "algorithm" fails to meat the usual understanding of the notion of an algorithm, nevertheless, for the lack of a better term, we will stick to the proposed terminology.

Still one more important remark concerning the decision algorithm seems in order. Formulas can be true or false but the decision algorithm, which is a set of formulas, can not have attributes of truth or falsity. Instead consistency and inconsistency will be the basic features of decision algorithms. In other words our account, in contrast to philosophy of deduction, stress rather consistency (or inconsistency) of data then their truth (or falsity), and our main interest is not in investigation of theorem proving mechanisms in the introduced logic, but in analysis, in computational terms (decision algorithms, or condition-action rules), of how some facts are derived from data.

With the above remarks in mind we start in the next section considerations on a formal language for decision logic.

2. Language of Decision Logic

The language of decision logic (*DL*-language) we are going to define and discuss here will consists of *atomic* formulas, which are attribute-value pairs, combined by means of sentential connectives *and*, *or*, *not* etc. in a standard way, forming compound formulas.

Formally the language is defined inductively as follows.

First we start with the alphabet of the language which consists of :

a) A - the set of *attribute constants*

b) $V = \bigcup_{a \in A} V_a$ - the set of *attribute value constants*

c) Set $\{\sim, \vee, \wedge, \rightarrow, \equiv\}$ of *propositional connectives*, called respectively *negation*, *disjunction*, *conjunction*, *implication* and *equivalence* respectively.

The propositional connectives symbols may be considered as abbreviations of the logical connectives "not", "or", "and", "if ... then", "if and only if".

Let us note that the alphabet of the language contains no variables and its expressions will be built up only from the above symbols, i.e. attribute and attribute value symbols, logical connectives and some auxiliary symbols like parenthesis - which means that formulas in the *DL*-language are in fact sentences.

Moreover, we should pay attention to the fact that sets A and V_a are treated as sets of names of attributes and attribute values respectively. Hence in order to distinguish if necessary, attributes and attribute names we will use bold and italic alphabets respectively. For example **color** is the attribute and *color* is the attribute constant (name).

The case of values of attributes is quite similar. For example, if one of the values of the attribute color were red, then the corresponding attribute value constant would be *red*.

Next we define the set of formulas in our language, which are defined below.

The set of formulas of *DL*-language is the least set satisfying the following conditions:

1) Expressions of the form (a, v), or in short a_v, called *elementary* (*atomic*) *formulas*, are formulas of the

DL-language for any $a \in A$ and $v \in V_a$.

2) If ϕ and Ψ are formulas of the *DL*-language, then so are $\sim\Phi$, $(\Phi \vee \Psi)$, $(\Phi \wedge \Psi)$, $(\Phi \rightarrow \Psi)$, and $(\Phi \equiv \Psi)$.

3. Semantics of Decision Logic Language

Formulas are meant to be used as descriptions of objects of the universe. Of course some objects may have the same description, thus formulas may describe also subsets of objects obeying properties expressed by these formulas. In particular atomic formula (a, v) is interpreted as a description of all objects having value v for attribute a. Compound formulas are interpreted in the usual way.

In order to express this problem more precisely we define Tarski's style semantics of the *DL*-language employing the notions of a model and satisfiability.

By the model we will simply mean the knowledge representation system (KR-system) $S = (U, A)$, where U is a finite set called the *universe*, A is the set of attributes and each attribute $a \in A$ is a function $a: U \rightarrow V_a$, which to each object $x \in U$ assigns an attribute value $v \in V_a$. In other words a model is an attribute-value table columns of which are labelled by attributes and rows - by objects; every entry of the table corresponding to an object x and an attribute a is attribute value $a(x)$.

Thus the model S describes the meaning of symbols of predicates (a, v) in U, and if we properly interpret formulas in the model then each formula becomes a meaningful sentence, expressing properties of some objects.

This can be voiced more precisely using the the concept of satisfiability of a formula by an object, which follows next.

An object $x \in U$ satisfies a formula ϕ in $S = (U, A)$, denoted $x \models_S \phi$ or in short $x \models \phi$, if S is understood, if and only if the following conditions are satisfied:

(1) $x \models (a, v)$ iff $f(a, x) = v$

(2) $x \models \sim\phi$ iff non $x \models \phi$

(3) $x \models \phi \vee \psi$ iff $x \models \phi$ or $x \models \Psi$

(4) $x \models \phi \wedge \psi$ iff $x \models \phi$ and $x \models \Psi$

As a corollary from the above conditions we get

(5) $x \models \phi \rightarrow \psi$ iff $x \models \sim\phi \vee \psi$

(6) $x \models \phi \equiv \psi$ iff $x \models \phi \rightarrow \psi$ and $x \models \psi \rightarrow \phi$

If ϕ is a formula then the set $|\phi|_S$ defined as follows

$$|\phi|_S = \{x \in U: x \models_S \phi\}$$

will be called the *meaning* of the formula ϕ in S. Thus the meaning is a function whose arguments are formulas of the language and whose values are subsets of the set of objects of the system.

