19731

ROUGH SETS
Theoretical Aspects of Reasoning about Data

Zdzisław Pawlak

Warsaw 1990

Contents

Preface

This book is devoted to some areas of research in Artificial Intelligence: knowledge, imprecison, vagueness, learning, induction and others. The topics addressed in this book have long history and overlap with other fields like philosophy, psychology and logic, and variety of issues have been discussed in this context. Because we are aiming at practical applications, therefore we avoid rather general discussion of the considered topics and no attempt is made to set the obtained results in more general framework.

The main issue we are interested in is reasoning from imprecise data, or more specifically, discovering relationships in data. Thus our research interest is closely related to statistics, however our approach is entirely different: instead of employing probability to express vagueness of data we propose using rough sets to this end.

The idea of the rough set consists in approximation of a set by a pair of sets called the lower and the upper approximation of this set.(cf. Pawlak, (1982)). In fact these approximations are interior and closure operations in certain topology generated by available data about elements of the set. In other words, the rough set approach is based on knowledge of an agent (or group of agents) about some reality and his ability to discern some phenomena, processes, objects, etc. Thus the approach is based on the ability to classify data obtained from observation, measurements etc.

The rough set concept overlaps in many aspects many other mathematical ideas developed to deal with imprecision and vagueness, in particular with fuzzy sets theory (cf. Zadeh, (1965)) and theory of evidence (cf. Shafer, (1976)). Interesting comparison of rough sets and fuzzy sets has been published by Dubois and Prade (cf. Dubois et al. (1990), and discussion of the relationship between rough sets and the the evidence theory can be found in Grzymała-Busse (1988) and Skowron (1989). Also it is worthwhile to mention a comparison of the rough set theory and discriminant analysis made by Krusińska, Słowiński and Stefanowski (cf. Krusińska, et al. (1990)).

The idea of the rough set has been proved to be very useful in practice and many real life applications of this concept have been implemented. Exemplary applications are listed below: medical data analysis (cf. Pawlak et al. (1985), Fibak et al. (1986), Słowiński et al. (1988), (1990), Kandulski et al.(1990)), generation of a cement kiln control algorithm from observation of stoker's actions (cf. Mrózek (1987), (1989)), aircraft pilot performance evaluation (cf. Krasowski (1988)), geology (cf. Reinhard et al. (1989), (1990)), pharmacology (cf. Krasiński (1990)), vibration analysis (cf. Nowicki et al. (1990), synthesis of switching circuits (cf. Rybnik (1990)).

Very promising results have been also obtained while using rough sets in voice recognition, approximate classification and others.

There are up to now about 400 articles and rapports published on rough sets theory and its applications. (Selected list of papers on rough sets is enclosed at the end of the book). Further research on the theory and applications of rough sets is under development.

The book is composed of two parts.

In the first part the basic ideas underlying the rough set theory and related subjects are given and the contents is organized according to the mathematical very natural structure of the material presented. Not all issues discussed in this part have direct practical applications. In order however to understand better the considered concepts a elaborate presentation of some problems seems to be justified.

The second part contains some applications of the discussed ideas, however they are not the real life ones but the aim of this part is meant to be rather illustrative as to how the concepts introduced previously can be used to formulate and solve various problems, and does not cover the discussion of variety of important details which occur when more realistic tasks are considered.

The book is intended primarily for computer scientists interested in artificial intelligence however specialists from other fields who are interested in data analysis may also find some parts of the book worth reading.

No previous knowledge on rough sets is needed to understand the book, however some familiarity with basic knowledge of set theory and logic is necessary in order to follow the reasoning. We realize that elaborate mathematical formulation of results is sometimes not welcome in computer science community, we believe however that precision of presentation is badly needed in many areas of artificial intelligence, therefore we try to formulate our claims in precise mathematical language and provide proofs of basic theorems. Secondly our intention is of making the book as areference book in the area of rough sets and their applications, therefore precise definitions and proofs of some theorems seem to be justified. The proofs however are not necessary to follow the applications discussed in Part 2, and can be skipped by those who are not interested in formal aspect of rough sets theory.

Acknowledgements

The author wants to express his gratitude to Professor Cecylia Rauszer and Professor Andrzej Skowron for many beneficial discussions and comments.

References

Grzymała-Busse, J. (1988). Dempster-Shafer Theory Interpretation of Rough-Set Approach to Knowledge Acquisition under Uncertainty.(Manuscript).

Dubois, D. and Prade, H. (1988). Rough Fuzzy Sets and Fuzzy Rough Sets. *Internal Conference on Fuzzy Sets in Informatics, Moscow, September 20-23 and International Journal of General Systems.* (To appear).

Fibak, J., Pawlak, Z., Słowiński, K. and Słowiński. R. (1986). Rough Sets Based Decision Algorithms for Treatment of Duodenal Ulcer by HSV. *Bull. Pol. Acad. Sci. Biol.*, **34**, pp. 227-246.

Fibak, J., Słowiński, K. and Słowiński, R. (1986). The Application of Rough Set Theory to the Verification of Indications for Treatment of Duodenal Ulcer by HSV. *Proc. 6th International Workshop on Expert Systems & Their Applications, Avignon, France, April 28-30, 1986*, pp. 587-599.

Kandulski, T., Litewka, B., Mrózek, A. and Tukałło. K. (1990). An Attempt to Establish the Hierarchy of Factors of a Surgical Wound Infection by Means of the Rough Set Theory. *Bull. Acad. Sci. Biol.*, (To appear).

Krasowski. H. (1988). Aircraft Pilot Performance Evaluation Using Rough Sets. *Ph. D. Dissertation*, Technical University of Rzeszów (Poland). (In Polish).

Krusińska, E., Słowiński, R. and Stefanowski, J. (1990). Discriminant Versus Rough Sets Approach to Vague Data Analysis. *Journal of Applied Statistics and Data Analysis*, (to appear)

Krysiński, J. (1990). Rough Set Approach to Analysis of Relationship between Structure and Activity of Quaternary Imidazolium Compounds. *Arzenmittel-Forschung Drug Research.* (To appear).

Mrózek, A. (1987). Rough Sets and Some Aspects of Expert Systems Realization. *Proc. 7-th International Workshop on Expert Systems & Their Applications, Avignon, France, 1987*, pp. 597-611.

Mrózek, A. (1989). Rough Set Dependency Analysis Among Attributes in Computer Implementation of Expert Inference Models. *Int. Journal of Man-Machine Studies*, **30**, pp. 457-473.

Nowak, R., Słowiński, R. and Stefanowski, J. (1990). Rough Sets Based Diagnostic Classifier of Reducers. *Maintenance Management International.* (Submitted).

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

Pawlak, Z., Słowiński, K. and Słowiński, R. (1986). Rough Classification of Patients After Highly Selective Vagotomy for Duodenal Ulcer. *Int. Journal of Man-Machine Studies,* 24, pp. 413-433.

Reinhard, A., Stawski, B. and Weber, T. (1989). Application of Rough Sets to Study the Water Outflow from the River Basin. *Bull. Pol. Acad. Sci. Tech.*, 37, pp. 97-104.

Reinhard, A., Stawski, B. Szwast, W. and Weber, T. (1990). An Attempt to Use the the Rough Sets Theory for the Control of Water-Air Relation on a Given Polder. *Bull. Pol. Acad. Sci. Tech.*, (To appear).

Rybnik, J. (1990). Minimization of Partially Defined Switching Functions Using Rough Sets. *Ph. D. Dissertation.*

Shafer, G. (1976). A Mathematical Theory of Evidence. Princeton Univ. Press, Princeton, N.Y.

Skowron. A. (1989). The Relationship Between the Rough Set Theory and Evidence Theory. *Bull. Pol. Acad. Sci. Tech.*, 37, pp. 87-90.

Słowiński, K., Słowiński, R. and Stefanowski, J. (1989). Rough Sets Approach to Analysis of Data from Peritoneal Lavage in Acute Pancreatitis. *Medical Informatics,* 13, pp. 143-159.

Słowiński, R. and Stefanowski, J. (1989). Rough Classification in Incomplete Information Systems. *Mathematical and Computing Modeling,* 12, pp. 1347-1357.

Słowiński, K. and Słowiński, R. (1990). Sensitivity Analysis of Rough Classification. *Int. Journal of Man-Machine Studies*. **32**, pp. 693-705.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*. **8**, pp. 338-353.

Motto: " knowing is a relation of the organizm to something else or to a part of itself". Bertrand Russell, in An Inquiry into Meaning and Truth

P A R T I

T H E O R E T I C A L F O U N D A T I O N S

1. KNOWLEDGE

1. Introduction

Theory of knowledge has long lasting and rich history (cf., Hempel (1952), Hintika (1962), Hunt (1974), Hunt et. al. (1966), Popper (1959), Russell (1940)).

Various aspects of knowledge are widely discussed issues nowadays, mainly by logicians and Artificial Intelligence (AI) researchers. There is, however a variety of opinions and approaches in this area, as to how to understand, represent and manipulate knowledge (cf. Aikins (1983), Bobrow (1977), Bobrow et al. (1977), Brachman et al.(1980), Brachman et al. (1986), Buchanann et al.(1984), Davis et al. (1982), Halpern (1986), (Hayes-Roth et al. (1978), Holland et al. (1986), McDermott (1978), Minski (1975), Newell (1982).

Intuitively, knowledge can be perceived as a body of information about some parts of reality, which constitute our domain of interest. This definition, however fails to meet precision standards and at a closer look it has multiple meanings, and it tends to mean one of several things depending on the context and the area of interest.

We propose here a formal definition of the term "knowledge" and we show some of its basic properties. We do not aim to give full account of the concept of knowledge and we realize that the proposed understanding of knowledge might seem to be not sufficiently general to cover various understanding of this concept in current literature, in particular in widely assumed paradigm of knowledge in the AI community nowadays - yet it seems to be of interest for a variety of domains, like machine learning, pattern recognition, decision support systems, expert systems and others.

The concept of knowledge presented here is rather close to that considered in some areas of cognitive sciences, than that discussed in AI. We do not, however, aim to form a new, general theory of knowledge, in contrast for example to Holland (cf. Holland et al. (1986)), but we have in mind rather practical applications.

We advocate here a rough set concept as a theoretical framework for discussions about knowledge, particularly when imprecise knowledge is of primary concern.

2. Knowledge and Classification

Our claim is that knowledge is deep-seated in the classificatory abilities of human beings and other species. For example, knowledge about the environment is primarily manifested as an ability to classify a variety of situations from the point of view of survival in the real world. Complex classification patterns of sensor signals probably form fundamental mechanisms of every living being. Classification on more abstract levels, seems to be the key issue in reasoning, learning and decision making, not to mention that in science classification it is of primary importance, too.

Also a robot which would be able to behave "intelligently" in an environment, exploiting sensory signals about outer realm and its internal states must classify possible situations and act accordingly.

We simply assume here, that knowledge is based on the ability to classify objects, and by object we mean anything we can think of, for example, real things, states, abstract concepts, processes, moments of time, etc.

Thus knowledge in our approach is strictly connected with the variety of classification patterns related to specific parts of real or abstract world, called here *the universe of discourse* (in short *the universe*). Nothing particular about the nature of the universe and knowledge will be assumed here. In fact knowledge consists of a family of various classification patterns, of a domain of interest, which provide *explicit* facts about the reality - together with the reasoning capacity able to deliver *implicit* facts derivable from *explicit* knowledge.

In what follows we shall explain this idea in some more detail. First we are going to discuss more precisely some properties of classifications and reasoning about classifications will be considered later.

3. Knowledge Base

Suppose we are given a finite set U (the universe) of objects, we are interested in. Any subset $X \subseteq U$ of the universe will be called a *concept* or a *category* in U and any family of concepts in U will be referred to as *abstract knowledge* (or in short *knowledge*) about U. For formal reason we also admit the empty set $\emptyset$ as a concept.

Mostly we will be interested in this book with concepts which form a partition (classification) of a certain universe U, i.e. in families $C = \{X_1, X_2, \ldots, X_n\}$ such that $X_i \subseteq U$, $X_i \neq \emptyset$, $X_i \cap X_j = \emptyset$ for $i \neq j$, $i,j=1,\ldots,n$ and $\bigcup X_i = U$.

Usually we will deal not with a single classification but with some families of classifications over U. A family of classifications over U will be called a *knowledge base* over U. Thus knowledge base represents a variety of basic classification skills (e.g. according to colors, temperature, etc) of an "intelligent" agent or group of agents (e.g. organisms or robots) which constitute the fundamental equipment of the agent needed to define its relation to the environment or itself.

In what follows we are going to explain the idea more precisely.

For mathematical reasons we shall often use instead of classifications - equivalence relations, since these two notions are mutually exchangeable and relations are easier to deal with. Let us give now some necessary definitions.

If R is an equivalence relation over U, then by U/R we mean the family of all equivalence classes of R (or classification of U) referred to as *categories* or *concepts* of R, and $[x]_R$ denotes a category in R containing an element $x \in U$.

By a *knowledge base* we can understand now a relational system $K = (U, \mathbf{R})$, where U is a finite set called *the universe*, and $\mathbf{R}$ is a family of equivalence relations over U.

If $\mathbf{P} \subseteq \mathbf{R}$ and $\mathbf{P} \neq \emptyset$, then $\bigcap \mathbf{P}$ (intersection of all equivalence relations belonging to $\mathbf{P}$) is also an equivalence relation, and will be denoted by $IND(\mathbf{P})$, and will be called an *indiscernibility relation* over $\mathbf{P}$. Moreover

$$[x]_{IND(\mathbf{P})} = \bigcap_{R \in \mathbf{P}} [x]_R \, .$$

Thus $U/IND(\mathbf{P})$ (i.e. the family of all equivalence classes of the equivalence relation $IND(\mathbf{P})$) denotes knowledge associated with the family of equivalence relations $\mathbf{P}$, called **P**-*basic knowledge about U in K*. For simplicity of notation we will write $U/\mathbf{P}$ instead of $U/IND(\mathbf{P})$ and $\mathbf{P}$ will be also called **P**-*basic knowledge*, (or in short - *basic knowledge*, if $\mathbf{P}$, U and K is understood) provided that it does not cause confusion. Equivalence classes of $IND(\mathbf{P})$ are called *basic categories (concepts)* of knowledge $\mathbf{P}$. In particular if $Q \in \mathbf{R}$, then Q will be called a *Q-elementary knowledge* (about U in K) and equivalence classes of Q are referred to as *Q-elementary concepts (categories)* of knowledge $\mathbf{R}$.

For example if elements of the universe are categorized according to colors, then the corresponding elementary categories would be subsets of all objects having specific colors, for instance *green*, *red* etc., whereas basic categories are combined from some elementary categories. For example if *old* and *ill* are elementary categories, in some knowledge base then *old and ill* is a basic category in this knowledge base.

In fact **P**-basic categories are those basic properties of the universe which can be voiced employing knowledge **P**. In other words they are fundamental building blocks of our knowledge, or basic properties of the universe which can be expressed employing knowledge **P**.

The family of all **P**-basic categories for all $\emptyset \neq \mathbf{P} \subseteq \mathbf{R}$ will be called the family of *basic categories* in knowledge base $K = (U, \mathbf{R})$.

We will also need the following notations. Let $K = (U, \mathbf{R})$

be a knowledge base. By *IND*(*K*) we denote the family of all equivalence relations defined in *K* as *IND*(*K*) = {*IND*(**P**): ∅ ≠ **P** ⊆ **R**}

Thus *IND*(*K*) is the minimal set of equivalence relations, containing all elementary relations of *K* and closed under set theoretical intersection of equivalence relations.

Every union of **P**-basic categories will be called **P**-*category*.

Finally the family of all categories in the knowledge base *K* = (*U*, **R**) will be referred to as *K*-categories.

Example 1

Suppose we are given the following set of toy blocks $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$. Assume that these toys have different colors (red, blue, yellow), shapes, (square, round, triangular) and size (small, large). For example a toy block can be small, red and round or large, square and yellow etc.

Thus the set of toy blocks *U* can be classified according to color, shape and size, for example as shown below.

Toys

x_1, x_3, x_7	- are *red*,
x_2, x_4	- are *blue*,
x_5, x_6, x_8	- are *yellow*,

toys

x_1, x_5	- are *round*,
x_2, x_6	- are *square*,
x_3, x_4, x_7, x_8	- are *triangular*,

and toys

x_2, x_7, x_8	- are *large*,
x_1, x_3, x_4, x_5, x_6	- are *small*.

In other words by these classifications we defined three equivalence relations R_1, R_2 and R_3 having the following equivalence classes

$$U/R_1 = \{\{x_1, x_3, x_7\}, \{x_2, x_4\}, \{x_5, x_6, x_8\}\}$$

$$U/R_2 = \{\{x_1, x_5\}, \{x_2, x_6\}, \{x_3, x_4, x_7, x_8\}\}$$

$$u/R_3 = \{\{x_2, x_7, x_8\}, \{x_1, x_3, x_4, x_5, x_6\}\}$$

which are elementary concepts (categories) in our knowledge base $K = (U, \{R_1, R_2, R_3\})$.