The following is an important proposition which explains the meaning of an arbitrary formula.

Proposition 1

(a) $|(a, v)|_S = \{x \in U: a(x) = v\}$

(b) $|\sim\phi|_S = -|\phi|_S$

(c) $|\phi \vee \Psi|_S = |\phi|_S \cup |\Psi|_S$

(d) $|\phi \wedge \Psi|_S = |\phi|_S \cap |\Psi|_S$

(e) $|\phi \rightarrow \Psi|_S = -|\phi|_S \cup |\Psi|_S$

(f) $|\phi \equiv \Psi|_S = |\phi|_S \cap |\Psi|_S \cup -|\phi|_S \cup -|\Psi|_S$ ■

Thus meaning of the formula ϕ is the set of all objects having the property expressed by the formula ϕ, or the meaning of the formula ϕ is the description in the *KR*-language of the set of objects $|\phi|_S$.

We need also in our logic the notion of truth.

A formula ϕ is said to be *true* in a *KR*-system S, $|=_S \phi$, if and only if $|\phi|_S = U$, i.e. the formula is satisfied by all objects of the universe in the system S.

Formulas ϕ and Ψ are equivalent in S if and only if $|\phi|_S = |\Psi|_S$.

The following proposition gives simple properties of the introduced notions.

Proposition 2

(a) $|=_S \phi$ iff $|\phi|_S = U$

(b) $|=_S \sim\phi$ iff $|\phi|_S = \emptyset$

(c) $|=_S \phi \rightarrow \psi$ iff $|\phi|_S \subseteq |\psi|_S$

(d) $|=_S \phi \equiv \psi$ iff $|\phi|_S = |\psi|_S$ ■

At the end let us stress once more that the meaning of the formula depends on the knowledge we have about the universe, i.e. on the knowledge representation system. In particular a formula may be true in one knowledge representation system but false in another one. However, there are formulas which are true independent of the actual values of attributes appearing in them, but depend only on their formal structure. They will play special role in our considerations. Note, that in order to find the meaning of such formula, one need not to be acquainted with the knowledge contained in any specific knowledge representation system because their meaning is determined by its formal structure only. Hence, if we ask whether a certain fact is true in the light of our actual knowledge (represented in a given knowledge representation system), it is sufficient to use this knowledge in an appropriate way. However, in case of formulas which are true (or not) in every possible knowledge representation system, we do not need in fact any particular knowledge but only suitable logical tools. They will be considered in the next section.

4. Deduction in Decision Logic

In this section we are going to study the deductive structure of the decision logic.To this end we have to introduce some axioms and inference rules.

Before we start a detailed discussion of this problem, let us first give some intuitive background for the proposed solution.

The language introduced in the previous section was intended to express knowledge contained in a specific knowledge representation system. However, the same language can be treated as a common language for many knowledge representation systems with different sets of objects but with identical sets of attributes and identical attribute values sets. From syntactical aspects, all the languages of such systems are identical. However, their semantics differ due to the different sets of objects and their properties are represented in specific knowledge representation systems, in which the meaning of formulas is to be defined.

In order to define our logic, we need to verify the semantic equivalence of formulas. To do this we need to end up with suitable rules for transforming formulas without changing their meanings are necessary. Of course, in theory we could also verify the semantic equivalence of formulas by computing their meaning accordingly to the definition, and comparing them in order to check whether they are identical or not. Unfortunately, such a procedure would be highly unpractical, though - due to the finiteness of the considered knowledge (tables) - it is always possible. However, this method cannot be used for verifying the equivalence of formulas in every knowledge representation system because of the necessity of computing the meanings of these formulas in an infinite number of systems. Hence suitable axioms and inference rules are needed to prove equivalence of formulas in a formal way.

Basically axioms will correspond closely to axioms of classical propositional calculus, however some specific axioms connected with the specific properties of knowledge representation systems are also needed - and the only inference rule will be modus ponens.

Thus the set of all axioms of *DL*-logic consists of all propositional tautologies and some specific axioms.

Before we list specific axioms which hold in each concrete knowledge representation system we need some auxiliary notions and denotations.

We will use the following abbreviations:

$$\Phi \wedge \sim\phi =_{df} 0 \text{ and } \Phi \vee \sim\phi =_{df} 1$$

Obviously $\models 1$ and $\models \sim 0$. Thus 0 and 1 can be assumed to denote *falsity* and *truth* respectively.