Basic categories are set theoretical intersections of elementary categories. For example sets

$$\{x_1, x_3, x_7\} \cap \{x_3, x_4, x_7, x_8\} = \{x_3, x_7\}$$

$$\{x_2, x_4\} \cap \{x_2, x_6\} = \{x_2\}$$

$$\{x_5, x_6, x_8\} \cap \{x_3, x_4, x_7, x_8\} = \{x_8\}$$

are $\{R_1, R_2\}$-basic categories *red and triangular*, *blue and square*, *yellow and triangular* respectively. Sets

$$\{x_1, x_3, x_7\} \cap \{x_3, x_4, x_7, x_8\} \cap \{x_2, x_7, x_8\} = \{x_7\}$$

$$\{x_2, x_4\} \cap \{x_2, x_6\} \cap \{x_2, x_7, x_8\} = \{x_2\}$$

$$\{x_5, x_6, x_8\} \cap \{x_3, x_4, x_7, x_8\} \cap \{x_2, x_7, x_8\} = \{x_8\}$$

are exemplary $\{R_1, R_2, R_3\}$-basic categories *red and triangular and large*, *blue and square and large*, *yellow and triangular and small* respectively. Sets

$$\{x_1, x_3, x_7\} \cup \{x_2, x_4\} = \{x_1, x_2, x_3, x_4, x_7\}$$

$$\{x_2, x_4\} \cup \{x_5, x_6, x_8\} = \{x_2, x_4, x_5, x_6, x_8\}$$

$$\{x_1, x_3, x_7\} \cup \{x_5, x_6, x_8\} = \{x_1, x_3, x_5, x_6, x_7, x_8\}$$

are R_1-categories *red or blue (not yellow)*, *blue or yellow* *(not red), red or yellow (not blue)* respectively.

Note that some categories are not available in this knowledge base. For example sets

$$\{x_2, x_4\} \cap \{x_1, x_5\} = \emptyset$$

$$\{x_1, x_3, x_7\} \cap \{x_2, x_6\} = \emptyset$$

are empty which means that categories *blue and round* and *red and square* do not exist in our knowledge base (are empty categories).■

4. Equivalence, Generalization and Specialization of Knowledge

Let $K = (U, P)$ and $K' = (U, Q)$ be two knowledge bases. We will say that K and K' (P and Q) are *equivalent*, denoted $K \simeq K'$, $(P \simeq Q)$, if $IND(P) = IND(Q)$, or what is the same, if $U/P = U/Q$. Hence $K \simeq K'$, if both K and K' have the same set of basic categories, and consequently - the set of all categories. This means that knowledge in knowledge bases K and K' enables us to express exactly the same facts about the universe.

Let $K = (U, P)$ and $K' = (U, Q)$ be two knowledge bases. If $IND(P) \subset IND(Q)$ we say that knowledge P (knowledge base K) is *finer* than knowledge Q, (knowledge base K'), or Q is *coarser* than P. We will also say, that if P is finer than Q, then P is *specialization* of Q, and Q is *generalization* of P.

For example if $P, Q \in R$, and both U/P and U/Q are classifications of the universe with respect to color, but the classification U/P contains one category of, say green objects, whereas the classification U/Q contains more categories of green objects, each referring to specific shadow of green, (and similarly for other colors) - than Q is specialization of P, and P is generalization of Q, provided that every category of shadow of green in U/Q is included in the category of green in U/P.

Thus generalization consists in combining together some categories, whereas specialization lies in splitting categories into smaller units.

It is worthwhile to mention, that if all equivalence classes of an equivalence relation are single element sets, then the relation is an equality relation and in such case the relation represents the most accurate (precise) knowledge, which might seem to be the most desirable situation, but this is often not the case.

Let us turn back to the example of colors. Precise distinguishing of colors, unable us to form one color category, say green. Thus in order to have categories, like green ,red, small, tall, hot, cold etc. we have to have the ability to combine object into groups ignoring "small" differences between them, or in other words group them together according to some similarities. (Usually similarity is expressed by a dischtance function, but in our approach we avoid numerical characterization of imprecision employing quantitative concepts (classification) to this end, instead). Hence finer categorization should not always be interpreted as an advantage, for it may sometimes make difficult forming categories (concepts).

In our philosophy, classification (partitioning) is basically used to create categories which are "building blocks" of knowledge.

Summary

We have shown in this chapter that knowledge is strictly connected with classification (partition), and "building blocks" of knowledge, categories, are classes of some classifications.

Exercises

1. Compute the family of all equivalence classes of the relation $R_1 \cap R_2 \cap R_3$, i.e. $U/(R_1 \cap R_2 \cap R_3)$, where R_1, R_2 and R_3 are as in the Example 1.

2. Are the following sets

$\{x_5\}$

$\{x_2, x_6, x_8\}$

$\{x_1, x_5, x_6, x_7\}$

concepts in this knowledge base.

3. Check whether the knowledge bases $K = (U, R_1, R_2, R_3)$ and $K' = (U, R_1, R_2)$, are equivalent or not, where $U = \{x_1, x_2, x_3, x_4, x_5\}$, and

$$U/R_1 = \{\{x_1, x_3\}, \{x_2, x_4, x_5\}\}$$

$$U/R_2 = \{\{x_1\}, \{x_2, x_3, x_4, x_5\}\}$$

$$U/R_3 = \{\{x_1, x_4\}, \{x_2, x_3\}. \{x_5\}\}.$$

4. Give examples of two knowledge bases K_1 and K_2such that K_1 is finer then K_2.

5. Give real life examples of generalization and specialization of knowledge.

References

Aikins, J. S. (1983). Prototypic a Knowledge for Expert Systems. *Artificial Intelligence*, **20**, pp. 163-210.

Bobrow, D. G. (1977). A Panel on Knowledge Representation. *Proc. Fifth International Joint Conference on Artificial Intelligence, Carnegie-Melon University*. Pittsburgh, PA.

Bobrow, D. G. and Winograd, T. (1977). An Overview of KRL: A Knowledge Representation Language. *Journal of Cognitive Sciences*, **1**, pp. 3-46.

Brachman, R. J. and Smith, B. C. (1980). Special Issue of Knowledge Representation. *SIGART Newsletter*, **70**, pp. 1-138.

Brachman, R.J. and Levesque, H. J. (Ed.) (1986). Readings in Knowledge Representation, *Morgan Kaufmann Publishers, Inc.*

Buchanan, B. and Shortliffe, E. (19884). Rule Based Expert Systems. *Reading, Massachusetts:* Addison-Wesley.

Davis, R. and Lenat, D. (1982). Knowledge-Based Systems in Artificial Intelligence. *McGrav-Hill.*

Halpern, J. (ed.) (1986). Theoretical Aspects of Reasoning about Knowledge. *Proceedings of the 1986 Conference.* Morgan Kaufman, Los Altos, California.

Hayes-Roth, B. and McDermott, J. (1978). An Inference Matching for Inducing Abstraction. *Communication of the ACM*, 21, pp. 401-410.

Hempel, C. G. (1952). Fundamental of Concept Formation in Empirical Sciences. *University of Chicago Press.* Chicago.

Hintika, J. (1962). Knowledge and Belief. *Cornell University Press.* Chicago.

Holland, J. H., Holyoak, K. J., Nisbett, R. E. and Thagard, P. R. (1986). Induction: Processes of Inference, Learning, and Discovery. *MIT Press.*

Hunt, E. B. (1974). Concept Formation. *John Wiley and Sons.* New York.

Hunt, E. D., Marin, J. and Stone, P. J. (1966). Experiments in Inductions. *Academic Press.* New York.

McDermott, D. (1978). The Last Survey of Representation of Knowledge. *Proc. of the AISB/GI Conference on AI.* Hamburg, pp. 286-221.

Minski, M. (1975). A Framework for Representation Knowledge. *In: Winston, P. (ed.) The Psychology of Computer Vision.*

McGraw-Hill. New York, pp. 211-277.

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 18, pp. 87-127.

Popper, K. (1959). The Logic of Scientific Discovery. London: Hutchinson.

Russell, B. (1950). An Inquiry into Meaning and Truth. *George Allen and Unwin*, London.

2. IMPRECISE CATEGORIES, APPROXIMATIONS AND ROUGH SETS

1. Introduction

Fundamental concepts in the proposed theory of knowledge are classifications and categories. In fact categories are features (i.e. subsets) of objects which can be worded using knowledge available in a given knowledge base. Certainly some categories can be definable in one knowledge base but undefinable in another one. Thus, if a category is not definable in a given knowledge base, the question arises whether it can be defined "approximately" in the knowledge base. In other words we want to address here the central point of our approach, the vague categories.

There is a variety of conceptions of vagueness in logical and philosophical literature (cf. Balmer et al. (1983), Black (1937, 1963), Fine (1975), Kohl (1969), Russel (1923)). The one presented here is a direct consequence of the assumed understanding of knowledge and is based on the idea of a rough set, which will be our major concern in this chapter. In the next section we will examine the idea more precisely.

2. Rough Sets

Let $X \subseteq U$, and R be an equivalence relation. We will say that X is *R-definable*, if X is union of some R-basic categories; otherwise X is *R-undefinable*.

The R-definable sets are those subsets of the universe which can be exactly defined in the knowledge base K, whereas the R-undefinable sets cannot be defined in this knowledge base.

The R-definable sets will be also called *R-exact* sets, and R-undefinable sets will be also said to be *R-inexact* or *R-rough*.

Set $X \subseteq U$ will be called *exact in K* if there exists an equivalence relation $R \in IND(K)$ such that X is R-exact, and X is said to be *rough in $K*$, if X is R-rough for any $R \in IND(K)$.

Rough sets can however be defined approximately and to this end we will employ two exact sets referred to as a lower and an upper approximation of the set.

In the next chapter we are going to examine the idea more closely.

Remark

The idea of the rough set was proposed by the author in Pawlak (1982). By now number of papers have been published on rough set theory, but we will not go into details about rough sets and we confine our shelf only to some basic properties needed in the remainder of this book.

This chapter is an extended version of Pawlak (1989). More about rough sets can be found for example in Iwiński (1987), Nieminen (1988), Novotny (1985a, 1985b, 1985c), and Obtułowicz (1988).

3. Approximation of Sets

As we already demonstrated in the previous section some categories (subsets of objects) cannot be expressed exactly by employing available knowledge. Hence we arrive at the idea of approximation of set by another sets, which will be discussed in detail in this section. Suppose we are given knowledge base $K = (U,R)$. With each subset $X \subseteq U$ and an equivalence relation $R \in IND(K)$. we associate two subsets:

$$\underline{R}X = \bigcup\{Y \in U/R: Y \subseteq X\}$$

$$\bar{R}X = \bigcup\{Y \in U/R: Y \cap X \neq \emptyset\}$$

called the *R-lower* and *R-upper approximation* of X respectively.

Remark.

The lower and upper approximations can be also presented in an equivalent form as shown below:

$$\underline{R}X = \{x \in U: [x]_R \subseteq X\}$$

$$\bar{R}X = \{x \in U: [x]_R \cap X \neq \emptyset\}$$

or

$$x \in \underline{R}X \text{ if and only if } [x]_R \subseteq X$$

$$x \in \bar{R}X \text{ if and only if } [x]_R \cap X \neq \emptyset$$

Set $BN_R(X) = \bar{R}X - \underline{R}X$ will be called the *R-boundary* of X.

The set $\underline{R}X$ is the set of all elements of U which can be with *certainty* classified as elements of X, in the knowledge R; Set $\bar{R}X$ is the set of elements of U which can be *possibly* classified as elements of X, in employing knowledge R; Set $BN_R(X)$ is the set of elements which cannot be classified either to X or to $-X$ having knowledge R.

We shall also employ the following denotations:

$POS_R(X) = \underline{R}X$, *R-positive region* of X

$NEG_R(X) = U - \underline{R}X$, *R-negative region of* X.

$BN_R(X)$ - *R-borderline region* of X.

If an object $x \in POS_R(X)$, then x will be called a *R-positive example of* X, and similarly for $NEG_R(X)$ and $BN_R(X)$.

The positive region $POS_R(X)$ or the lower approximation of X is the collection of those objects which can be classified with full certainty as members of the set X, using knowledge R.

Similarly, the negative region $NEG_R(X)$ is the collection of objects with which it can be determined without any ambiguity, employing knowledge R, that they do not belong to the set X, that is, they belong to the complement of X.

The boundary region is in a sense undecidable area of the universe, i.e. none of the objects belonging to the boundary can be classified with certainty into X or $-X$ as far as knowledge R is concern.

Finally, the upper approximation of X consists of objects with which knowledge R does not allow us to exclude the possibility that those objects may belong to X. Formally, the upper approximation is the set theoretical union of positive and boundary regions.

The following property is obvious:

Proposition 1.

a) X is R-definable if and only if $\underline{R}X = \bar{R}X$.

b) X is rough with respect to R if and ony if $\underline{R}X \neq \bar{R}X$. ■

Let us also observe that $\underline{R}X$ is the maximal R-definable set contained in X, whereas $\bar{R}X$ is the minimal R-definable set containing X.

Thus categories are items of information which can be expressed by available knowledge. In other words, categories are subsets of objects having the same properties expressible in terms of our knowledge. In general, not all subset of object form categories in a given knowledge base, i.e. concepts which can be expressed by the knowledge, therefore such subsets may be regarded as rough categories (i.e. imprecise or approximate categories) which can be only roughly defined employing our knowledge - by using two exact categories -the lower and the upper upper approximation.

Hence the concept of approximation allows us to speak precisely about imprecise notions.

4. Properties of Approximation

Directly from the definition of approximations we can get the following properties of the R-lower and the R-upper approximations:

Proposition 2.

1) $\underline{R}X \subseteq X \subseteq \bar{R}X$

2) $\underline{R}\emptyset = \bar{R}\emptyset = \emptyset$; $\underline{R}U = \bar{R}U = U$

3) $\bar{R}(X \cup Y) = \bar{R}X \cup \bar{R}Y$

4) $\underline{R}(X \cap Y) = \underline{R}X \cap \underline{R}Y$

5) $X \subseteq Y$ implies $\underline{R}X \subseteq \underline{R}Y$

6) $X \subseteq Y$ implies $\bar{R}X \subseteq \bar{R}Y$

7) $\underline{R}(X \cup Y) \supseteq \underline{R}X \cup \underline{R}Y$

8) $\bar{R}(X \cap Y) \subseteq \bar{R}X \cap \bar{R}Y$

9) $\underline{R}(-X) = -\bar{R}X$

10) $\bar{R}(-X) = -\underline{R}X$

11) $\underline{R}\underline{R}X = \bar{R}\underline{R}X = \underline{R}X$

12) $\bar{R}\bar{R}X = \underline{R}\bar{R}X = \bar{R}X$

Proof.

1a) If $x \in \underline{R}X$, then $[x] \subseteq X$, but $x \in [x]$ hence $x \in X$ and $\underline{R}X \subseteq X$.

1b) If $x \in X$, then $[x] \cap X \neq \emptyset$ (because $x \in [x] \cap X$) hence $x \in \bar{R}X$, and $X \subseteq \bar{R}X$.

2a) From 1) $\underline{R}\emptyset \subseteq \emptyset$ and $\emptyset \subseteq \underline{R}\emptyset$ (because the empty set is included in every set) thus $\underline{R}\emptyset = \emptyset$.

2b) Assume $\bar{R}\emptyset \neq \emptyset$. Then there exists x such that $x \in \bar{R}\emptyset$. Hence $[x] \cap \emptyset \neq \emptyset$, but $[x] \cap \emptyset = \emptyset$, what contradicts the assumption, thus $\bar{R}\emptyset = \emptyset$.

2c) From 1) $\underline{R}U \subseteq U$. In order to show that $U \subseteq \underline{R}U$ let us observe that if $x \in U$, then $[x] \subseteq U$, hence $x \in \underline{R}U$, thus $\underline{R}U = U$.

2d) From 1) $\bar{R}U \supseteq U$, and obviously $\bar{R}U \subseteq U$, thus $\bar{R}U = U$.

3) $x \in \bar{R}(X \cup Y)$ iff $[x] \cap (X \cup Y) \neq \emptyset$ iff $[x] \cap X \cup [x] \cap Y \neq \emptyset$ iff $[x] \cap X \neq \emptyset \vee [x] \cap Y \neq \emptyset$ iff $x \in \bar{R}X \vee x \in \bar{R}Y$ iff $x \in \bar{R}X \cup \bar{R}Y$. Thus $\bar{R}(X \cup Y) = \bar{R}X \cup \bar{R}Y$.

4) $x \in \underline{R}(X \cap Y)$ iff $[x] \subseteq X \cap Y$ iff $[x] \subseteq X \wedge [x] \subseteq Y$ iff $x \in RX \cap RY$.

5) Because $X \subseteq Y$ iff $X \cap Y = X$ by virtue of 4) we have $\underline{R}(X \cap Y) = \underline{R}X$ iff $\underline{R}X \cap \underline{R}Y = \underline{R}X$ which yields $\underline{R}X \subseteq \underline{R}Y$.

6) Because $X \subseteq Y$ iff $X \cup Y = Y$, hence $\overline{R}(X \cup Y) = \overline{R}Y$ and by virtue of 3) we have $\overline{R}X \cup \overline{R}Y = \overline{R}Y$ and hence $\overline{R}X \subseteq \overline{R}Y$.

7) Since $X \subseteq X \cup Y$ and $Y \subseteq X \cup Y$, we have $\underline{R}X \subseteq \underline{R}(X \cup Y)$ and $\underline{R}Y \subseteq \underline{R}(X \cup Y)$ which yields $\underline{R}X \cup \underline{R}Y \subseteq \underline{R}(X \cup Y)$.

8) Since $X \cap Y \subseteq X$ and $X \cap Y \subseteq Y$, we have $\overline{R}(X \cap Y) \subseteq \overline{R}X$ and $\overline{R}(X \cap Y) \subseteq \overline{R}Y$ hence $\overline{R}(X \cap Y) \subseteq \overline{R}X \cap \overline{R}Y$.

9) $x \in \underline{R}(x)$ iff $[x] \subseteq X$ iff $[x] \cap -X = \emptyset$ iff $x \notin \overline{R}(-X)$ iff $x \in -\overline{R}(-X)$, hence $\underline{R}(X) = -\overline{R}(-X)$.

10) By substitution $-X$ for X in 9) we get $\overline{R}(X) = -\underline{R}(-X)$.