Formula of the form

$(a_1, v_1) \wedge (a_2, v_2) \wedge \ldots \wedge (a_n, v_n)$,

where $v_i \in V_{a_i}$, $P = \{a_1, a_2, \ldots, a_n\}$, and $P \subseteq A$, will be called a *P-basic formula* or in short *P-formula*. *A*-basic formulas will be called basic formulas.

Let $P \subseteq A$, ϕ be a *P*-formula and $x \in U$. If $x \models \phi$, then ϕ will be called the *P-description* of x in S. The set of all *A*-basic formulas satisfiable in the *KR*-system $S = (U, A)$ will be called *the basic knowledge* in S.

We will need also a formula $\Sigma_S(P)$, or in short $\Sigma(P)$, which is disjunction of all *P*-formulas satisfied in S; if $P = A$ then $\Sigma(A)$ will be called the *characteristic formula* of the *KR*-system $S = (U, A)$.

Thus the characteristic formula of the system represents somehow the whole knowledge contained in the *KR*-system S.

In other words each row in the table, is in our language represented by a certain *A*-basic formula, and the whole table is now represented by the set of all such formulas so that instead tables we can now use sentences to represent knowledge.

Now let us give specific axiom of *DL*-logic.

(1) $(a, v) \wedge (a, u) \equiv 0$, for any $a \in A$, $v, u \in V_a$ and

$v \neq u$

(2) $\bigvee_{v \in V_a} (a, v) \equiv 1$, for every $a \in A$

(3) $\sim(a, v) \equiv \bigvee_{u \in V_a, u \neq v} (a, u)$, for every $a \in A$

We will also need the following proposition.

Proposition 3

$\models_S \Sigma_S(P) \equiv 1$, for any $P \subseteq A$. ■

The axioms of the first group are counterparts of propositional calculus axioms. The axioms of the second group require a short comment, for they are characteristic to our notion of the knowledge representation system.

The axiom (1) follows from the assumption that each object can have exactly one value of each attribute. For example, if something is red, it cannot be either blue or green.

The second axiom (2) follows from the assumption that each attribute must take one of the values of its domain for every object in the system. For example, if the attribute in question is color, then each object must be of some color which is the value of this attribute.

The axiom (3) allows us the get rid of negation in such a way that instead of saying that an object does not posses a given property we can say that it has one of the remaining properties. For example instead of saying that something is not red we can say that it is either green, or blue or violet etc. Of course, this rule is admissible due to the finiteness assumption about the set of values of each set of attributes.

The Proposition 3 means that the knowledge contained in the knowledge representation system is the whole knowledge available at the present stage, and corresponds to so called closed word assumption (*CWA*).

Now we are ready to define basic concepts of this section.

We say that a formula ϕ is *derivable* from a set of formulas Ω, (i.e.from Σ_S) denoted $\Omega \vdash \phi$, if and only if it is derivable from axioms and formulas of Ω, by finite application of the inference rule (modus ponenes).

A formula ϕ is a *theorem* of *DL*-logic, symbolically $\vdash \phi$, if it is derivable from the axioms only.

A set of formulas Ω is *consistent* if and only if the formula $\phi \wedge -\phi$ is not derivable from Ω.

The set of theorems of *DL*-logic is identical with the set of theorems of classical propositional calculus with specific axioms (1-3), in which negation can be eliminated.

5. Normal Forms

Formulas in the *DL*-language can be presented in a special form called *normal form*, which is similar to that in classical propositional calculus.

Let $P \subseteq A$ be subset of attributes and let ϕ be a formula in *DL*-language.

We say that ϕ is in a *P-normal form* in *S*, (in short in *P-normal form*) if and only if either ϕ is 0 or ϕ is 1, or ϕ is a disjunction of non empty *P*-basic formulas in *S*. (The formula ϕ is non-empty if $|\phi|_S \neq \emptyset$).

A-normal form will be referred to as *normal form*.

The following is an important property of formulas in the *DL*-language.

Proposition 4

Let ϕ be a formula in *DL*-language and let *P* contain all attributes occurring in ϕ. Moreover assume axioms (1) -(3) and the formula $\Sigma_S(A)$. Then, there is a formula ψ in the *P*-normal form such that $\vdash \phi \equiv \psi$.■

6. Decision Rules and Decision Algorithms

In this section we are going to define two basic concept in the *DL*-language, namely that of a decision rule and a decision algorithm.

Any implication $\phi \rightarrow \psi$ will be called a *decision rule* in the *KR*-language; ϕ and ψ are referred to as the *predecessor* and the *successor* of $\phi \rightarrow \psi$ respectively.