11a) From 1) $\underline{R}\underline{R}X \subseteq \underline{R}X$, thus we have to show that $\underline{R}X \subseteq \underline{R}\underline{R}X$. If $x \in \underline{R}X$ then $[x] \subseteq X$, hence $\underline{R}[x] \subseteq \underline{R}X$ but $\underline{R}[x] = [x]$, thus $[x] \subseteq \underline{R}X$ and $x \in \underline{R}\underline{R}X$, that is $\underline{R}X \subseteq \underline{R}\underline{R}X$.

11b) From 1) $\underline{R}X \subseteq \overline{R}\underline{R}X$, thus it is enough to show that $\underline{R}X \supseteq \overline{R}\underline{R}X$. If $x \in \overline{R}\underline{R}X$, then $[x] \cap \underline{R}X \neq \emptyset$, i.e. there exists $y \in [x]$ such that $y \in \underline{R}X$, hence $[y] \subseteq X$ but $[x] = [y]$, thus $[x] \subseteq X$ and $x \in \underline{R}X$ which is to mean that $\underline{R}X \supseteq \overline{R}\underline{R}X$.

12a) From 1) $\overline{R}X \subseteq \overline{R}\overline{R}X$. We have to show, that $\overline{R}X \supseteq \overline{R}\overline{R}X$. If $x \in \overline{R}\overline{R}X$, then $[x] \cap \overline{R}X \neq \emptyset$ and for some $y \in [x]$ $y \in \overline{R}X$, hence $[y] \cap X \neq \emptyset$ but $[x] = [y]$, thus $[x] \cap X \neq \emptyset$, i.e. $x \in \overline{R}X$, which yields $\overline{R}X \supseteq \overline{R}\overline{R}X$.

12b) From 1) $\underline{R}\overline{R}X \subseteq \overline{R}X$. We have to show, that $\underline{R}\overline{R}X \supseteq \overline{R}X$. If $x \in \overline{R}X$ then $[x] \cap X \neq \emptyset$. Hence $[x] \subseteq \overline{R}X$ (because if $y \in [x]$, then $[y] \cap X = [x] \cap X \neq \emptyset$, i.e. $y \in \overline{R}X$) and $x \in \underline{R}\overline{R}X$, which gives $\underline{R}\overline{R}X \supseteq X$.■

Example 1.

For the sake of illustration let us consider a very simple example depicting the introduced notions. Suppose we are given an knowledge base $K = (U, \mathbf{R})$, where $U = \{x_1, x_2, \ldots, x_8\}$, and an equivalence relation $R \in IND(K)$ with the following equivalence classes :

$$E_1 = \{x_1, x_4, x_8\}$$

$$E_2 = \{x_2, x_5, x_7\}$$

$$E_3 = \{x_3\}$$

$$E_4 = \{x_6\}$$

i.e.

$$U/R = \{E_1, E_2, E_3, E_4\}$$

Let

$$X_1 = \{x_1, x_4, x_7\} \text{ and } X_2 = \{x_2, x_8\}.$$

$$\underline{R}(X_1 \cup X_2) = E_1$$

$$\underline{R}X_1 = \emptyset, \ \underline{R}X_2 = \emptyset.$$

Hence

$$\underline{R}(X_1 \cup X_2) \supset \underline{R}X_1 \cup \underline{R}X_2.$$

Let

$$Y_1 = \{x_1, x_3, x_5\} \text{ and } Y_2 = \{x_2, x_3, x_4, x_6\}.$$

$$\bar{R}(Y_1 \cap Y_2) = E_3$$

$$\bar{R}Y_1 = E_1 \cup E_2 \cup E_3$$

$$\bar{R}Y_2 = E_1 \cup E_2 \cup E_3 \cup E_4$$

Hence

$$\bar{R}(Y_1 \cap Y_2) \subset \bar{R}Y_1 \cap \bar{R}Y_2. \qquad \blacksquare$$

Let us briefly comment on some properties of approximations.

The properties 7) and 8) are of great importance for they do not allow for step by step computation of approximations. In fact they say that in general the knowledge included in a "distributed" knowledge base is less than in the integrated one. Or in other words dividing the knowledge base into smaller units in general causes loss of information. This is a quite obvious remark, however what is interesting here, that this property is a logical, formal consequence of a assumed definition of the knowledge base and imprecision, expressed in the form of approximations.

The properties 9) and 10) are also of interest and demonstrate the nature of the relationship between the lower and the upper approximation of sets.

Remark.

Let us note that each knowledge base $K = (U, \mathbf{R})$ uniquely defines a topological space $T_A = (U, DIS(R))$, where $DIS(R)$ is the family of all open and closed sets in T_A (topology for U), and U/R is a base for T_A. The R-lower and the R-upper approximation of X in A are interior and closure operations in the topological space T_A respectively.

The topology generated by the equivalence relations, differs however from the general topology because properties $\bar{R}\underline{R}X = \underline{R}X$ and $\underline{R}\bar{R}X = \bar{R}X$ are not valid in the general topology, where we have instead $\bar{R}\underline{R}X = \underline{R}\bar{R}\underline{R}\bar{R}X$ and $\underline{R}\bar{R}X = \bar{R}\underline{R}\bar{R}\underline{R}X$, respectively.

More about the topology generated by the equivalence relation can be found in Wiweger (1988) and Skowron (1988).

5. Approximations and Membership Relation

The concept of approximations of sets leads to a new conception of membership relation. Because definition of a set in our approach is associated with knowledge about the set, hence also a membership relation must be related to the knowledge. Formally this can be defined as follows:

$$x \underline{\in}_R X \text{ if and only if } x \in \underline{R}X$$
$$x \bar{\in}_R X \text{ if and only if } x \in \bar{R}X$$

where $\underline{\in}_R$ reads "x *surely belongs* to X with respect to R " and $\bar{\in}_R$ - "x possibly *belongs* to X with respect to R ", and will be called the *lower* and the *upper* membership relation respectively.

Both membership relations again are referring to our knowledge, i.e. whether an object belongs to a set depends upon our knowledge and it is not an absolute property.

Immediately from **Proposition 2** we obtain the following properties of membership relations.

Proposition 3

1) $x \underline{\in} X$ implies $x \in X$ implies $x \bar{\in} X$
2) $X \subseteq Y$ implies ($x \underline{\in} X$ implies $x \underline{\in} Y$ and $x \bar{\in} X$ implies $x \bar{\in} Y$)
3) $x \bar{\in} (X \cup Y)$ if and only if $x \bar{\in} X$ or $x \bar{\in} Y$
4) $x \underline{\in} (X \cap Y)$ if and only if $x \underline{\in} X$ and $x \underline{\in} Y$
5) $x \underline{\in} X$ or $x \underline{\in} Y$ implies $x \underline{\in} (X \cup Y)$
6) $x \bar{\in} (X \cap Y)$ implies $x \bar{\in} X$ and $x \bar{\in} Y$
7) $x \underline{\in} (-X)$ if and only if non $x \bar{\in} X$
8) $x \bar{\in} (-X)$ if and only if non $x \underline{\in} X$ ■

For simplicity we dropped the subscript R in the above formulas.

The membership relation is a basic issue when speaking about sets. In set theory absolute knowledge is required about elements of the universe, i.e. we assume that each element of the universe can be properly classified to a set X or its complement for any $X \subseteq U$. In our philosophy, this is not the case since we claim that the membership relation is not a primitive notion but one that must be based on knowledge we have about objects to be classified. As a consequence two membership relations are necessary to express the fact that some objects of the universe cannot be properly classified employing available knowledge. That is still another explanation of the fact that imprecise categories cannot be precisely defined by available knowledge.

It should be quite obvious that precise categories do not require two membership relations, but that one "classical" membership relation suffices.

In the case when R is an equality relation, all three membership relation are the same and coincide with ordinary set theoretical membership relation.

6.Numerical Characterization of Imprecision

Inexactness of a set (category) is due to the existence of a boundary region. The greater the boundary region of a set, the lower is the accuracy of the set. In order to express this idea more precisely we introduce the *accuracy measure*

$$\alpha_R(X) = \frac{card\ \underline{R}X}{card\ \bar{R}X}$$

where $X \neq \emptyset$.

The accuracy measure $\alpha_R(X)$ is intended to capture the degree of completeness of our knowledge about the set X. Obviously $0 \leq \alpha_R(X) \leq 1$, for every R and $X \subseteq U$; if $\alpha_R(X) = 1$ the R-boundary region of X is empty and the set X is R-definable; if $\alpha_R(X) < 1$ the set X has some non-empty R-boundary region and consequently is R-undefinable.

Of course some other measures can also be defined in order to express the degree of inexactness of the set X.

For example, it is possible to use a variety of $\alpha_R(X)$ defined as

$$\rho_R(X) = 1-\alpha_R(X)$$

and referred to as a *R-roughness* of X. Roughness as opposed to accuracy represents the degree of incompleteness of knowledge **R** about the set X.

Example 2.

We shall illustrate the above introduced notions by means of simple examples. Assume the knowledge base and the equivalence relation as in **Example 1** i.e.

$E_1 = \{x_1, x_4, x_8\}$

$E_2 = \{x_2, x_5, x_7\}$

$E_3 = \{x_3\}$

$E_4 = \{x_6\}$

Let us illustrate the above introduced notions for the following three sets:

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

$X_2 = \{x_3, x_5\}$

$X_3 = \{x_3, x_6, x_8\}$.

$\underline{R}X_1 = \emptyset$

$\bar{R}X_1 = E_1 \cup E_2 = \{x_1, x_2, x_4, x_5, x_7, x_8\}$

$BN_R(X_1) = E_1 \cup E_2 = \{x_1, x_2, x_4, x_5, x_7, x_8\}$

$NEG_R(X_1) = E_3 \cup E_4 = \{x_3, x_6\}$

$\alpha_R(X_1) = 0/6 = 0$

$\underline{R}X_2 = E_3 = \{x_3\}$

$\bar{R}X_2 = E_2 \cup E_3 = \{x_2, x_3, x_5, x_7\}$

$BN_R(X_2) = E_2 = \{x_2, x_5, x_7\}$

$NEG_R(X_2) = E_1 \cup E_4 = \{x_1, x_4, x_6, x_8\}$

$\alpha_R(X_2) = 1/4$

$\underline{R}X_3 = E_3 \cup E_4 = \{x_3, x_6\}$

$\bar{R}X_3 = E_1 \cup E_3 \cup E_4 = \{x_1, x_3, x_4, x_6, x_8\}$

$BN_R(X_3) = E_1 = \{x_1, x_4, x_8$

$NEG_R(X_3) = E_2 = \{x_2, x_5, x_7\}$

$\alpha_R(X_3) = 2/5$ ■

A brief intuitive comment on the accuracy measure is in order.

The numerical value of imprecision is not pre assumed, like in probability theory or fuzzy sets - but is calculated on the basis of approximations which are the fundamental concepts used to express imprecision of knowledge. Thus the assumed numerical representation of imprecision is a consequence of limited knowledge (ability to classify objects). As a result we do not require from an agent to assign precise numerical values to express imprecision of his knowledge but instead imprecision is expressed by quantitative concepts (classifications).

The numerical characterization of imprecision will be used to express exactness of concepts and are of great value in many practical applications.

7. Topological Characterization of Imprecision

Besides characterization of rough sets by means of numerical values (accuracy coefficient), one can also introduce an interesting characterization of rough sets employing the notion of the lower and the upper approximation. It turns out then that there are four important and different kinds of rough sets defined as shown below:

a) If $\underline{R}X \neq \emptyset$ and $\bar{R}X \neq U$, then we say that X is *roughly R-definable*

b) If $\underline{R}X = \emptyset$ and $\bar{R}X \neq U$, then we say that X is *internally R-undefinable*

c) If $\underline{R}X \neq \emptyset$ and $\bar{R}X = U$, then we say that X is *externally R-undefinable*

d) If $\underline{R}X = \emptyset$ and $\bar{R}X = U$, then we say that X is *totally R-undefinable*

The intuitive meaning of this classification is the following:

If set X is roughly R-definable, this means that we are able to decide for some elements of U whether they belong to X or $-X$.

If X is internally R-undefinable, this is to mean that we are able to decide whether some elements of U belong to $-X$, but we are unable to decide for any element of U, whether it belongs to X or not.

If X is externally R-undefinable, this means that we are able to decide for some elements of U whether they belong to X, but we are unable to decide, for any element of U whether it belongs to $-X$ or not.

If X is totally R-undefinable, we are unable to decide for any element of U whether it belongs to X nor $-X$.

In other words, set X is roughly definable if there are some objects in the universe which can be positively classified, based on the information available to us as belonging to the set X. This definition also implies that there are some other objects which can be negatively classified without any ambiguity as being outside the set X.

External R-undefinability of a set refers to the situation when positive classification is possible for some objects but is impossible to determine that an object does not belong to X.

The following example will depict the classification of rough sets in more detail:

Example 3.

Let $K = (U, R)$ where $U = \{x_0, \ldots ,x_{10}\}$ and $R \in IND(K)$ with the following equivalence classes:

$E_1 = \{x_0,x_1\}$

$E_2 = \{x_2,x_6,x_9\}$

$E_3 = \{x_3,x_5\}$

$E_4 = \{x_4,x_8\}$

$E_5 = \{x_7,x_{10}\}$

The sets

$X_1 = \{x_0,x_1,x_4,x_8\}$

$Y_1 = \{x_3,x_4,x_5,x_8\}$

$Z_1 = \{x_2,x_3,x_5,x_6,x_9\}$

are examples of R-definable sets.

The sets

$X_2 = \{x_0,x_3,x_4,x_5,x_8,x_{10}\}$

$Y_2 = \{x_1,x_7,x_8,x_{10}\}$

$Z_2 = \{x_2,x_3,x_4,x_8\}$

are examples of roughly R-definable sets. The corresponding approximations, boundaries and accuracies are:

$\underline{R}X_2 = E_3 \cup E_4 = \{x_3,x_4,x_5,x_8\}$

$\bar{R}X_2 = E_1 \cup E_3 \cup E_4 \cup E_5 = \{x_0,x_1,x_3,x_4,x_5,x_7,x_8,x_{10}\}$

$BN_R(X_2) = E_1 \cup E_5 = \{x_0,x_1,x_7,x_{10}\}$

$\alpha_R(X_2) = 4/8 = 1/2$

$\underline{R}Y_2 = E_5 = \{x_7,x_{10}\}$

$\bar{R}Y_2 = E_1 \cup E_4 \cup E_5 = \{x_0,x_1,x_4,x_7,x_8,x_{10}\}$

$BN_R(Y_2) = E_1 \cup E_4 = \{x_4,x_7,x_8,x_{10}\}$

$\alpha_R(Y_2) = 2/6 = 1/3$

$\underline{R}Z_2 = E_4 = \{x_4,x_8\}$

$\bar{R}Z_2 = E_2 \cup E_3 \cup E_4 = \{x_2,x_3,x_4,x_5,x_6,x_8,x_9\}$

$BN_R(Z_2) = E_2 \cup E_3 = \{x_2,x_3,x_5,x_6,x_9\}$

$\alpha_R(Z_2) = 2/7$

The sets

$X_3 = \{x_0,x_1,x_2,x_3,x_4,x_7\}$

$Y_3 = \{x_1,x_2,x_3,x_6,x_8,x_9,x_{10}\}$

$Z_3 = \{x_0,x_2,x_3,x_4,x_8,x_{10}\}$

are examples of externally R-indiscernible sets.

The corresponding approximations, boundaries and accuracies are as follows:

$\underline{R}X_3 = E_1 = \{x_0, x_1\}$

$\bar{R}X_3 = U$

$BN_R(X_3) = E_2 \cup E_3 \cup E_4 \cup E_5 = \{x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$

$\alpha_R(X_3) = 2/11.$

$\underline{R}Y_3 = E_2 = \{x_2, x_6, x_4\}$

$\bar{R}Y_3 = U$

$BN_R(Y_3) = E_1 \cup E_3 \cup E_4 \cup E_5 = \{x_0, x_1, x_3, x_4, x_5, x_7, x_8, x_{10}\}$

$\alpha_R(Y_3) = 3/11$

$\underline{R}Z_3 = E_4 = \{x_4, x_8\}$

$\bar{R}Z_3 = U$

$BN_N\ (Z_3) = E_1 \cup E_2 \cup E_3 \cup E_5 = \{x_0, x_1, x_2, x_3, x_5, x_6, x_7, x_9, x_{10}\}$

$\alpha_R(Z_3) = 2/11.$

The sets

$X_4 = \{x_0, x_2, x_3\}$

$X_4 = \{x_1, x_2, x_4, x_7\}$

$Z_4 = \{x_2, x_3, x_4\}$

are examples of internally R-indiscernible sets.

Below upper approximations of these sets are given :

$\bar{R}X_4 = E_1 \cup E_2 \cup E_3 = \{x_0, x_1, x_2, x_3, x_5, x_6, x_9\}$

$\bar{R}Y_4 = E_1 \cup E_2 \cup E_4 \cup E_7 = \{x_0, x_1, x_2, x_4, x_6, x_7, x_8, x_9, x_{10}\}$

$\bar{R}Z_4 = E_2 \cup E_3 \cup E_4 = \{x_2, x_3, x_4, x_5, x_6, x_8, x_9\}.$

Certainly lower approximations of these sets are empty sets and accuracy is equal to zero.

Below, are given examples of sets which are totally R-indiscernible:

$X_5 = \{x_0, x_2, x_3, x_4, x_7\}$

$Y_5 = \{x_1, x_5, x_6, x_8, x_{10}\}$

$Z_5 = \{x_0, x_2, x_4, x_5, x_7\}$ ■

Next we give an interesting property of the defined topological classification of sets.

Proposition 4

a) Set X is R-definable (roughly R-definable, totally R-undefinable) if and only if so is $-X$;

b) Set X is externally (internally) R-undefinable, if and only if, $-X$ is internally (externally) R-undefinable.

Proof.

a) If X is R-definable, then $\underline{R}X \neq \emptyset$ and $\bar{R}X \neq U$. Because $\underline{R}X \neq \emptyset \equiv$ there exists $x \in X$, such that $[x]_R \subseteq X$ iff $[X]_R \cap -X = \emptyset$ iff $\bar{R}(-X) \neq U$; similarly, $\bar{R}(-X) \neq U$, iff there exists $y \in X$, such that $[y]_R \cap \bar{R}X = \emptyset$ iff $[y]_R \subseteq -\bar{R}(X)$ iff $[y]_R \subseteq \underline{R}(-X)$ iff $\underline{R}(-X) \neq \emptyset$.

The proofs of remaining cases are similar. ■

We would like to conclude this section with the following remark. We have introduced two ways of characterizing rough sets. The first is that of accuracy coefficient and the second - the classification of rough sets given in this section.

The accuracy expresses how large is the boundary region of the set but says nothing about the structure of the boundary, whereas the classification of rough sets gives no information about the size of the boundary region but provides us with some insight as to how the boundary region is structured.

In addition to that observe that there is the relationship between both characterizations of rough sets. First, if the set is internally or totally indiscernible, then the accuracy of this set is zero. Secondly, if the set is externally or totally indiscernible, then the accuracy of its complement is zero. Thus by knowing the accuracy of a set we are still unable to tell exactly its topological structure and conversely the knowledge of the topological structure of the set gives no information about its accuracy.

Therefore, in practical applications of rough sets we combine both kind of information about the boundary region, that is, of the accuracy coefficient as well as the information about the topological classification of the set under consideration.

The introduced classification of rough sets means that there are four fundamental varieties of imprecise categories. Thus imprecision is not a primitive concept since the introduced topological classification of rough sets indicates internal structure of imprecise categories.

8. Approximation of Classifications

Because we are basically intersted in classifications, thus it is interesting to have the notion of approximation of classifications. This is a simple extension of the definition of approximation of sets. Namely if $F = \{X_1, X_2, \ldots ,X_n\}$ is a family of sets, then $\underline{R}F = \{\underline{R}X_1, \underline{R}X_2, \ldots ,\underline{R}X_n\}$ and $\bar{R}F = \{\bar{R}X_1, \bar{R}X_2, \ldots ,\bar{R}X_n\}$, are called the *R-lower* and the *R-upper approximation* of the family *F*.
We will define two measures to describe inexactness of approximate classifications.

The first one is the extension of the measure defined to describe accuracy of approximation of sets, which is the following:

The *accuracy of approximation of F by R* is defined as

$$\alpha_R(F) = \frac{\Sigma \; card \; \underline{R}X_i}{\Sigma \; card \; \bar{R}X_i}$$

The second measure called the *quality of approximation of F by R* is the following:

$$\gamma_R(F) = \frac{\Sigma \; card \; \underline{R}X}{card \; U}$$

The accuracy of classification says what percentage of possible decisions when classifying objects employing the knowledge R is at most correct.The quality of classification says what percentage of object can be correctly classified to classes of F employing knowledge ~~F~~ R. We shall use both measures in what follows.

It is also interesting to describe inexactness of classification using topological means, as in the case of sets. To this end in what follows we shall quote (with slight modifications) interesting results of Grzymala-Busse.(cf. Grzymala-Busse (1988))

Proposition 5

Let $F = \{X_1, X_2, \dots ,X_n\}$, where $n > 1$ be a classification of U and let R be an equivalence relation. If there exists $i \in \{1, 2, \dots ,n\}$ such that $\underline{R}X_i \neq \emptyset$, then for each $j \neq i$ and $j \in \{1, 2, \dots ,n\}$ $\bar{R}X_j \neq U$. (The opposite is not true).

Proof.

If $\underline{R}X_i \neq \emptyset$, then there exists $x \in X$ such that $[x]_R \leq X_i$, which implies $[x]_R \cap X_j = \emptyset$ for each $j \neq i$. This yields $\bar{R}X_j \cap [x]_R = \emptyset$and consequently $\bar{R}X_j \neq U$ for each $j \neq i$.■

Example 4.

Let R be an equivalence relation on $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$, with the equivalence classes as below

$X_1 = \{x_1, x_3, x_5\}$

$X_2 = \{x_2, x_4\}$

$X_3 = \{x_6, x_7, x_8\}$

and let the classification $C = \{Y_1, Y_2, Y_3\}$ be given, where

$Y_1 = \{x_1, x_2, x_4\}$

$Y_2 = \{x_3, x_5, x_8\}$

$Y_3 = \{x_6, x_7\}$.

Because $\underline{R}Y_1 = X_2 = \{x_2, x_4\} \neq \emptyset$, then $\bar{R}Y_2 = X_1 \cup X_3 = \{x_1, x_2, x_4, x_6, x_7\} \neq U$, and $\bar{R}Y_3 = X_3 = \{x_6, x_7, x_8\} \neq U$. ∎

Proposition 6

Let $F = \{X_1, X_2, \ldots, X_n\}$, where $n > 1$ be a classification of U and let R be an equivalence relation. If there exists $i \in \{1, 2, \ldots, n\}$ such that $\bar{R}X_i = U$, then for each $j \neq i$ and $j \in \{1, 2, \ldots, n\}$ $\underline{R}X_j = \emptyset$. (The opposite is not true).

Proof.

If $\bar{R}X_i = U$, then for every $x \in X$ we have $[x]_R \cap X_i \neq \emptyset$. This implies that $[x]_R \subseteq X_j$ does not hold, for each $j \neq i$ and consequently $\underline{R}X_j = \emptyset$ for each $j \neq i$.∎

Example 5.

Assume an equivalence relation as in Example 4, and the following classification $C = \{Z_1, Z_2, Z_3\}$, where

$Z_1 = \{x_1, x_2, x_6\}$

$Z_2 = \{x_3, x_4\}$

$Z_3 = \{x_5, x_7, x_8\}$

Because $\bar{R}Z_1 = X_1 \cup X_2 \cup X_3 = U$. then $\underline{R}Z_2 = \emptyset$ and $\underline{R}Z_3 = \emptyset$. ∎

Direct consequences of the above propositions are the following interesting properties.

Proposition 7

Let $F = \{X_1, X_2, \ldots, X_n\}$, where $n > 1$ be a classification of U and let R be an equivalence relation. If for each $i \in \{1, 2, \ldots, n\}$ $\underline{R}X_i \neq \emptyset$ holds, then $\bar{R}X_i \neq U$ for each $i \in \{1, 2, \ldots, n\}$. (The opposite is not true). ■

Proposition 8

Let $F = \{X_1, X_2, \ldots, X_n\}$, where $n > 1$ be a classification of U and let R be an equivalence relation. If for each $i \in \{1, 2, \ldots, n\}$ $\bar{R}X_i = U$ holds, then $\underline{R}X_i = \emptyset$ for each $i \in \{1, 2, \ldots, n\}$. (The opposite is not true).■

These propositions say that classification with all externally or internally undefinable classes do not exist and they coincide with classifications having all classes totally undefinable.

Thus approximation of sets and approximation of families of sets (in this case classifications) are two different issues, and equivalence classes of approximate classifications can not be arbitrary ~~sets~~, but they are strongly related. Intuitively this means, that if we have positive examples of every category in ~~the~~ approximate classification, then we must have also negative examples for each category. For example if categories are say, colors red, green, blue etc and if we have examples of red, green, blue etc, objects in our knowledge base then we must also also have examples of not red, not green, not blue etc, objects in the knowledge base.

This points out the difference between complement in the case of sets and classifications. Finally this may lead to new concepts of negation in the case of binary and multi-valued logics.

9. Rough Equality of Sets

As we have already indicated the concept of rough set differs essentially from the ordinary concept of the set because for the rough sets we are unable to define uniquely the membership relation and we have two of them instead.

There is another important difference between those concepts namely that of equality of sets. In set theory, two sets are equal if they have exactly the same elements. In our approach we need another concept of equality of sets, namely approximate (rough) equality. Thus two sets can be in equal in set theory but can be approximately equal from our point of view. This is an important feature from practical point of view, for often by using the available knowledge we might be unable to tell whether two sets are equal or not (i.e. they have exactly the same elements) but we can only say that according to our state of knowledge that they have close features which are enough to be assumed approximately equal (because they differ "slightly").

In fact we introduce not one but three kinds of approximate equality of sets.

We are now ready to give the formal definitions.

Let $K = (U, \mathbf{R})$ be a knowledge base, $X, Y \subseteq U$ and $R \in IND(K)$. We say that

a) Sets X and Y are *bottom R-equal* $(X \eqsim_R Y)$ if $\underline{R}X = \underline{R}Y$

b) Sets X and Y are *top R-equal* $(X \simeq_R Y)$ if $\overline{R}X = \overline{R}Y$.

c) Sets X and Y are *R-equal* $(X \approx_R Y)$ if $X \eqsim_R Y$ and $X \simeq_R Y$.

It is easy to see that $\eqsim_R$, $\simeq_R$ and $\approx_R$ are equivalence relations for any indiscernibility relation R.

We can associate the following interpretations, with the above notions of rough equality (For simplicity of notation we shall omit the subscripts.):

a) If $X \eqsim Y$, this means that positive examples of the sets X and Y are the same.

b) If $X \simeq Y$, then the negative examples of sets X and Y are the same.

c) If $X \approx Y$, then both positive and negative examples of sets X and Y are the same.

(By positive or negative examples of a set we mean the elements of the universe belonging to either positive or negative region of the set, respectively.)

Example 6

Let us consider knowledge base $K = (U, \mathbf{R})$ where the universe $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ and the relation $R \in IND(K)$ has the following equivalence classes:

$$E_1=\{x_2,x_3\}, E_2 = \{x_1,x_4,x_5\}$$
$$E_3=\{x_6\}, E_4= \{x_7,x_8\}.$$

For sets $X_1 = \{x_1,x_2,x_3\}$ and $X_2 = \{x_2,x_3,x_7\}$ we have $\underline{R}X_1 = E_1$ and $\underline{R}X_2=E_1$, thus $X_1 \eqsim_R X_2$, i.e. X_1 and X_2 are bottom R-equal.

Sets $Y_1 = \{x_1,x_2,x_7\}$ and $Y_2 = \{x_2,x_3,x_4,x_8\}$ are top R-equal ($Y_1 \simeq_R Y_2$) because $\bar{R}Y_1 = \bar{R}Y_2 = E_1 \cup E_2 \cup E_4$.

Sets $Z_1 = \{x_1,x_2,x_6\}$ and $Z_2 = \{x_3,x_4,x_6\}$ are R-equal ($Z_1 \approx_R Z_2$) because $\underline{R}Z_1 = \underline{R}Z_2 = E_3$ and $\bar{R}Z_1 = \bar{R}Z_2 = E_1 \cup E_2 \cup E_3$.

■

The following exemplary properties of relations $\eqsim_R$, $\simeq_R$ and $\approx_R$ are immediate consequences of the definitions:

Proposition 9.

For any equivalence relation we have the following properties:

1) $X \eqsim Y$ if $X \cap Y \eqsim X$ and $X \cap Y \eqsim Y$

2) $X \simeq Y$ if $X \cup Y \simeq X$ and $X \cup Y \simeq Y$

3) If $X \simeq X'$ and $Y \simeq Y'$, then $X \cup Y \simeq X' \cup Y'$

4) If $X \eqsim X'$ and $Y \eqsim Y'$, then $X \cap Y \eqsim X' \cap Y'$

5) If $X \simeq Y$, then $X \cup - Y \simeq U$

6) If $X \eqsim Y$, then $X \cap -Y \eqsim \emptyset$

7) If $X \subseteq Y$ and $Y \simeq \emptyset$, then $X \simeq \emptyset$

8) If $X \subseteq Y$ and $X \simeq U$, then $Y \simeq U$

9) $X \simeq Y$ if and only if $-X \eqsim -Y$

10) If $X \eqsim \emptyset$ or $Y \eqsim \emptyset$, then $X \cap Y \eqsim \emptyset$

11) If $X \simeq U$ or $Y \simeq U$, then $X \cup Y \simeq U$ ∎

Let us note that if we replace $\eqsim$ by $\simeq$ (or conversely) the above properties are not valid.

Now we are in the position to express lower and upper approximations of sets in terms of rough equalities as stated in the following proposition:

Proposition 10.

For any equivalence relation R

a) $\underline{R}X$ is the intersection of all $Y \subseteq U$ such that $X \eqsim_R Y$.

b) $\bar{R}X$ is the union of all $Y \subseteq U$, such that $X \simeq_R Y$. ∎

We shall conclude this section with the remark that the concept of approximate equality of sets refers to the topological structure of sets being compared but not to the elements they consist of. Thus the sets having quite different elements can be roughly equal in this context. What really matters here is the fact of having the same lower or/and upper approximation by different sets which is a kind of topological characterization. Let us also mention that the definition of rough equality refers to our knowledge about the universe.

Thus in our approach the notion of equality of sets is of relative character. For example two sets can be exactly equal in one approximation space and they can be approximately equal or not equal in another one. This simple observation is confirmed by every ones experience: things are equal or not equal from our point of view ~~depending~~ what we know about

depends on

them.

At the end let us note that equality of precise categories can be expressed employing usual equality of sets, and the above defined approximate equalities are necessary only when speaking about equality of imprecise categories.

Let us also note that if R is an equality relation then, all three above defined rough equalities of sets coincide with "classical" set theoretical equality of sets.

10. Rough Inclusion of Sets

One of the fundamental notion of the set theory is the inclusion relation. Analogous notion can be introduced in rough sets framework. The rough inclusion of sets is defined here in much the same way as the rough equality of sets.

The formal definition of rough inclusion is as follows.

Let $K = (U, \mathbf{R})$ be a knowledge base, $X, Y \subseteq U$, and $R \in IND(K)$. We will say that:

a) Set X is *bottom R-included* in Y ($X \underset{\sim}{\subset}_R Y$) if $\underline{R}X \subseteq \underline{R}Y$.

b) Set X is *top R-included* in Y ($X \overset{\sim}{\subset}_R Y$) if $\overline{R}X \subseteq \overline{R}Y$.

c) Set X is *R-included* in Y ($X \overset{\sim}{\underset{\sim}{\subset}}_R Y$), if $X \overset{\sim}{\subset}_R Y$ and $X \underset{\sim}{\subset}_R Y$.

One can easily see that $\underset{\sim}{\subset}_R$, $\overset{\sim}{\subset}_R$ and $\overset{\sim}{\underset{\sim}{\subset}}_R$ are ordering relations. For simplicity we shall omit the subscript R.

In fact we have, as in the case of rough equality of sets, three rough inclusion relations of sets, called the lower, upper and rough inclusion relation respectively. The intuitive meaning of these inclusions is the following:

a) $X \underset{\sim}{\subset} Y$ means that the positive examples of the set X are also positive examples of the set Y.

b) $X \overset{\sim}{\subset} Y$ means that negative examples of set Y are also the negative examples of the set X.

c) If $X \overset{\sim}{\underset{\sim}{\subset}} Y$, then both a) and b) hold.

It should be quite clear by now that rough inclusion of sets does not imply the inclusion of sets.

The example below depict the concept of rough inclusions.

Example 7

Let us consider the knowledge base as in Example 4. In this knowledge base set $X_1 = \{x_2, x_4, x_6, x_7\}$ is bottom R-included in the set $X_2 = \{x_2, x_3, x_4, x_6\}$, $(X_1 \underset{\sim}{\subset}_R X_2)$ because $\underline{R}X_1 = E_3$ and $\underline{R}X_2 = E_1 \cup E_3$. Set $Y_1 = \{x_2, x_3, x_7\}$ is top R-included in the set $Y_2 = \{x_1, x_2, x_7\}$, $(Y_1 \tilde{\subset}_R Y_2)$ because $\overline{R}Y_1 = E_1 \cup E_4$ and $\overline{R}Y_2 = E_1 \cup E_2 \cup E_4$. Set $Z_1 = \{x_2, x_3\}$ is R-included in $Z_2 = \{x_1, x_2, x_3, x_7\}$.

■

Immediately from the definitions we can derive the following simple properties:

Proposition 11.

1) If $X \subseteq Y$, then $X \underset{\sim}{\subset} Y$, $X \tilde{\subset} Y$ and $X \underset{\sim}{\tilde{\subset}} Y$.
2) If $X \underset{\sim}{\subset} Y$ and $Y \underset{\sim}{\subset} X$, then $X \underset{\sim}{=} Y$.
3) If $X \tilde{\subset} Y$ and $Y \tilde{\subset} X$, then $X \simeq Y$.
4) If $X \underset{\sim}{\tilde{\subset}} Y$ and $Y \underset{\sim}{\tilde{\subset}} X$, then $X \approx Y$.
5) $X \tilde{\subset} Y$ iff $X \cup Y \simeq Y$.
6) $X \underset{\sim}{\subset} Y$ iff $X \cap Y \underset{\sim}{=} Y$.
7) If $X \subseteq Y$, $X \underset{\sim}{=} X'$ and $Y \underset{\sim}{=} Y'$, then $X' \underset{\sim}{\subset} Y'$.
8) If $X \subseteq Y$, $X \simeq X'$ and $Y \simeq Y'$, then $X' \tilde{\subset} Y'$
9) If $X \subseteq Y$, $X \approx X'$ and $Y \approx Y'$ then $X' \underset{\sim}{\tilde{\subset}} Y'$.
10) If $X' \tilde{\subset} X$ and $Y' \tilde{\subset} Y$, then $X' \cup Y' \tilde{\subset} X \cup Y$.
11) If $X' \underset{\sim}{\subset} X$ and $Y' \underset{\sim}{\subset} Y$, then $X' \cap Y' \underset{\sim}{\subset} X \cap Y$.
12) $X \cap Y \underset{\sim}{\subset} X \tilde{\subset} X \cup Y$.
13) If $X \underset{\sim}{\subset} Y$ and $X \underset{\sim}{=} Z$, then $Z \underset{\sim}{\subset} Y$.
14) If $X \tilde{\subset} Y$ and $X \simeq Z$, then $Z \tilde{\subset} Y$.
15) If $X \underset{\sim}{\tilde{\subset}} Y$ and $X \approx Z$, then $Z \underset{\sim}{\tilde{\subset}} Y$.