If a decision rule $\phi \rightarrow \psi$ is true in *S* we will say that the decision rule is *consistent* in *S*, otherwise the decision rule is *inconsistent* in *S*.

If $\phi \rightarrow \psi$ is a decision rule and ϕ and ψ are *P*-basic and *Q*-basic formulas respectively, then the decision rule $\phi \rightarrow \psi$ will be called a *PQ-basic decision rule*, (in short *PQ-rule*), or *basic rule* when *PQ* is known. The sets of attributes *P* and *Q* will be refered to as condition and decison (action) attributes respectivelly.

If $\phi_1 \rightarrow \psi$, $\phi_2 \rightarrow \psi$, ... $\phi_n \rightarrow \psi$ are basic decision rules then the decision rule $\phi_1 \vee \phi_2 \vee \ldots \vee \phi_n \rightarrow \psi$ will be called *combination* of basic decision rules $\phi_1 \rightarrow \psi$, $\phi_2 \rightarrow \psi$, ... $\phi_n \rightarrow \psi$, or in short *combined* decision rule.

A *PQ*-rule $\phi \rightarrow \psi$ is *admissible* in *S* if $\phi \wedge \psi$ is satisfiable in *S*.

Throughout the remainder of this paper we will consider admissible rules only, except when the contrary is explicitly stated.

The following simple property can be employ to check whether a *PQ*-rule is true or false (consistent or inconsistent)

Proposition 5

A *PQ*-rule is true (consistent) in *S*, if and only if all $(P \cup Q)$-basic formulas which occur in the $(P \cup Q)$-normal form of the predecessor of the rule, and occur also in the $(P \cup Q)$-normal form of the successor of the rule; otherwise the rule is false (inconsistent) in *S*.■

Any finite set of decision rules in a *DL*-language, is referred to as a *decision algorithm* in the *Dl*-language.

We recall, as already mentioned in the Introduction, that by an algorithm we mean a *set* of instructions (decision rules), and not as usually - a *sequence* of instructions. Thus our conception of algorithm differs from the existing one, and can be understood as generalization of the latter.

Now we are going to define the the basic concept of this section.

Any finite set of basic decision rules will be called a *basic decision algorithm*.

If all decision rules in a basic decision algorithm are *PQ*-decision rules, then the algorithm is said to be *PQ-decision algorithm*, or in short *PQ-algorithm*, and will be denoted by (P,Q).

A *PQ*-algorithm is *admissible* in S, if the algorithm is the set of all *PQ*-rules admissible in S.

A *PQ*-algorithm is *complete* in S, if for every $x \in U$ there exists a *PQ*-decision rule $\phi \rightarrow \psi$ in the algorithm such that $x \models \phi \wedge \psi$ in S; otherwise the algorithm is *incomplete* in S.

In what follows we shall consider admissible and complete *PQ*-algorithms only, if not stated otherwise.

The *PQ*-algorithm is *consistent* in S, if and only if all its decision rules are consistent (true) in S; otherwise the algorithm is *inconsistent in S*.

Sometimes consistency (inconsistency) may be interpreted as determinism (indeterminism), however we shall stick to the concept of consistency (inconsistency) instead of determinism (nondeterminism), if not stated otherwise.

Thus when we are given a *KR*-system, then any two arbitrary, nonempty subsets of attributes P, Q in the system, determine uniquely a *PQ*-decision algorithm. Note that the *KR*-system with distinguished condition and decision attributes may be regarded as a decision table. (cf. Pawlak

(1985, 1986, 1987a)).

7. Truth and Indiscernibility

In order to check whether a decision algorithm is consistent or not we have to check whether all its decision rules are true or not. To this end we could employ Proposition 5, however the following propositions gives a much simpler method to solve this problem which will be used in what follows.

Proposition 6

A *PQ*-decision rule $\phi \rightarrow \psi$ in a *PQ*-decision algorithm is consistent (true) in S, if and only if for any *PQ*-decision rule $\phi' \rightarrow \psi'$ in (P,Q), $\phi = \phi'$ implies $\psi = \psi'$.■

Note that in this proposition order of terms is important, since we require equality of expresions.

Let us also remark tha in order to check whether a decision rule $\phi \rightarrow \psi$ is true or not we have to show that the predecessor of the rule (the formula ϕ) discerns the decision class ψ from the remaining decision classes of the decision algorithm in question. Thus the concept of truth is somehow replaced by the concept of indiscernibility.

We will often treat decision tables as a convenient way of representation of decision algorithms, for this form is more compact and easy to follow, then the *DL*-language. Note however that formally decision algorithms and decision tables are different concepts.

8. Depenedency of Attributes

Now we are ready to define the most essential concept of our approach - the dependency of attributes.

We will say that the set of attributes Q *depends totally*, (or in short *depends*) on the set of attributes P in S, if there exists a consistent *PQ*-algorithm in S. If Q depends on P in S we will write $P \Rightarrow_S Q$ or in short $P \Rightarrow Q$.

We can also define partial dependency of attributes.

We say that the set of attributes Q *depends partially* on the set of attributes P in S if there exists only an inconsistent PQ-algorithm in S.

Similarly as before we are able to define the degree of dependency between attributes.

Let (P,Q) be a PQ-algorithm in S. By a *positive region* of the algorithm (P,Q), denoted $POS(P,Q)$ we mean the set of all consistent (true) PQ-rules in the algorithm.

In other words the positive region of the decision algorithm (P,Q) is the consistent part (possibly empty) of the inconsistent algorithm.

Obviously a PQ-algorithm is inconsistent if and only if $POS\ (P,Q) \neq (P,Q)$ or what is the same $card\ (POS\ (P,Q)) \neq card\ (P,Q)$.

With every PQ-decision algorithm we can associate a number $k = card\ (POS\ (P,Q))\ /\ card\ (P,Q)$, called the *degree of consistency* of the algorithm, or in short the *degree* of the algorithm, and we will say that the PQ-algorithm has the degree (of consistency) k.

Obviously $0 \leq k \leq 1$. If a PQ-algorithm has degree k we can say that the set of attributes Q *depends in degree* k on the set of attributes P, and we will write $P \Rightarrow_k Q$.

Naturally the algorithm is consistent if and only if $k = 1$, otherwise, i.e. if $k \neq 1$, the algorithm is inconsistent.

Let us note that in the consistent algorithm all decisions are uniquely determined by conditions in the decision algorithm, which is not the case in inconsistent algorithm. In other words all decisions in a consistent algorithm are discernible by means of conditions available in the decision algorithm.

9. Reduction of Consistent Algorithms

The problem we are going to consider in this section, concerns simplification of decision algorithms, more exactly we will investigate whether all condition attributes are necessary to make decisions. In this section we will discuss the case of a consistent algorithm.

Let (P,Q) be a consistent algorithm, and $a \in P$.

We will say that the attribute a is *dispensable* in the (P,Q)-algorithm if and only if the algorithm $((P-\{a\}),Q)$ is consistent; otherwise the attribute a is *indispensable* in the algorithm (P,Q).

If all attributes $a \in P$ are indispensable in the algorithm (P,Q), then the algorithm (P,Q) will be called *independent*.

The subset of attributes $R \subseteq P$ will be called a *reduct* of P in the algorithm (P,Q), if the algorithm (R,Q) is independent and consistent.

If R is a reduct of P in the algorithm (P,Q), then the algorithm (R,Q) is said to be a *reduct* of the algorithm (P,Q).

The set of all indispensable attributes in an algorithm (P,Q) will be called the *core* of the algorithm (P,Q), and will be denoted by $CORE\ (P,Q)$.

One can prove the following imporatnt theorem.

Proposition 7

$$CORE\ (P,Q) = \bigcap RED\ (P,Q)$$

where $RED\ (P,Q)$ is the set of all reducts of (P,Q). ■

If all rules in a basic decision algorithm are reduced, then the algorithm is said to be *reduced*.

The following example will illustrate the above ideas.

10. Reduction of Inconsistent Algorithms

In the case of inconsistent PQ-algorithm in S the reduction and normalization goes in a similar way.

Let (P,Q) be a inconsistent algorithm, and $a \in P$.

An attribute a is *dispensable* in P,Q-algorithm, if $POS\ (P,Q) = POS\ ((P - \{a\}),Q)$; otherwise the attribute a is *indispensable* in (P,Q).

The algorithm (P,Q) is *independent* if all $a \in P$ are indispensable in (P,Q).

The set of attributes $R \subseteq P$ will be called a *reduct* of (P,Q), if (R,Q) is independent and $POS\ (P,Q) = POS\ (R,Q)$.

As before the set of all indispensable attributes in (P, Q) will be called the *core* of (P,Q), and will be denoted by $CORE\ (P,Q)$. In this case the Proposition 7 is also valid.

Thus the case of the consistent algorithm is a special case of the inconsistent one.