■

The above properties are not valid if we replace $\underset{\sim}{-}$ by $\simeq$ (or conversely).

It is interesting to compare the properties of rough inclusion with that of ordinary set theoretical inclusion but, we leave this to the interested reader.

The intuitive motivation behind the introduced notions is the following: if *X* and *Y* are imprecise categories, and *X* is top (bottom, roughly) included in *Y*, then *Y* is more a general category then *X*, however three sub cases should be distinguished in this case, inclusion of positive examples of the concept (lower inclusion), - inclusion of negative examples of the concept (upper inclusion) and both, i.e. inclusion of negative and positive examples (rough inclusion).

In the case when *R* is an equality relation, all three inclusions reduce to ordinary inclusion.

Summary

It is interesting to compare the the concept of the rough sets with that of conventional set. Basic properties of roughs sets, like membership of elements, equality and inclusion of sets are related to our knowledge about the universe of discourse expressed by the indiscernibility relation. Consequently whether an element belongs to a set or not is not an objective property of the element but depends upon our knowledge about it. Similarly equality and inclusion of sets are not decidable in absolute sense but depend on what we know about the sets in question. In general all properties of rough sets are not absolute but are related to what we know about them. In this sense the rough set approach could be viewed as a subjective counterpart of the "classical" set theory.

Because we are aiming at formal investigation of categories, the above given definitions and properties can be interpreted in terms of categories. In particular the classification, rough equality rough inclusion of rough sets can be interpreted as classification, rough equality and rough inclusion of rough (approximate) concepts.

Excercises

1.Suppose we are given an equivalence relation R with the following equivalence classes:

$$X_1 = \{x_2, x_4, x_5, x_8\}$$

$$X_2 = \{x_1, x_3\}$$

$$X_3 = \{x_6, x_7, x_9\}$$

Compute the lower and upper approximation, the boundary and the accuracy for sets

$$Y_1 = \{x_1, x_3, x_5\}$$

$$Y_2 = \{x_2, x_3, x_7\}$$

$$Y_3 = \{x_2, x_3, x_5\}$$

$$Y_4 = \{x_1, x_2, x_3, x_6\}.$$

2. Give topological classification for sets Y_1, Y_2, Y_3 and Y_4 considered in Example 1.

3.Give proper (lower or upper) membership relation for elements x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9 and set Y_1 (Y_2, Y_3, Y_4).

4. Check whether the following pairs of sets are roughly equal or not.

$$Z_1 = \{x_2, x_3, x_4, x_5, x_7, x_9\}$$

$$Z_2 = \{x_1, x_2, x_4, x_5\}$$

$$Z_3 = \{x_2, x_3, x_4, x_5, x_8\}$$

$$Z_4 = \{x_1, x_3, x_5, x_8\}$$

$Z_5 = \{x_3, x_6, x_7, x_9\}$

$Z_6 = \{x_1, x_6, x_7, x_9\}$

5. Check whether the following pairs of sets are roughly included or not.

$V_1 = \{x_1, x_2, x_3, x_4, x_5, x_8\}$

$V_2 = \{x_2, x_4, x_5, x_6, x_7, x_8, x_9\}$

$V_3 = \{x_1, x_2, x_3, x_4, x_5\}$

$V_4 = \{x_2, x_4, x_5, x_8\}$

$V_5 = \{x_1, x_2, x_3, x_6, x_7, x_9\}$

$V_6 = \{x_3, x_6, x_7, x_9\}$.

6. Give examples of classification illustrating Proposition 7 and 8.

References

Ballmer, T. and Pinkal, M. (eds.) (1983). Approaching Vagueness. *North-Holland Linguistic Series.* Amsterdam, New York, Oxford.

Black, M. (1937). Vagueness. *The Philosophy of Sciences.* pp. 427-455.

Black, M. (1963). Reasoning with Loose Concepts. *Dialog.* 2, pp. 1-12.

Fine, K. (1975). Vagueness, Truth and Logic. *Synthese.* 30, pp. 265-300.

Grzymala-Busse, J. (1988). Knowledge Acquisition under Uncertainty - a Rough Set Approach. *Journal of Intelligent and Robotics Systems*, 1, pp. 3-16.

Iwiński, T. (1987). Algebraic Approach to Rough Sets. *Bull. Polish Acad. Sci. Math.*, 35, pp. 673-683.

Kohl, M. (1969). Bertrand Russell on Vagueness. *Australian Journal of Philosophy.* 147, pp. 31-41.

Nieminen, J. (1988). Rough Tolerance Equality and Tolerance Black Boxes. *Fundamenta Informaticae* (to appear).

Novotny, M. and Pawlak, Z. (1985a). Black Box Analysis and Rough Top Equality. *Bull. Polish Acad. Sci. Math.*, 33, pp. 105-113.

Novotny, M. and Pawlak, Z. (1985b). Characterization of Rough Top Equalities and Rough Bottom Equalities. *Bull. Polish. Acad. Sci. Math.*, 33, pp. 91-97.

Novotny, M. and Pawlak, Z. (1985c). On Rough Equalities. *Bull. Polish. Acad. Sci. Math.*, 33, pp. 99-104.

Obtułowicz, A. (1988). Rough Sets and Heyting Algebra Valued Sets. *Bull. Polish Acad. Sci. Math.*, 35, pp. 667-673.

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

Pawlak, Z. (1989), Indiscernibility, Partitions and Rough Sets. *Commemorative Volume on Theoretical Computer Science -in Honor of Prof. Siromony*, World Scientific Publishing Comp., Co., (Ed., R.Narasimhan) Singapore (to appear).

Pomykala, J. and Pomykala, J. A. (1988). The Stone Algebra of Rough Sets. *Bull. Polish. Acad. Sci. Math.*, 36, pp. 495-508.

Russel, B. (1923). Vagueness. *Australian Journal of Philoso-*

phy, 1, pp. 84-92.

Skowron, A. (1988). On the Topology in Information Systems. *Bull. Polish. Acad. Sci. Math.*, 36, pp. 477-480.

Skowron, A. (1989). The Relationship between Rough Set Theory and Evidence Theory *Bull. Polish. Acad. Sci. Math.*, 37, pp. 87-90.

Wiweger, A. (1988). Topology and Rough Sets *Bull. Polish. Acad. Sci. Math.*, (to appear).

3. REDUCTION OF KNOWLEDGE

1. Introduction

A fundamental problem we are going to address in this section is whether the whole knowledge is always necessary to define some categories available in the knowledge considered. This problem arises in many practical applications and will be referred to as knowledge reduction. Its significance will be clearly seen in the second part of the book, where various applications are discussed.

It is interesting to note that the problem of reduction of knowledge is strictly related to the general concept of independence discussed in mathematics as formulated by Marczewski (cf. Marczewski (1958) see also an overview paper by Głazek (1979)), however results of independence theory can not be used in our approach, for they do not address problems which are relevant to issues connected with knowledge reduction. Some mathematical problems connected with knowledge reduction are considered in the paper by Łoś (cf. Łoś (1979)).

2. Reduct and Core of Knowledge

In reduction of knowledge basic role is played, in the proposed approach, by two fundamental concepts - a reduct and the core. Intuitively a reduct of knowledge is its essential part, which suffices to define all basic concepts occurring in the considered knowledge, whereas the core is in a certain sense its most important part. Examples given in the second part of the book will make these ideas more clear, and in this section we confine ourselves to formal definitions of areduct and the core, before however we need some auxiliary notions, which are given next.

Let $\mathbf{R}$ a family of equivalence relations and let $R \in \mathbf{R}$. We will say that R is *dispensable* in $\mathbf{R}$ if $IND(\mathbf{R})=IND(\mathbf{R}-\{R\})$; otherwise R *indispensable* in $\mathbf{R}$.

The family $\mathbf{R}$ is *independent* if each $R \in \mathbf{R}$ is indispensable in $\mathbf{R}$.

Remark. The defined concept of independence is a special case of independence introduced by Marczewski (cf. Marczewski (1958), see also an overview paper by Głazek (1975)).

Proposition 1

If **R** is independent and $\mathbf{P} \subseteq \mathbf{R}$, then **P** is also independent.

Proof. The proof is by contradiction. Suppose $\mathbf{Q} \subset \mathbf{P}$ and **Q** is dependent, then there exists $\mathbf{S} \subset \mathbf{Q}$ such that $IND(\mathbf{S}) = IND(\mathbf{Q})$ which implies $IND(\mathbf{S} \cup (\mathbf{P} - \mathbf{Q})) = IND(\mathbf{P})$ and $\mathbf{S} \cup (\mathbf{P} - \mathbf{Q}) \subset \mathbf{P}$. Hence **P** is dependent, which is a contradiction.■

$\mathbf{Q} \subseteq \mathbf{P}$ is a *reduct* of **P** if **Q** is independent and $IND(\mathbf{Q}) = IND(\mathbf{P})$.

Obviously **P** may have many reducts.

Set of all indispensable relations in **P** will be called the *core* of **P**, and will be denoted *CORE* (**P**).

The following is an important property establishing the relationship between the core and reducts.

Proposition 2

$$CORE(\mathbf{P}) = \bigcap RED(\mathbf{P})$$

where $RED(\mathbf{P})$ is the family of all reducts of **P**.

Proof.

If **Q** is a reduct of **P** and $R \in \mathbf{P} - \mathbf{Q}$, then $IND(\mathbf{P}) = IND(\mathbf{Q})$, $\mathbf{Q} \subseteq \mathbf{P} - \{R\} \subseteq \mathbf{P}$. Note that, if **P**, **Q**, **R** are sets of equivalence relations, $IND(\mathbf{P}) = IND(\mathbf{Q})$, and $\mathbf{Q} \subseteq \mathbf{R} \subseteq \mathbf{P}$, then $IND(\mathbf{Q}) = IND(\mathbf{R})$. Assuming that $\mathbf{R} = \mathbf{P} - \{R\}$ we conclude that R is superfluous, i.e. $R \notin CORE(\mathbf{P})$, and $CORE(\mathbf{P}) \subseteq \bigcap\{\mathbf{Q}: \mathbf{Q} \in RED(\mathbf{P})\}$.

Suppose $R \notin CORE(\mathbf{P})$, i.e. R is superfluous in **P**. That means $IND(\mathbf{P}) = IND(\mathbf{P} - \{R\})$, which implies that there exists independent subset $\mathbf{S} \subseteq \mathbf{P} - \{R\}$, such that $IND(\mathbf{S}) = IND(\mathbf{P})$.

Obviously **S** is a reduct of **P** and $R \notin$ **S**. This shows that $CORE(\mathbf{P}) \supseteq \bigcap\{\mathbf{Q} : \mathbf{Q} \in RED(\mathbf{P})\}$.■

The use of the concept of the core is twofold. First, it can be used as a basis for computation of all reducts, for the core is included in every reduct, and its computation is straightforward. Secondly, the core can be interpreted as the set of most characteristic part of knowledge which can not be eliminated when reducing the knowledge.

Example 1

Suppose we are given a family $\mathbf{R} = \{P, Q, R\}$ of three equivalence relations P, Q and R with the following equivalence classes:

$$U/P = \{\{x_1, x_4, x_5\}, \{x_2, x_8\}, \{x_3\}, \{x_6, x_7\}\}$$

$$U/Q = \{\{x_1, x_3, x_5\}, \{x_6\}, \{x_2, x_4, x_7, x_8\}\}$$

$$U/R = \{\{x_1, x_5\}, \{x_2, x_7, x_8\}, \{x_3, x_4\}\}$$

Thus the relation $IND(\mathbf{R})$ has the equivalence classes

$$U/IND(\mathbf{R}) = \{\{x_1, x_5\}, \{x_2, x_8\}, \{x_3\}, \{x_4\}, \{x_6\}, \{x_7\}\}.$$

The relation P is indispensable in **R**, since

$$U/IND(\mathbf{R} - \{P\}) = \{\{x_1, x_5\}, \{x_2, x_7, x_8\}, \{x_3\}, \{x_4\},$$
$$\{x_6\}\} \neq U/IND(\mathbf{R}).$$

For relation Q we have

$$U/IND(\mathbf{R} - \{Q\}) = \{\{x_1, x_5\}, \{x_2, x_8\}, \{x_3\}, \{x_4\}, \{x_6\},$$
$\{x_7\}\} = IND(\mathbf{R})$, thus the relation Q is dispensable in **R**.

Similarly for relation R

$$U/IND(\mathbf{R} - \{R\}) = \{\{x_1, x_5\}, \{x_2, x_8\}, \{x_3\}, \{x_4\},$$

$\{x_6\}, \{x_7\}\} = U/IND(\mathbf{R})$, hence the relation R is also dispensable in $\mathbf{R}$.

That means that the classification defined by the set of three equivalence relations P, Q and R is the same as the classification defined by relation P and Q or P and R. In order to find reducts of the set $\mathbf{R} = \{P, Q, R\}$ we have to check whether pairs of sets P, Q and P, R are independent or not. Because $U/IND(\{P, Q\}) \neq U/IND(Q)$ and $U/IND(\{P, Q\}) \neq U/IND(P)$, hence the relation P and Q are independent and consequently $\{P, Q\}$ is a reduct of $\mathbf{R}$. Proceeding in the same way we find that $\{P, R\}$ is also a reduct of $\mathbf{R}$.

Thus are two reducts of the family $\mathbf{R}$, namely $\{P, Q\}$ and $\{P, R\}$ and $\{P, Q\} \cap \{P, R\} = P$ is the core of $\mathbf{R}$.■

3. Relative Reduct and Relative Core of Knowledge

The concepts of reduct and core defined in the previous section are of limited value when speaking about applications. Therefore in this section we will give a generalization of these concepts needed for further considerations.

To this end we need first to define a notion of a positive region of a classification with respect to another classification. For mathematical reasons we will use equivalence relations instead of classifications, but as we mentioned before these two concepts are mutually exchangeable.

Let P and Q be equivalence relations over U.

By *P-positive region* of Q, denoted $POS_P(Q)$ we understand the set

$$POS_P(Q) = \bigcup_{X \in U/Q} \underline{P}X$$

The positive region of the classification U/Q with respect to U/P is the set of all objects of the universe which can be properly classified to classes of U/Q employing knowledge expressed by the classification U/P.

Now we are able to give definitions of generalized concepts considered in the previous section.

Let **P** and **Q** be families of equivalence relations over *U*.

We say that $R \in \mathbf{P}$ is **Q**-*dispensable in* **P**, if

$$POS_{IND(\mathbf{P})}(IND(\mathbf{Q})) \quad = \quad POS_{IND(\mathbf{P}-\{R\})}(IND(\mathbf{Q}))$$

otherwise *R* is **Q**-*indispensable in* **P**.

If every *R* in **P** is **Q**-*indispensable* we will say that **P** is **Q**-*independent* (or **P** is *independent with respect to* **Q**).

Family $\mathbf{S} \subseteq \mathbf{P}$ will be called a **Q**-*reduct* of **P**, if **S** is **Q**-independent subfamily of **P** and $POS_{\mathbf{S}}(\mathbf{Q}) = POS_{\mathbf{P}}(\mathbf{Q})$.

The set of all **Q**-indispensable primitive relations in **P** will be called the **Q**-*core* of **P**, and will be denoted as $CORE_{\mathbf{Q}}(\mathbf{P})$.

It is easily seen that if **P** = **Q** we get the definitions introduced in the previous section.

The following proposition is a counterpart of **Proposition 2.**

Proposition 3

$$CORE_{\mathbf{Q}}(\mathbf{P}) = \bigcap RED_{\mathbf{Q}}(\mathbf{P})$$

where $RED_{\mathbf{Q}}(\mathbf{P})$ is the family of all **Q**-reducts of **P**. ∎

Proof of this proposition is similar to that of **Proposition 2.**

Example 2.

This example will serve as an illustration of ideas considered in this section.

Consider family **R** = {*P*, *Q*, *R*} of equivalence relations having the following equivalence classes:

$$U/P = \{\{x_1, x_3, x_4, x_5, x_6, x_7\}, \{x_2, x_8\}\}$$

$$U/Q = \{\{x_1, x_3, x_4, x_5\}, \{x_2, x_6, x_7, x_8\}\}$$

$$U/R = \{\{x_1, x_5, x_6\}, \{x_2, x_7, x_8\}, \{x_3, x_4\}\}$$

The family **R** induces classification

$$U/IND(\mathbf{R}) = \{\{x_1, x_5\}, \{x_3, x_4\}, \{x_2, x_8\}, \{x_6\}, \{x_7\}\}.$$

Moreover assume that the equivalence relation S is given with the equivalence classes

$$U/S = \{\{x_1, x_5, x_6\}, \{x_3, x_4\}, \{x_2, x_7\}, \{x_8\}\}$$

The positive region of S with respect to **R** is the union of all equivalence classes of $U/IND(\mathbf{R})$ which are included in some equivalence classes of U/S, i.e. the set

$$POS_{\mathbf{R}}(S) = \{x_1, x_3, x_4, x_5, x_6, x_7\}.$$

In order to compute the core and reducts of **R** with respect to S we have first to find out whether the family **R** is S dependent or not. Accordingly to definitions given in this section we have to compute first whether P, Q and R are dispensable or not with respect to S (S-dispensable). Removing P we get

$$U/IND(\mathbf{R} - P) = \{\{x_1, x_5\}, \{x_3, x_4\}, \{x_2, x_7, x_8\}, \{x_6\}\}.$$

Because

$$POS_{(\mathbf{R} - \{P\})}(S) = \{x_1, x_3, x_4, x_5, \{x_6\} \neq POS_{\mathbf{R}}(S)$$

the P is S-indispensable in **R**.