11. Reduction of Decision Rules

The purpose of this section is to show how the decision logic can be used to further simplification of decision algorithms by elimination of unnecessary conditions in each decision rule of a decision algorithm separately, in contrast to reduction performed on all decision rules simultaneously, as defined in the previous sections. Before we give the necessary definitions, let us first introduce auxiliary denotation. If ϕ is P-basic formula and $Q \subseteq P$, then by ϕ/Q we mean the Q-basic formula obtained from the formula ϕ by removing from ϕ all elementary formulas (a, v_a) such that $a \in P - Q$.

Let $\phi \rightarrow \psi$ be a PQ-rule, and let $a \in P$. We will say that the attribute a is *dispensable* in the rule $\phi \rightarrow \psi$ if and only if

$$\models_S \phi \rightarrow \psi \text{ implies } \models_S \phi/(P-\{a\}) \rightarrow \psi$$

otherwise the attribute a is *indispensable* in $\phi \rightarrow \psi$.

If all attributes $a \in P$ are indispensable in $\phi \rightarrow \psi$ then $\phi \rightarrow \psi$ will be called *independent*.

The subset of attributes $R \subseteq P$ will be called a *reduct* of PQ-rule $\phi \rightarrow \psi$, if $\phi \rightarrow \psi$ is independent and $\models_S \phi \rightarrow \psi$ implies $\models_S \phi/R \rightarrow \psi$.

If R is a reduct of the PQ-rule $\phi \rightarrow \psi$, then $\phi/R \rightarrow \psi$ is said to be *reduced*.

The set of all indispensable attributes in $\phi \rightarrow \psi$ will be called the *core* of $\phi \rightarrow \psi$, and will be denoted by $CORE\ (\phi \rightarrow \psi)$.

One can easily verify that the following theorem is true.

Proposition 8

$$CORE\ (P \rightarrow Q) = \bigcap RED\ (P \rightarrow Q),$$

where $RED\ (P \rightarrow Q)$ is the set of all reducts of $(P \rightarrow Q)$. ■

12. Minimization of Decision Algorithms

In this section we will consider whether all decision rules are necessary in a decision algorithm, or more exactly we aim at elimination of superfluous decision rules associated with the same decision class. It is obvious that some decision rules can be dropped without disturbing the decision making process, since some other rules can overtake the job of the eliminated rules. This is equivalent to the problem of elimination of superfluous sets in union of certain sets, discussed in Chapter 3.4., which become more evident as the study progress. Before we state the problem more precisely some auxiliary notions are needed.

Let $\mathcal{A}$ be a basic algorithm, and let $S = (U,A)$ be a KR-system. The set of all basic rules in $\mathcal{A}$ having the same successor ψ will be denoted $\mathcal{A}_\psi$, and $\mathcal{P}_\psi$ is the set of all predecessors of decision rules belonging to $\mathcal{A}_\psi$

A basic decision rule $\phi \rightarrow \psi$ in $\mathcal{A}$ is *dispensable* in $\mathcal{A}$, if $\models_S \vee \mathcal{P}_\psi \equiv \vee \{\mathcal{P}_\psi - \{\phi\}\}$, where $\vee\mathcal{P}_\psi$ denotes disjunction of all formulas in $\mathcal{P}_\psi$; otherwise the rule is *indispensable* in $\mathcal{A}$. If all decision rules in $\mathcal{A}_\psi$ are indispensable then the set of rules $\mathcal{A}_\psi$ is called *independent*.

A subset $\mathcal{A}'_\psi$ of decision rules of $\mathcal{A}_\psi$ is a *reduct* of $\mathcal{A}_\psi$ if all decision rules in $\mathcal{A}'_\psi$ are independent and $\models_S \vee \mathcal{P}_\psi \equiv \vee \mathcal{P}'_\psi$.

A set of decision rules $\mathcal{A}_{\psi}$ is *reduced*, if reduct of $\mathcal{A}_{\psi}$ is $\mathcal{A}_{\psi}$ it shelf.

Now we are ready to give the basic definition of this section.

A basic algorithm $\mathcal{A}$ is *minimal*, if every decision rule in $\mathcal{A}$ is reduced and for every decision rule $\phi \rightarrow \psi$ in $\mathcal{A}$, $\mathcal{A}_{\psi}$ is reduced.

Thus in order to simplify a *PQ*-algorithm, we must first reduce the set of attributes, i.e. we present the algorithm in a normal form (note that many normal forms are possible in general). The next step consists in the reduction of the algorithm, i.e. simplifying the decision rules. The least step removes all superfluous decision rules from the algorithm.

The example which follows will depict the above defined concepts.

Example

Supouse we are given the following *KR*-system

U	*a*	*b*	*c*	*d*	*e*
1	1	0	0	1	1
2	1	0	0	0	1
3	0	0	0	0	0
4	1	1	0	1	0
5	1	1	0	2	2
6	2	2	0	2	2
7	2	2	2	2	2

Table 1

and assume that $P = \{a, b, c, d\}$ and $Q = \{e\}$ are condition and decision attributes respectively.