Dropping now Q from **R** we get

$$U/IND(\mathbf{R} - Q) = \{\{x_1, x_5, x_6\}, \{ x_3, x_4\}, \{x_2, x_8\}, \{x_7\}\}$$

which yields the positive region

$$POS_{(\mathbf{R} - \{Q\})}(S) = \{x_1, x_3, x_4, x_5, x_6, x_7\} = POS_{\mathbf{R}}(S)$$

hence Q is S-dispensable in **R**.

Finally omitting R in **R** we obtain

$$U/IND(\mathbf{R} - R) = \{\{x_1, x_3, x_4, x_5\}, \{x_2, x_8\}, \{x_6, x_7\}\}$$

and the positive region is

$$POS_{(\mathbf{R} - \{R\})}(S) = \emptyset, \neq POS_{\mathbf{R}}(S),$$

which means that R is S-indispensable in **R**.

Thus the S-core of **R** is the set $\{P, R\}$, which is also the S-reduct of **R**.■

Let us briefly comment the above defined notions.

Set $POS_{\mathbf{P}}(\mathbf{Q})$ is the set of all objects which can be classified to elementary categories of knowledge **Q**, employing knowledge **P**.

Knowledge **P** is **Q**-independent if the whole knowledge **P** is necessary to classify objects to elementary categories of knowledge **P**.

The **Q**-core knowledge of **P** is the most essential part of knowledge **P**, which cannot be eliminated without disturbing the ability to classify objects to elementary categories of **Q**.

The **Q**-reduct of knowledge **P** is minimal subset of knowledge **P**, which provides the same classification of objects to elementary categories of knowledge **Q** as the whole knowledge **P**. Let us observe that knowledge **P** can have more than one reduct.

Knowledge **P** with only one **Q**-reduct is in a sense deterministic, i.e. there is only one way of using elementary categories of knowledge **P** when classifying objects to elementary categories of knowledge **Q**. In the case of nondeterministic knowledge i.e. if knowledge **P** has many **Q**-reducts, there are in general many ways of using elementary categories of **P** when classifying objects to elementary categories of **Q**. This nondeterminism is particularly strong if the core knowledge is void. Hence nondeterminism introduces synonymy to the knowledge, which in some cases may be a drawback.

More about the discussed approach can be found in Grzymała-Busse (1986), Novotny (1988, 1989a,b, 1990) and Rauszer (1985, 1986, 1988, 1990).

4. Reduction of Categories

Basic categories are pieces of knowledge, which can be considered as "building blocks" of concepts. Every concept in the knowledge base can be only expressed (exactly or approximately) in terms of basic categories. On the other hand every basic category is "built up" (is an intersection) of some elementary categories. Thus the question arises, whether all the elementary categories are necessary to define the basic categories in question.

From mathematical point of view this problem is similar to that of reducing knowledge i.e. elimination of equivalence relations which are superfluous to define all basic categories in knowledge P.

The problem can be formulated precisely as follows.

Let $F = \{X_1, \ldots, X_n\}$, be a family of sets such that $X_i \subseteq U$.

We say that X_i is *dispensable* in F, if $\bigcap (F - \{X_i\}) = \bigcap F$; otherwise the set X_i is *indispensable* in F.

The family $G \subseteq F$ is *independent* if all of its components are indispensable; otherwise G is *dependent*.

The family $H \subseteq F$ is a *reduct* of F, if H is independent and $\bigcap H = \bigcap F$.

The family of all indispensable sets in F will be called the *core* of F, denoted $CORE(F)$. The introduced above definitions are counterparts of definitions given in section 2, with the only difference that instead of relations, we deal now with sets.

Now we have the following counterparts of Proposition 2.

Proposition 4.

$$CORE(F) = \bigcap RED(F)$$

where $RED(F)$ is the family of all reducts of F. ■

Example 3

Suppose we are given family of three sets $F = \{X, Y, Z\}$, where

$X = \{x_1, x_3, x_8\}$

$Y = \{x_1, x_3, x_4, x_5, x_6\}$

$Z = \{x_1, x_3, x_4, x_6, x_7\}$

Hence $\bigcap F = X \cap Y \cap Z = \{x_1, x_3\}$.

Because

$\bigcap(F - \{X\}) = Y \cap Z = \{x_1, x_3, x_4, x_6\}$

$\bigcap(F - \{Y\}) = X \cap Z = \{x_1, x_3\}$

$\bigcap(F - \{Z\}) = X \cap Y = \{x_1, x_3\}$

sets Y and Z are dispensable in the family F, hence the family F is dependent, set X is the core of F, families $\{X, Y\}$ and $\{X, Z\}$ are reducts of F, and $\{X, Y\} \cap \{X, Z\} = X$. ■

Remark

The problem mentioned above is somewhat similar to so called "minimal cover" (cf for example **M.R. Garey** and D.S.Johnson, Computers and Intractability A Guide to the Theory of NP-Completeness, *W.H.Freeman and Company 1979*, San Francisco).■

We will also need in further chapters a method to eliminate superfluous categories from categories which are union of some categories. The idea will be of particular We will also need in further chapters a method to eliminate superfluous categories from categories which are union of some categories. The idea will be of particular interest, in the context of minimization of instruction in decision tables, considered in Chapter 6. The problem can be formulated in a similar way as the previous one, with the exception that now instead of intersection of sets we will need union of sets, as formulated below.

Let $F=\{X_1,\dots,X_n\}$, be a family of sets such that $X_i \subseteq U$.

We say that X_i is *dispensable* in $\bigcup F$, if $\bigcup(F - \{X_i\}) = \bigcup F$; otherwise the set X_i is *indispensable* in $\bigcup F$.

The family $G \subseteq F$ is *independent* if all of its components are indispensable; otherwise G is *dependent*.

The family $H \subseteq F$ is a *reduct* of $\bigcup F$, if H is independent and $\bigcup H = \bigcup F$.

Example 4.

This example will serve as illustration of the idea discussed above. Let $F = \{X, Y, Z, T\}$, where

$$X = \{x_1, x_3, x_8\}$$

$$Y = \{x_1, x_3, x_4, x_5, x_6\}$$

$$Z = \{x_1, x_3, x_4, x_6, x_7\}$$

$$T = \{x_1, x_3, x_5, x_7\}$$

Obviously $\bigcup F = X \cup Y \cup Z \cup T = \{x_1, x_2, x_3, x_4, x_4, x_5, x_6, x_7, x_8\}$. Because we have

$$\bigcup (F - \{X\}) = \bigcup \{Y, Z, T\} = \{x_1, x_2, x_3, x_4, x_4, x_5, x_6, x_7\} \neq \bigcup F$$

$$\bigcup (F - \{Y\}) = \bigcup \{X, Z, T\} = \{x_1, x_2, x_3, x_4, x_4, x_5, x_6, x_7, x_8\} = \bigcup F$$

$$\bigcup F(- \{Z\}) = \bigcup \{X, Y, T\} = \{x_1, x_2, x_3, x_4, x_4, x_5, x_6, x_7, x_8\} = \bigcup F$$

$$\bigcup (F - \{T\}) = \bigcup \{X, Y, Z\} = \{x_1, x_2, x_3, x_4, x_4, x_5, x_6, x_7, x_8\} = \bigcup F$$

thus the only indispensable set in the family F is the set X, and remaining sets Y, Z and T are dispensable in the family. Hence the following sets are reducts of F: $\{X, Y, Z\}$,

$\{X, Y, T\}$, $\{X, Z, T\}$. That means that the concept $\bigcup F = X \cup Y \cup Z \cup T$, which is the union of some concepts X, Y, Z and T can be simplified and represented as union of smaller numbers of concepts.■

5. Relative Reduct and Core of Categories

Similarly as in section 3, we can generalize the concept of the reduct, and the core, with respect to a specific set. This will be needed in further chapters, in order to investigate of how much knowledge is necessary to preserve some properties of categories.

Suppose we are given a family $F = \{X_1, \ldots, X_n\}$, $X_i \subseteq U$ and a subset $Y \subseteq U$, such that $\bigcap F \subseteq Y$.

We say that X_i is *Y-dispensable* in F, if $\bigcap (F - \{X_i\}) \subseteq Y$; otherwise the set X_i is *Y-indispensable* in F.

The family $G \subseteq F$ is *Y-independent* if all of its components are Y-indispensable; otherwise G is *Y-dependent*.

The family $H \subseteq F$ is a *Y-reduct* of F, if H is Y-independent and $\bigcap H \subseteq Y$.

The family of all Y-indispensable sets in F will be called the *Y-core* of F. We will also say that a Y-reduct (Y-core) is relative reduct (core) with respect to Y.

The counterpart of **Proposition 3** is also valid in the present framework as shown below:

Proposition 5.

$$CORE_Y(F) = \bigcap RED_Y(F)$$

where $RED_Y(F)$ is the family of all Y-reducts of F. ■

Thus superfluous elementary categories can be eliminated from the basic categories in a similar way as the equivalence relations discussed in previous section in connection with reduction of relations.

Let us also note that if $Y = \bigcap F$, we obtain the case considered at the beginning of this section.

Example 5

Consider again sets as in Example 3, and assume that $T = \{X_1, X_3\}$. In this case the T-reducts and T-core coincide with that obtained in Example 3.

Consider another example of the family of three sets $F = \{X, Y, Z\}$, where

$$X = \{x_1, x_3, x_8\}$$

$$Y = \{x_1, x_3, x_4, x_5, x_6\}$$

$$Z = \{x_1, x_3, x_4, x_6, x_7\}.$$

and $\bigcap F = X \cap Y \cap Z = \{x_1, x_3\}$.

Let $T = \{x_1, x_3, x_8\} \supseteq \bigcap F$. Now we are able to see whether sets X, Y, Z are T-dispensable in the family F or not. To this end we have to compute the following:

$$\bigcap(F - \{X\}) = Y \cap Z = \{x_1, x_3, x_4, x_6\}$$

$$\bigcap(F - \{Y\}) = X \cap Z = \{x_1, x_3, x_4\}$$

$$\bigcap(F - \{Z\}) = X \cap Y = \{x_1, x_3, x_4\}.$$

Hence the sets X, Y and Z are T-indispensable, thus the family F is T-independent, the T-core of F is the empty set, and there are three T-reducts of the family F, $\{X, Y\}$, $\{X, Z\}$ and $\{Y, Z\}$.

■

Summary

Reducing of knowledge consists in removing of superfluous partitions (equivalence relations) or/and superfluous basic categories in the knowledge base in such a way that the set of elementary categories in the knowledge base is preserved. This procedure enable us to ~~the~~ eliminate all unnecessary knowledge from the knowledge base, preserving only this part of knowledge which is really useful.

Exercises

1. Prove Proposition 3, 4 and 5.

2. The family of all indispensable sets in $\bigcup F$ will be called the *core* of $\bigcup F$, denoted $CORE(F)$. Is the counterpart of Proposition 4, i.e.the following proposition valid?

$$CORE(F) = \bigcap RED(F),$$

where $RED(F)$ is the family of all reducts of F.

3. Check whether the following family of partitions (or corresponding equivalence relations) is independent or not.

$$F_1 = \{\{x_2, x_4\}, \{x_5, x_1, x_3\}, \{x_6, x_8\}, \{x_7, x_9\}\}$$

$$F_2 = \{\{x_1, x_3, x_4\}, \{x_2, x_5\}, \{x_6, x_7, x_8, x_9\}\}$$

$$F_3 = \{\{x_2, x_3, x_5\}, \{x_1, x_4, x_6\}, \{x_7, x_8, x_9\}\}$$

Compute the core and reducts.

4. Check whether

a) the family $\{F_1, F_2\}$ is F_3-independent

b) the family $\{F_1, F_3\}$ is F_2-independent

c) the family $\{F_2, F_3\}$ is F_1-independent

Compute corresponding cores and reducts.

5. Check whether the following family of sets is independent or not.

$$X_1 = \{x_1, x_2, x_5, x_6\}$$

$$X_2 = \{x_2, x_3, x_5\}$$

$$X_3 = \{x_1, x_3, x_5, x_6\}$$

$$X_4 = \{x_1, x_5, x_6\}$$

Compute the core and reducts.

6. Check whether

a) the family $\{X_1, X_2, X_2\}$ is X_4-independent

b) the family $\{X_2, X_3, X_4\}$ is X_1-independent

c) the family $\{X_1, X_3, X_4\}$ is X_2-independent

7. Check whether the union of sets X_1, X_2, X_3, X_4 is independent or not.

References

Głazek, K. (1979). Some Old and New Problems in the Independence Theory. *Colloquium Mathematicum*, 17, pp. 127-189.

Grzymała-Busse, J. (1986). On the Reduction of Knowledge Representation Systems. *Proc. of the 6th International Workshop on Expert Systems and their Applications.* Avignon, France, April 28-30, 1, pp. 463-478.

Łoś, J. (1979). Characteristic Sets of a System of Equivalence Relations. *Colloquium Mathematicum* XLII, pp. 291-293.

Marczewski, E. (1958). A general Scheme of Independence in Mathematics. *BAPS*, pp. 731-736.

Novotny, M. & Pawlak, Z. (1988a). Independence of Attributes. *Bull. Polish Acad. Sci. Math.*, 36, pp. 459-465.

Novotny, M. & Pawlak, Z. (1988b). Partial Dependency of Attributes *Bull. Polish Acad. Sci. Math.*, 36, pp. 453-458.

Novotny, M. & Pawlak, Z. (1989). Algebraic Theory of Independence in Information Systems. *Institute of Mathematics Czechoslovak Academy of Sciences Report*, 51.

Novotny, M. (1990). On Superreducts. *Bull. Polish Acad. Sci. Math.*, (to appear).

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

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. (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).

4. DEPENDENCIES IN KNOWLEDGE BASE

1. Introduction

Theorizing, besides classification, is the second most important aspect when drawing inferences about the world. Essentially, developing theories is based on discovering inference rules of the form "if ... then ... ". (Sometimes the rules can describe causal relationships). In our philosophy this can be formulated as how from a given knowledge another knowledge can be induced.

More precisely, knowledge **Q** is *derivable* from knowledge **P**, if all elementary categories of **Q** can be defined in terms of some elementary categories of knowledge **P**. If **Q** is derivable from **P** we will also say that **Q** *depends* on **P** and can be written $\mathbf{P} \Rightarrow \mathbf{Q}$. This problem will be considered in more detail in Chapter 7 in logical setting. In this chapter we are going to show the semantic aspects of dependency.

More about the notion of dependency considered in this chapter can be found in Novotny et al. (1988, 1989, 1990) and Pawlak (1985)).

The articles of Buszkowski et al. (1986) and Rauszer (1984, 1985a,b,c, 1986, 1987, 1988) investigate the relationship between functional dependencies in relational databases and those considered here.

2. Dependency of Knowledge

Formally the dependency can be defined as shown below: Let $K = (U, \mathbf{R})$ be a knowledge base and let $\mathbf{P}, \mathbf{Q} \subseteq \mathbf{R}$.

1) Knowledge **Q** *depends* on knowledge **P** if $IND(\mathbf{P}) \subseteq IND(\mathbf{Q})$.

2) Knowledge **P** and **Q** are *equivalent*, denoted as $\mathbf{P} \equiv \mathbf{Q}$, if $\mathbf{P} \Rightarrow \mathbf{Q}$ and $\mathbf{Q} \Rightarrow \mathbf{P}$.

3) Knowledge **P** and **Q** are *independent*, denoted as $\mathbf{P} \not\equiv \mathbf{Q}$ if neither $\mathbf{P} \Rightarrow \mathbf{Q}$ nor $\mathbf{Q} \Rightarrow \mathbf{P}$ hold.

Obviously $P \equiv Q$, if and only if $IND(P) = IND(Q)$.

The following example will demonstrate the definition of dependency.

Example 1.

Suppose we are given knowledge P and Q with the following partitions: $U/P = \{\{1,5\}, \{2,8\}, \{3\}, \{4\}, \{6\}, \{7\}\}$ and $U/Q = \{\{1,5\}, \{2,7,8\}, \{3,4,6\}\}$. Hence $IND(P) \subseteq IND(Q)$ and consequently $P \Rightarrow Q$.■

It is easy to show by simple computation the following properties.

Proposition 1.

The following conditions are equivalent:

1) $P \Rightarrow Q$
2) $IND(P \cup Q) = IND(P)$
3) $POS_P(Q) = U$
4) $\underline{P}X = X$ for all $X \in U/Q$ ■

The **Proposition 1** demonstrates that if Q depends on P, then knowledge Q is superfluous within the knowledge base in the sense that the knowledge $P \cup Q$ and P provide the same characterization of objects.

The following are also important properties of dependencies:

Proposition 2.

If P is a reduct of Q, then $P \Rightarrow Q - P$ and $IND(P)=IND(Q)$.

■

Proposition 3.

a) If P is dependent, then there exists a subset $Q \subseteq P$ $(P \neq Q)$ such that Q is a reduct of P.

b) If $P \subseteq Q$ and P is independent, then all basic relations in P are pair wise independent.

c) If $\mathbf{P} \subseteq \mathbf{Q}$ and $\mathbf{P}$ is independent, then every subset $\mathbf{R}$ of $\mathbf{P}$ is independent. ■

The properties given below characterize the dependency in more detail.

Proposition 4.

1) If $\mathbf{P} \Rightarrow \mathbf{Q}$, and $\mathbf{P}' \supset \mathbf{P}$, then $\mathbf{P}' \Rightarrow \mathbf{Q}$
2) If $\mathbf{P} \Rightarrow \mathbf{Q}$, and $\mathbf{Q}' \subset \mathbf{Q}$, then $\mathbf{P} \Rightarrow \mathbf{Q}'$.
3) $\mathbf{P} \Rightarrow \mathbf{Q}$ and $\mathbf{Q} \Rightarrow \mathbf{R}$ imply $\mathbf{P} \Rightarrow \mathbf{R}$
4) $\mathbf{P} \Rightarrow \mathbf{R}$ and $\mathbf{Q} \Rightarrow \mathbf{R}$ imply $\mathbf{P} \cup \mathbf{Q} \Rightarrow \mathbf{R}$
5) $\mathbf{P} \Rightarrow \mathbf{R} \cup \mathbf{Q}$ imply $\mathbf{P} \Rightarrow \mathbf{R}$ and $\mathbf{P} \Rightarrow \mathbf{Q}$
6) $\mathbf{P} \Rightarrow \mathbf{Q}$ and $\mathbf{Q} \cup \mathbf{R} \Rightarrow \mathbf{T}$ imply $\mathbf{P} \cup \mathbf{R} \Rightarrow \mathbf{T}$
7) $\mathbf{P} \Rightarrow \mathbf{Q}$ and $\mathbf{R} \Rightarrow \mathbf{T}$ imply $\mathbf{P} \cup \mathbf{R} \Rightarrow \mathbf{Q} \cup \mathbf{T}$. ■

3. Partial Dependency of Knowledge

The derivation (dependency) can also be partial, which means that only part of knowledge Q is derivable from knowledge P. The partial derivability can be defined using the notion of the positive region of knowledge.

We will now define the partial derivability formally.

Let $K = (U,\mathbf{R})$ be knowledge base and $\mathbf{P},\mathbf{Q} \subseteq \mathbf{R}$. We say that knowledge $\mathbf{Q}$ is *derivable in a degree* k $(0 \le k \le 1)$ from knowledge $\mathbf{P}$, symbolically $\mathbf{P} \Rightarrow_k \mathbf{Q}$, if

$$k = \gamma_{\mathbf{P}}(\mathbf{Q}) = \frac{card\ POS_{\mathbf{P}}(\mathbf{Q})}{card\ (U)}$$

where *card* denotes cardinality of the set.

If $k = 1$ we will say that $\mathbf{Q}$ *is totally derivable from* $\mathbf{P}$; if $0 < k < 1$, we say that $\mathbf{Q}$ *is roughly (partially) derivable from* $\mathbf{P}$, and if $k = 0$ we say that $\mathbf{Q}$ is *totally nonderivable from* $\mathbf{P}$. If $\mathbf{P} \Rightarrow_1 \mathbf{Q}$ we shall also write $\mathbf{P} \Rightarrow \mathbf{Q}$.

The above described ideas can also be interpreted as an ability to classify objects. More precisely, if $k = 1$, then all elements of the universe can be classified to elementary categories of U/Q by using knowledge P. If $k \neq 1$, only those element of the universe which belong to the positive region can be classified to categories of knowledge Q, employing knowledge P. In particular, if $k = 0$, none of the elements of the universe can be classified using knowledge P - to elementary categories of knowledge Q.

More precisely, from the definition of dependency follows, if $P \Rightarrow_k Q$ then the positive region of the partition U/Q induced by Q covers $k*100$ percent of all objects in the knowledge base. On the other hand, only those objects belonging to positive region of the partition can be uniquely classified. This means that $k*100$ percent of objects can be classified into blocks of partition U/Q employing knowledge P.

Thus the coefficient $\gamma_P(Q)$ can be understood as a degree of dependency between Q and P. In other words if we restrict the set of objects in the knowledge base to the set $POS_P Q$, we would obtain the knowledge base in which $P \Rightarrow Q$ is a total dependency.

Of course one could use another measure of rough dependency but the one assumed here seems to be very well suited to various applications and it is also easy to compute and interpret.

The measure k of dependency $P \Rightarrow_k Q$ does not capture how actually this partial dependency is distributed among classes of U/Q. For example some decision classes can be fully characterized by P, whereas others may be characterized only partially. To this end we will need also a coefficient $\gamma_P(X) = card\ \underline{P}X / card\ X$ where $X \in U/Q$ which says how elements of each class of U/Q can be classified by employing knowledge P.

Thus the two numbers $\gamma_P(Q)$ and $\gamma_P(X)$, $X \in U/Q$ give us full information about the "classification power" of the knowledge P with respect to the classification U/Q.

Example 2.

Let us compute the degree of dependency of knowledge Q from knowledge P where the corresponding partitions are the following: $U/Q = X_1 = \{1\}$, $X_2 = \{2,7\}$, $X_3 = \{3,6\}$, $X_4 = \{4\}$,

$X_5 = \{5,8\}$ and $U/P = Y_1 = \{1,5\}$, $Y_2 = \{2,8\}$, $Y_3 = \{3\}$, $Y_4 = \{4\}$, $Y_5 = \{6\}$ and $Y_6 = \{7\}$.

Because $\underline{P}X_1 = \emptyset$, $\underline{P}X_2 = Y_6$, $\underline{P}X_3 = Y_3 \cup Y_5$, $\underline{P}X_4 = Y_4$ and $\underline{P}X_5 = \emptyset$, thus $POS_P(Q) = Y_3 \cup Y_4 \cup Y_5 \cup Y_6 = \{3,4,6,7\}$. That is to say that only these elements can be classified into blocks of the partition U/Q employing the knowledge **P**. Hence the degree of dependency between **Q** and **P** is $\gamma_P(Q)=4/8=0.5$. ■

The proposition below is a counterpart of **Proposition 3.**

Proposition 5.

use different letter

1) If $R \Rightarrow_k P$ and $Q \Rightarrow_l P$, then

   $R \cup Q \Rightarrow_m P$, for some $m \geq \max(k,l)$.

2) If $R \cup P \Rightarrow_k Q$, then $R \Rightarrow_l Q$ and

   $P \Rightarrow_m Q$, for some l, $m \leq k$.

3) If $R \Rightarrow_k Q$ and $R \Rightarrow_l P$, then

   $R \Rightarrow_m Q \cup P$, for some $m \leq \min(k,l)$.

4) If $R \Rightarrow_k Q \cup P$, then $R \Rightarrow_l Q$ and

   $R \Rightarrow_m P$, for some l, $m \geq k$.

5) If $R \Rightarrow_k P$ and $P \Rightarrow_l Q$, then $R \Rightarrow_m Q$,

   for some $m \geq l+k-1$. ■

Summary

Dependencies, in particular partial dependencies, in a knowledge base are basic tools when drawing conclusions from basic knowledge, for they state some of the relationships between basic categories in the knowledge base.

Exercises

1. Prove Propositions 1, 2, 3, 4 and 5.

2. Let $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ and assume that the following partitions of U are given:

$$P_1 = \{\{x_1, x_2\}, \{x_3, x_4, x_5\}, \{x_6, x_7, x_8\}\}$$

$$P_2 = \{\{x_1, x_4, x_5\}, \{x_2, x_3\}, \{x_6, x_7, x_8\}\}$$

$$P_3 = \{\{x_4\}, \{x_2, x_3\}, \{x_1, x_5, x_8\}, \{x_6, x_7\}\}$$

$$P_4 = \{\{x_6\}, \{x_2, x_3, x_4\}, \{x_1, x_5, x_7, x_8\}\}$$

Compute degree of dependency between the following partitions: $P_1 \Rightarrow P_2$, $P_2 \Rightarrow P_3$, $P_3 \Rightarrow P_4$, $P_1 \Rightarrow P_4$, $P_2 \Rightarrow P_4$.

References

Buszkowski, W. and Orlowska, E. (1986). On the Logic of Database Dependencies. *Bull. Polish Acad. Sci. Math.*, 34, pp. 345-354.

Novotny, M. and Pawlak, Z. (1988). Independence of Attributes. *Bull. Polish Acad. Sci. Math.*, 36, pp. 459-465.

Novotny, M. and Pawlak, Z. (1989a). Partial Dependency of Attributes. *Bull. Polish Acad. Sci. Math.*, 36, pp. 453-458.

Novotny, M. and Pawlak, Z. (1989b). Algebraic Theory of Independence in Information Systems. *Institute of Mathematics Czechoslovak Academy of Sciences Report*, 51.

Orłowska, E. (1983). Dependencies of attributes in Pawlak's Information Systems. *Fundamenta Informaticae*, 6, pp. 247-256.

Pawlak, Z. (1981). Information Systems-Theoretical Foundations. *(The book in Polish)*, PWN Warsaw.

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

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

Pawlak, Z. (1985). On Rough Dependency of Attributes in Information Systems. *Bull. Polish Acad. Sci. Tech.*, **33**, pp. 551-559.

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. (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).

5. KNOWLEDGE REPRESENTATION

1. Introduction

The issue of knowledge representation is of primary importance in current research in AI and variety of approaches can be found in this area (cf. Bobrow (1977), Bobrow et al. (1977), Brachmann et al.(1980), Davis et. al. (1982), Minski (1975), McDermott (1978), Newell (1982).

Our major concern in this chapter is to discuss the issue of knowledge representation in the framework of concepts introduced so far, i.e. knowledge understood as partition (classification), which can be viewed as semantic definition of knowledge.

For computational reasons we need however syntactic representation of knowledge. To this end we shall employ tabular representation of knowledge, which can be viewed as a special kind of "formal language" used to represent equivalence relations (or partitions) in symbolic form suitable for computer processing. Such a data table will be called *Knowledge Representation Systems* (*KR*-system, or *KRS*). (Sometimes called also *information systems* or *attribute-value system).*

Knowledge representation system can be perceived as a data table, columns of which are labelled by attributes, rows are labelled by objects (states, processes etc.) and each row represents a piece of information about the corresponding object. The data table can be obtained as a result of measurements, observations or represents knowledge of an agent or a group of agents. The origin of the data table is not important from our point of view and we shall be interested only in some formal properties of such tables.

It is easily seen that with each attribute we can associate an equivalence relation. For example the attribute "color" classifies all objects of the universe into categories of objects having the same color, like red, green, blue etc. Hence each table can be viewed as a notation for a certain family of equivalence relations, i.e. knowledge base. All the problems mentioned in the previous paragraphs can be now formulated in terms of classifications induced by attributes and their values. For example knowledge reduction and reasoning about knowledge can be formulated as reduction of attributes and detecting attribute (partial) dependencies.

2. Examples

Before we treat the issues mentioned above more formally, let us give some examples of Knowledge Representation Systems, which, we hope, would make the idea closer.

Example 1.

In this *KRS* pathomorfological changes in cells organelles are listed (cf. Moore et al. (1977)).

State	Volumne Density	Numerical Density	Surface Density
Normal	Normal	Normal	Normal
Proliferation	Normal	Increased	Increased
Hypertrophy	Increased	Normal	Normal
Hyperlasia	Increased	Increased	Increased
Hypoplasia	Decreased	Decreased	Decreased
Atrophy	Decreased	Normal	Decreased
Ageneration	Normal	Decreased	Decreased
Dysplasia	Increased	Decreased	Decreased
Dystrophy	Decreased	Increased	Increased

Table 1

Objects in the system are states of cells of organelle systems. The organelles systems are characterized by attributes Volume Density, Numerical Density and Surface Density.■

Example 2.

Here is an information about patients suffering from heart disease seen in one of the hospitals in Warsaw.

	Gasometry	Dyspnea	Cyanosis	Pulmonary Stasis	Heart Rate	Hepatomegaly	Edema	Degree of Disease Advance
P1	37	1	1	1	62	0	0	1
P2	43	2	3	4	76	8	3	3
P3	42	1	2	1	71	1	0	1
P4	43	0	3	2	80	5	1	1
P5	48	1	3	3	92	6	3	3
P6	38	1	3	2	87	5	1	2
P7	54	0	0	0	95	1	0	2
P8	40	3	0	0	128	1	0	0
P9	40	1	0	0	111	1	0	1
P1	50	0	1	0	68	2	1	1

Table 2

■

Example 3.

In this example a characterization of various animals in terms of Size, Animality and Color is given (cf. Hunt et al. (1966)).

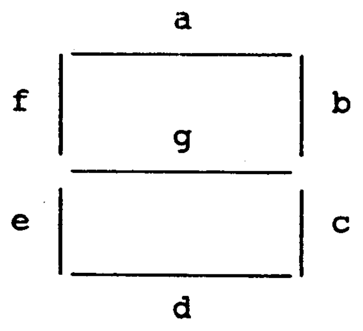
	Size	Animality	Colour
A1	small	bear	black
A2	medium	bear	black
A3	large	dog	brown
A4	small	cat	black
A5	medium	horse	black
A6	large	horse	black
A7	large	horse	brown

Table 3

■

Example 4.

The digit displays in calculators are composed of seven elements as shown below.



Then structure of each digit is shown in the information system below.

Digit	a	b	c	d	e	f	g
0	x	x	x	x	x	x	
1		x	x				
2	x	x		x	x		x
3	x	x	x	x			x
4		x	x			x	x
5	x		x	x		x	x
6	x		x	x	x	x	x
7	x	x	x				
8	x	x	x	x	x	x	x
9	x	x	x	x		x	x

Table 4

Objects in the information system are digits 0,...,9 and attributes are elements a,b,c,d,e,f,g of the display. ■

Example 5.

As is well known digital circuits can be described in aform of truth tables. Such tables can be treated as *KNS*. For example the following table describes a binary full adder.

	a_i	b_i	c_{i-1}	s_i	c_i
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Table 5

Objects in the system are states of the adder denoted 0, 1, 2, 3, 4, 5, 6 and 7, and the attributes are the augments a_i, b_i, sum s_i, previous and current carry c_{i-1}, c_i.■

In what follows we will ignore the semantic contents of the table, i.e. we will consider the data table regardless of what the objects, attributes or their values are, and we will treat *KRS* in an entirely formal way.

3. Formal Definition

In this section a formal definition of Knowledge Representation System will be given and some basic issues connected with this notion will be addressed. More about this approach can be found in Novotny et al.(1983), Grzymała-Busse (1988), Keh-Hsun Chen et al. (1984), Orłowska (1983), Orłowska et al. (1984a,b), Pagliani (1987), Pawlak (1981a,b, 1985), Rauszer (1984, 1985a,b,c, 1986, 1987, 1988), Vakarelov (1987, 1988), Ziarko (1988). Generalization of the idea for the case, when some data are missing in the table (so called incomplete information systems, or incomplete data bases) are discussed in Jeagerman (1978a,b), Lipski (1974, 1981), Słowiński et al. (1988) and others.

Our approach to knowledge representation overlaps in many ways with various fields (cf. Codd (1970), Salton (1968), Wille (1982)) however many profound differences should be noted.

Formally, knowledge representation system can be formulated as follows.

Knowledge Representation System is a pair $S = (U, A)$, where

U - is a nonempty, finite set called the *universe*.

A - is a nonempty, finite set of *primitive attributes*.

Every primitive attribute $a \in A$ is a total function $a:U \longrightarrow V_a$, where V_a- is the set of *values* of a, called the *domain* of a.

With every subset of attributes $B \subseteq A$, we associate a binary relation *IND(B), called an indiscernibility relation* and defined thus:

$$IND(B) = \{(x,y) \in U^2\text{: for every } a \in B,\ a(x) = a(y)\}.$$

Obviously *IND(B)* is an equivalence relation and

$$IND(B) = \bigcap_{a \in B} IND(a)$$

Every subset $B \subseteq A$ will be called an *attribute*. If B is a single element set then B is called primitive, otherwise the attribute is said to be *compound*. Attribute B may be considered as a name of the relation $IND(B)$, or in other words - as a name of knowledge represented by an equivalence relation $IND(B)$. For simplicity, if it does not cause confusion, we shall identify attributes and the corresponding relations.

Thus all notions and properties concerning knowledge expressed in the previous chapters can be now voiced in terms of attributes. For example we may now speak about reducts, core, dependencies of attributes etc, instead of equivalence relations (knowledge).

In particular the value $a(x)$ assigned by the primitive attribute a to the object x can be viewed as a name (or a description) of the primitive category of a to which x belongs (i.e. equivalence class of $IND(a)$ containing x), that is to say $a(x)$ is the name of $[x]_{IND(a)}$. The name (description) of an elementary category of attribute $B \subseteq A$ containing object x is a set of pairs (attribute, value), i.e. $\{a, a(x),\}_{a\in B}$.
$\underline{P}X$ and $\bar{P}X$ will denote the lower and the upper approximation of X respectively, etc.

These ideas can be formulated more precisely in the following way.

Let us note that there is a one to one correspondence between knowledge bases and knowledge representation systems (up to isomorphism of attributes and attribute names). It suffices to assign to arbitrary knowledge base $K = (U, \mathbf{R})$ a knowledge representation system $S = (U, A)$ - in the following way:

If $R \in \mathbf{R}$ and $U/R = \{X_1,\dots,X_k\}$, then to the set of attributes A belongs every attribute $a_R\colon U \longrightarrow V_{a_R}$, such that $V_{a_R} = \{1,\dots,k\}$ and $a_R(x) = i$ if and only if $x \in X_i$ for $i = 1,\dots,k$. Then all notions concerning knowledge bases can be easily expressed in terms of notions of knowledge representation systems.

Thus the knowledge representation system $S = (U, A)$ may be viewed as a description of a knowledge base $K = (U, \mathbf{R})$ (another description of knowledge bases will be given in chapter 7). Each equivalence relation in the knowledge base is represented in the the table by an attribute and each equivalence class of the relation - by an attribute value.

Consequently rows of the table can be viewed as names of some categories. Hence the whole table contains descriptions of all categories of the corresponding knowledge base. In fact the table contains also all possible rules which can be derived from the data included in the table. Hence the knowledge representation system may be considered as a description of all facts and rules which are available in the knowledge base.

In the example which follows we will depict how the concepts discussed in the previous chapters are related to the knowledge representation system.

Example 6

Let us consider the following *KRS*

U	*a*	*b*	*c*	*d*	*e*
1	1	0	2	2	0
2	0	1	1	1	2
3	2	0	0	1	1
4	1	1	0	2	2
5	1	0	2	0	1
6	2	2	0	1	1
7	2	1	1	1	2
8	0	1	1	0	1

Table 6

The universe U consist of 8 elements numbered 1, 2, 3, 4, 5, 6, 7 and 8, the set of attributes is $A = \{a, b, c, d, e\}$, whereas $V = V_a = V_b = V_c = V_d = V_e = \{0, 1, 2\}$.