It is easy to compute that the only *e*-dispensable condition attribute is *c*. Thus Table 1 can be simplified as shown in Table 2 below.

U	*a*	*b*	*d*	*e*
1	1	0	1	1
2	1	0	0	1
3	0	0	0	0
4	1	1	1	0
5	1	1	2	2
6	2	2	2	2
7	2	2	2	2

Table 2

In the next step we have to reduce the superfluous values of attributes, i.e. reduce all decision rules in the algorithm. To this end we have first computed core values of attributes, and the result is presented in Table 3.

U	*a*	*b*	*d*	*e*
1	–	0	–	1
2	1	–	–	1
3	0	–	–	0
4	–	1	1	0
5	–	–	2	2
6	–	–	–	2
7	–	–	–	2

Table 3

In the table below we have listed all value reducts

U	a	b	d	e
1	1	0	x	1
1'	x	0	1	1
2	1	0	x	1
2'	1	x	0	1
3	0	x	x	0
4	x	1	1	0
5	x	x	2	2
6	2	1	x	2
6'	2	x	2	2
6''	x	1	2	2
7	2	2	x	2
7'	2	x	2	2
7''	x	2	2	2

Table 4

As we can see from the table in row 1 we have two reducts of condition attributes - $a_1\ b_0$ and $b_0\ d_1$. Similarly for the row number 2 we have also two reducts - $a_1\ b_0$ and $a_1\ d_0$. There are two minimal sets of decision rules for decision class 1, namely

1) $a_1 b_0 \rightarrow e_1$
2) $b_0 d_1 \rightarrow e_1$
   $a_1 d_0 \rightarrow e_1$

or

$$b_0 d_1 \vee a_1 d_0 \rightarrow e_1$$

For decision class 0 we have one minimal set of decision rules

$$a_0 \rightarrow e_0$$
$$b_1 d_1 \rightarrow e_0$$

or

$$a_0 \vee b_1 d_1 \rightarrow e_0$$

For the decision class 2 we have also one minimal decision rule

$$d_2 \rightarrow e_2$$

Finally we get two minimal decision algorithms

$$a_1 b_0 \rightarrow e_1$$
$$a_0 \rightarrow e_0$$
$$b_1 d_1 \rightarrow e_0$$
$$d_2 \rightarrow e_2$$

and

$$b_0 d_1 \rightarrow e_1$$
$$a_1 d_0 \rightarrow e_1$$
$$a_0 \rightarrow e_0$$
$$b_1 d_1 \rightarrow e_0$$
$$d_2 \rightarrow e_2$$

The combined form of these algorithms are

$$a_1 b_0 \rightarrow e_1$$
$$a_0 \vee b_1 d_1 \rightarrow e_0$$
$$d_2 \rightarrow e_2$$

and

$b_0d_1 \vee a_1d_0 \rightarrow e_1$

$a_0 \vee b_1d_1 \rightarrow e_0$

$d_2 \rightarrow e_2$ ■

References

Ajdukiewicz, K. (1974). Pragmatic Logic. Translation from the Polish by Olgierd Wojtasiewicz, *Reidel, Polish Scientific Publishers*, Dordrecht, Warsaw.

Jian-Ming Gao and Nakamura, A. (1990). A Semantic Decision Method for the Logic of Indiscernibility Relation. *Fundamenta Informaticae*, (to appear).

Konikowska, B. (1987). A Formal Language for Reasoning about Indiscernibility. *Bull. Polish Acad. Sci. Math.*, 35, pp. 239-250.

Krynicki, M. (1989). Lineary Ordered Quantifiers. *Bull. Polish. Acad. Sci. Math.*, 37, 6, pp. 295-303.

Krynicki, M. (1990). Quantifiers Determined by Partial Order. *Zeitschrift feur Grundlagen der Mathematik und Logic*. 36, pp.79-86.

Krynicki, M. (1990). A note on Rough Concept Logic. *Fundamenta Informaticae*. XIII, pp. 227-235.

Krynicki, M. and Tuschnik, H. P. (1990). An Aximatisation of the Logic with Rough Quantifiers. *Journal of Symbolic Logic* (to appear).

Nakamura, A. and Jian-Ming Gao (1988). Modal Logic for Similarity-Based Data Analysis. *Hiroshima University Technical Report*, C-26.

Orłowska, E. and Pawlak, Z., (1984). Logical Foundations of Knowledge Representation. *Institute of Computer Science, Polish Academy of Sciences Reports*. 537, pp. 1-106.

Orłowska, E. (1985a). Logic of Indiscernibility Relation. *Bull. Polish Acad. Sci. Math.*, pp. 475-485.

Orłowska, E. (1985b). Logic Approach to Information Systems. *Fundamenta Informaticae*. 8, pp. 359-378.

Orłowska, E. (1989). Logic for Reasoning about Knowledge. *Zeitschr. f. Math. Logik und Grundlagen d. Math.*, 35, pp. 559-572.

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

Pawlak, Z. (1982). Rough Sets. *International Journal of Computer and Information Sciences*, 11, pp. 341-356.

Pawlak, Z. (1984). On Discernibility of Objects in Knowledge Representation Systems, *Bull. Pol. Acad. Sci. Tech.*, 32, pp. 613-615.

Pawlak, Z, (1985). Decision Tables and Decision Algorithms, *Bull. Pol. Acad. Sci. Tech.*, 33, pp. 487-494494

Pawlak, Z, (1986). Rough Sets and Decision Tables, *Lecture Notes, Springer Verlag*, 208, pp. 186-196196

Pawlak, Z, (1987a). Decision Tables - a Rough Set Approach, *Bull. EATCS*, 33, pp. 85-96

Pawlak, Z. (1987b). Rough Logic. *Bull. Polish Acad. Sci. Tech.*, 35, pp. 253-258.

Rasiowa, H. (1986). Rough Concepts and Multiple Valued Logic. *Proc. of 16th Intl. Symp. on Multiple Valued Logic, Computer Society Press*, pp. 228-288.

Rasiowa, H. and Skowron, A. (1985). Rough Concept Logic. *Proc. of the 5th Symp. on Computer Theory*, Zaborów, December 3-8, 1984. Lecture Notes in Computer Science., Springer Verlag,208, pp. 288-297.

Rasiowa, H. and Skowron, A. (1986a). The First Step Towards and Approximation Logic. *Meeting of the Association for*

Symbolic Logic, Chicago 1985, Journal of Symbolic Logic, 51 pp. 509.

Rasiowa, H. and Skowron, A. (1986b). Approximation Logic. *Proc. of Mathematical Methods of Specification and Synthesis of Software Systems Conf. 1985.* Akademie Verlag,Berlin, 31, pp. 123-139.

Rauszer, C. M. (1984). An Equivalence Between Indiscernibility Relations in Information Systems and a Fragment of Intuitionistic Logic. *Lecture Notes in Computer Science*, Springer Velag, Berlin, Heidelberg, New York, Tokyo, pp. 298-317.

Rauszer, C. M. (1985a). Dependency of Attributes in Information Systems. *Bull. Polish Acad. Sci. Math.*, 33 pp. 551-559.

Rauszer, C. M. (1985b). An Equivalence between Theory of Functional Dependencies and Fragment of Intuitionistic Logic. *Bull. Polish Acad. Sci. Math.*, 33, pp. 571-679.

Rauszer, C. M. (1985c). An Algebraic and Logical Approach to Indiscernibility Relations. *ICS PAS Reports* (1985) No 559.
Rauszer, C. M. (1986). Remarks on Logic for Dependencies. *Bull. Polish Acad. Sci. Math.*, 34, pp. 249-252.

Rauszer, C. M. (1987). Algebraic and Logical Description of Functional and Multivalued Dependencies. *Proc of the Sec. Int. Symp. on Methodologies for Intelligent Systems.* October 17, 1987, Charlotte, North Holland, pp. 145-155.

Rauszer, C. M. (1988). Algebraic Properties of Functional Dependencies. *Bull. Polish Acad. Sci. Math.*, 33, pp. 561-569.

Rauszer, C. M. (1990). Reducts in Information Systems. *Fundamenta Informaticae.* (to appear).

Szczerba, L. W. (1987). Rough Quantifiers. *Bull. Polish Acad. Sci. Math.*, 35, pp. 251-254.

Vakarelov, D. (1981). Abstract Characterization of Some Modal Knowledge Representation Systems and the Logic NIM of Nondeterministic Information. Jorraud, Ph. and Syurev, V. (ed.) *Artificial Intelligence, Methodology, Systems, Applications.* North Holland.

Vakarelov, D. (1989). Modal Logic of Knowledge Representation Systems. *Lecture Notes on Computer Science*, Springer Veralg, 363, pp. 257-277.

Wasilewska, A. (1988). On Correctness of Decision Algorithms in Information Systems. *Fundamenta Informaticae*, 11, pp. 219-239.

Wasilewska, A. (1989). Syntactic Decison Procedures in Information Systems. *International Journal of Man-Machine Studies*, 50, pp. 273-285.