In the table 8 elements 1, 4 and 5 of U are indiscernible by attribute a, elements 2, 7 and 8 are indiscernible by attributes b and c, and elements 2 and 7 are indiscernible by attributes d and e.

Exemplary partitions generated by attributes in this system are given below.

$U/IND(a) = \{\{2, 8\}, \{1, 4, 5\}, \{3, 6, 7\}\}$

$U/IND(b) = \{\{1, 3, 5\}, \{2, 4, 7, 8\}, \{6\}\}$

$U/IND\{c, d\} = \{\{1\}, \{3, 6\}, \{2, 7\}, \{4\}, \{5\}, \{8\}\}$

$U/IND\{a, b, c\} = \{\{1, 5\}, \{2, 8\}, \{3\}, \{4\}, \{6\}, \{7\}\}$

For example for the set of attributes $C = \{a, b, c\}$ and the subset of the universe $X = \{1, 2, 3, 4, 5\}$ we have $\underline{C}X = \{1, \cancel{2}, 3, 4, 5\}$, $\bar{C}X = \{1, 2, 3, 4, 5, 8\}$ and $BN_C(X) = \{\cancel{8}\}$. 2

Thus the set X is rough with respect to the attributes C, which is to say that we are unable to decide whether elements 2 and 8 are members of the set X or not, employing the set of attributes C. For the rest of the universe classification of elements using the set C of attributes, is possible.

The set of attributes $C = \{a, b, c\}$ is dependent. The attributes a and b are indispensable , whereas the attribute c is superfluous. There is only one reduct of the set C, namely the set $\{a, b\}$ which is also the core of C. Hence in the table 6 we have the following dependency $\{a, b\} \Rightarrow \{c\}$. (see the Proposition 4.2)

This dependency can be also computed from the definition as follows: because the indiscernibility relation $IND(\{a, b\})$ has the following blocks $\{1, 5\}$, $\{2, 8\}$, $\{3\}$, $\{4\}$, $\{6\}$, $\{7\}$ and the indiscernibility relation $IND(\{c\})$ has the blocks $\{1, 5\}$, $\{2, 7, 8\}$ and $\{3, 4, 6\}$, hence $IND(\{a, b\}) \subset IND(\{c\})$.

Let us compute the degree of dependency of attributes $D = \{d, e\}$ from the attributes $C = \{a, b, c\}$ in Table 6. The partition U/D consists of the following blocks, $X_1 = \{1\}$, $X_2 = \{2, 7\}$, $X_3 = \{3, 6\}$, $X_4 = \{4\}$, $X_5 = \{5, 8\}$ and the partition U/C consists of blocks $Y_1 = \{1, 5\}$, $Y_2 = \{2, 8\}$, $Y_3 = \{3\}$, $Y_4 = \{4\}$, $Y_5 = \{6\}$ and $Y_6 = \{7\}$. Because $\underline{C}X_1 = \emptyset$, $\underline{C}X_2 = Y_6$, $\underline{C}X_3 = Y_3 \cup Y_5$, $\underline{C}X_4 = Y_4$ and $\underline{C}X_5 = \emptyset$, thus $POS_C(D) = Y_3 \cup Y_4 \cup Y_5 \cup Y_6 = \{3, 4, 6, 7\}$. That is to say that only these elements can be classified into blocks of the partition U/D employing the set $C = \{a, b, c\}$ attributes. Hence the degree of dependency between C and D is $\gamma_C(D) = 4/8 = 0,5$.

The set of attributes C is D-dependent, and the attribute a is D-dispensable, which means that the D-core of C is one attribute set $\{a\}$. Consequently we have two D-reducts of C, namely $\{a, b\}$,and $\{a, c\}$. This means that there are the following dependencies: $\{a, b\} \Rightarrow \{d, e\}$, and $\{a, c\} \Rightarrow \{d, e\}$ in the table.

Compare also examples in the previous chapter. ∎

4. Significance od of Attributes

Before entering into any detail, a brief intuitive motivations as to the nature of problem discussed in this section seems to be in order.

When speaking about attributes it is obvious that they may have various importance in the analysis of issues being considered. This importance can be pre assumed in advance on the basis of auxiliary knowledge and expressed by properly chosen "weights". In our approach however we do avoid any additional information beside what is included in the table and we will try to compute from data available in the table whether all the attributes are of the same "strength" and if not how they differ in the respect of the "classificatory power".

It is well known that for example when describing patients in terms of symptoms, some of the symptoms may have greater significance for the recognition of the patient health status, then the other. In order to find out the significance of a specific attribute (or group of attributes) it seems reasonable to drop the attribute from the table and see how the classification will be changed without this attribute. If removing the attribute will change the classification considerably it means that its significance is high - in the opposite case the significance should be low.

The idea can be expressed more precisely employing the concept of a positive region discussed in chapter 3. As a measure of significance of the subset of attributes $B' \subseteq B$ with respect to the classification induced by a set of attribute C we will mean the difference

$$\gamma_B(C) - \gamma_{B-B'}(C)$$

which expresses how the positive region of the classification U/C when classifying object by means of attributes B will be affected if we drop some attributes (subset B') from the set B. (Instead of difference one could also take the quotient of $POS_{B-B'}(C)$ and $POS_B(C)$).

The example which follows will illustrate the concept of significance of attributes.

Example 7

Let us compute the significance of the attribute a, b, and c with respect to attributes d and e in Table 6. As shown in the **Example 6** $POS_C(D) = 0,4$, where $D = \{a, b, c\}$ and $C = \{d, e\}$. Because $U/\{b, c\} = \{\{1, 5\}, \{2, 7, 8\}, \{3\}, \{4\}, \{6\}\}$, $U/IND\{a, c\} = \{\{1,5\}, \{2,8\}, \{3,6\}, \{4\}, \{7\}\}$, $U/IND\{a,b\} = \{\{1, 5\}, \{2, 8\}, \{3\}, \{4,\}, \{6\}, \{7\}\}$ and $U/\{d, e\} = \{\{1\}, \{2, 7\}, \{3, 6\}, \{4\}, \{5, 8\}$ hence $POS_{C-a}(D) = \{3, 4, 6\}$, $POS_{C-b}(D) = \{3, 4, 6, 7\}$ and $POS_{C-c}(D) = \{3, 4, 6, 7\}$. Consequently corresponding accuracies are: $\gamma_{C-a}(D) = 0.375$, $\gamma_{C-b}(D) = 0.5$, $\gamma_{C-c}(D) = 0.5$. Thus the attribute a is most significant since it most change the positive region of U/D, i.e. without the attribute a we are unable to classify of object 7 to classes of U/D. Note that the attribute a is D-indispensable and attributes b and c are D-dispensable, thus the attribute a is the core of C with respect to D (D-core of C) and $\{a, b\}$, and $\{a, c\}$ are reducts of C with respect to D, (D-reducts of C). (Compare **Example 6**).

Remark

At the end it is worthwhile to mention that the notion of a Knowledge Representation System apparently looks like a relational table in the relational data base model (cf. Codd (1970)). There is ,however, an essential difference between these two models. Most importantly, the relational model is not interested in the meaning of the information stored in the table. The emphasis is placed on efficient data structuring and manipulation. Consequently the objects about which information is contained in the table are not represented in the table. This is in contrast with the primary assumption of our approach presented here. In the Knowledge Representation System all objects are explicitly represented and the attribute values i.e. the table entries have associated explicit meaning as features or properties of the objects. In addition to that the emphasis in our model is put mainly not on data structuring and manipulation but on analysis of actual dependencies existing in data, and data reduction, which is rather closer to statistical data model.

Summary

For representing knowledge we advocate using data tables, sometimes called information systems, attribute-value systems, conditions-actions tables, etc. Columns of such tables are labelled with attributes and rows with object of the universe. With each group of columns (subsets of attributes) we can associate an equivalence relation whereas with each group of rows - a set. Thus such a table can serve as a representation of knowledge base - in which attributes represent knowledge and sets - concepts. It is easy to represent basic properties of knowledge base like reducts, the core, dependencies etc in terms of data tables properties which allow the replacement of semantic definitions by syntactic ones. Semantic approach is needed when proving some properties of knowledge, while syntactical ones are necessary to develop algorithms.

Excercises

1. Compute the core and reducts of attribute for Examples 1, 3 and 4 given in section 2 (Tables 1, 3 and 4).

2. Compute significance of attributes in examples in Exercise 1.

3. Compute degree of dependency for arbitrary chosen attributes in examples in Exercise 1.

References

Bobrow, D. G. (1977). A Panel on Knowledge Representation. *Proc. Fifth International Joint Conference on Artificial Intelligence, Carnegie-Melon University.* Pittsburgh, PA.

Bobrow, D. G, and Winograd, T. (1977). An Overview of KRL: A Knowledge Representation Language. *Journal of Cognitive Sciences*, 1, pp. 3-46.

Brachman, R. J. and Smith, B. C. (1980). Special Issue of Knowledge Representation. *SIGART Newsletter*, **70**, pp. 1-138.

Codd, E.F. (1970). A Relational Model of Data for Large Shared Data Banks. *Comm. ACM.*, **13**, pp. 377,387.

Davis, R. and Lenat, D. (1982). Knowledge-Based Systems in Artificial Intelligence. *McGraw-Hill.*

Grzymała-Busse, J. (1988). Knowledge Acquisition under Uncertainty - a Rough Set Approach. *Journal of Intelligent and Robotics Systems*, 1, pp. 3-16.

Jaegerman, M. (1978a). Information Storage and Retrieval Systems with Incomplete Information. Part I. *Fundamenta Informaticae*, 2, pp. 17-41.

Jaegerman, M. (1978b). Information Storage and Retrieval Systems with Incomplete Information. Part II. *Fundamenta Informaticae*, 2, pp. 141-166.

Keh-Hsun Chen, Raś, Z. and Skowron, A. (1984). Attributes and Rough Properties in Information Systems. *Proc. of Conf. on Information Science and Systems*. March 14-16, 1984, Princeton University, Princeton, pp. 362-366.

Lipski, W. Jr. (1979). On Semantic Issues Connected with Incomplete Information Databases. *ACM Transactions on Database Systems*, 4, pp. 269-296.

Lipski, W. Jr. (1981). On Databases with Incomplete Information. *Journal of the ACM.*, 28, pp.41-70.

Moore, G. W., Riede, U. N. & Sandritter, G. (1977). Application of Quine's Nullities to a Quantitive Organelle Pathology. *Journal of Theoretical Biology*, 65, pp. 633-657.

McDermott, D. (1978). The Last Survey of Representation of Knowledge. *Proc. of the AISB/GI Conference on AI.* Hamburg, pp. 286-221.

Minski, M. (1975). A Framework for Representation Knowledge. *In: Winston, P.(ed.) The Psychology of Computer Vision. McGraw-Hill.* New York, pp. 211-277.

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 18, pp. 87-127.

Novotny, M. and Pawlak, Z. (1983). On a Representation of Rough Sets by Means of Information Systems. *Fundamenta Informaticae*, 6, pp. 286-296.

Novotny, M. and Pawlak, Z. (1988). Independence of Attributes. *Bull. Polish Acad. Sci. Math.*, 36, pp. 459-465.

Novotny, M. and Pawlak, Z. (1988). Partial Dependency of Attributes. *Bull. Polish Acad. Sci. Math.*, 36, pp. 453-458.

Orlowska, E. (1983). Dependencies of attributes in Pawlak's Information Systems. *Fundamenta Informaticae*, 6, pp. 247-256.

Orlowska, E. and Pawlak, Z. (1984a). Expressive Power of Knowledge Representation Systems. *International Journal of Man-Machine Studies*, 20, pp. 485-500.

Orlowska, E. and Pawlak, Z. (1984b). Representation of Non-deterministic Information. *Theoretical Computer Science*, 29, pp. 27-39.

Pagliani, P. (1987). Polystructured Model Spaces as Algorithm Oriented Models for Approximation Spaces and Pawlak's Information Systems. *Facolta di Science dell Informazione, Internal Raport PAG* /3.

Pawlak, Z. (1981a). Information Systems-Theoretical Foundations. *(The book in Polish) PWN Warsaw.*

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

Pawlak, Z. (1985). On Rough Dependency of Attributes in Information Systems. *Bull. Polish Acad. Sci. Tech.*, 33, pp. 481-485.

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

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. (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. Måth.*, 33, pp. 561-569.

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

Salton, G. (1968). Automatic Information Organization and Retrieval. *McGraw-Hill, New York.*

Słowiński, R. and Stefanowski, J. (1988). Rough Classification in Incomplete Information Systems. *Mathematical Modeling* (to appear).

Vakarelov, D.(1987). Abstract Characterization of Some Modal Knowledge Representation Systems and the Logic NIL of Nondeterministic Information.In: *Jorraud, Ph. and Squrev, V. (ed.) Artificial Intelligence II, 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. (1987). Definable Sets in Knowledge Representation Systems. *Bull. Polish Acad. Sci. Math.*, 35, pp. 629-635.

Wasilewska, A. (1988). Knowledge Representation Systems - Syntactic Methods. *Proc.IPMU-88 Conference, Springer Verlag, Lecture Notes on Computer Science*, 313, pp. 239-254.

Wille, R. (1982). Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. *In: I.Rival (Ed.), Ordered Sets, Reidel,* Dordrecht - Boston, pp. 445-470.

Ziarko, W. (1987). On Reduction of Knowledge Representation. *Proc. 2-nd International Symp.on Methodologies of for Intelligent Systems*. Charlotte, NC. North Holland, pp. 99-113.

Ziarko, W. (1990). Variable Precision Rough Set Model. *Journal of Computer an System Science.* (~~submitted).~~ (to appear)

6. DECISION TABLES

1. Introduction

In this section we will consider special, important class of knowledge representation systems, called *decision tables*, ~~which~~ play an important part in many applications. There is a variety of literature on decision tables and basics facts on this subject can be found for example in ~~Montalbano~~ (1974) or Hurley (1983).

ies Decision table is a kind of a prescription, which specify what decisions (actions) should be undertaken when some conditions are satisfied. Most decision problems can be formulated employing decision table formalism, therefore this tool is particularly useful in decision making.

It turns out that the concepts introduced in previous chapters provide a very good framework as a basis of decision tables theory (cf.Pawlak (1985, 1986, 1987), Boryczka et al. (1988), Wong et al. (1986)) and their applications (cf. for example Mrózek (1987), Słowiński et al. (1988)).

In this chapter we want to discuss some basic problems of decision tables theory in terms of rough sets philosophy.

2. Formal Definition and Some Properties

Decision tables can be defined in terms of *KR*-systems as follows.

Let $K = (U, A)$ be a knowledge representation system and let $C, D \subset A$ be two subsets of attributes called *condition* and *decision attributes* respectively. *KR*-system with distinguished condition and decision attributes will be called a *decision table* and will be denoted $T = (U, A, C, D)$, or in short *CD*-decision table.

Equivalence classes of the relations $IND(C)$ and $IND(D)$ will be called *condition* and *decision classes*, respectively.

With every $x \in U$ we associate a function $d_x: A \longrightarrow V$, such that $d_x(a) = a(x)$, for every $a \in C \cup D$; the function d_x will be called a *decision rule* (in T), and x will be referred to as a *label* of the decision rule d_x.

Remark

Let us note that elements of the set U in a decision table do not represent in general any real objects but are simple identifiers of decision rules. ■

If d_x is a decision rule, then the restriction of d_x to C, denoted $d_x|C$, and the restriction of d_x to D, denoted $d_x|D$ will be called *conditions* and *decisions (actions)* of d_x respectively.

The decision rule d_x is *consistent* (in T) if for every $y \neq x$, $d_x|C = d_y|C$ implies $d_x|D = d_y|D$; otherwise the decision rule is *inconsistent*.

A decision table is *consistent* if all its decision rules are consistent; otherwise the decision table is *inconsistent*.

Remark.

Consistency (inconsistency) sometimes may be interpreted as determinism (nondeterminism), but we shall not use this interpretation in this book, except if stated not otherwise.■

The following is the important property that establishes relationship between consistency and dependency of attributes in a decision table.

Proposition 1.

A decision table $T = (U,A,C,D)$ is consistent, if and only if $C \Rightarrow D$.■

From **Proposition 1** it follows the practical method of checking consistency of decision table by simply computing the degree of dependency between condition and decision attributes. If the degree of dependency equals to 1 then we conclude that the table is consistent; otherwise it is inconsistent.

The next property is also important from a practical perspective.

Proposition 2.

Each decision table $T = (U, A, C, D)$ can be uniquely decomposed into two decision tables $T_1 = (U_1, A, C, D)$ and $T_2 = (U_2, A, C, D)$ such that $C \Rightarrow_1 D$ in T_1 and $C \Rightarrow_0 D$ in T_2, where $U_1 = POS_C(D)$ and $U_2 = \bigcup_{X \in U/IND(C)} BN_C(X)$. ∎

The Proposition 2 have the following interpretation. Suppose that we have computed the dependency between condition and decision attributes. If the table turned out to be inconsistent i.e. the dependency degree was less than 1, then according to Proposition 2 we can decompose the table into two sub tables: one totally inconsistent with dependency coefficient equal to zero and the second entirely consistent with the dependency equal to one. This decomposition however is possible only if the degree of dependency is greater than zero and different from 1.

Example 1.

Let us consider Table 1 given below

U	*a*	*b*	*c*	*d*	*e*
1	1	0	2	2	0
2	0	1	1	1	2
3	2	0	0	1	1
4	1	1	0	2	2
5	1	0	2	0	1
6	2	2	0	1	1
7	2	1	1	1	2
8	0	1	1	0	1

Table 1

Assume that *a*,*b* and *c* are condition attributes and *d* and *e* are decision attributes. In this table for instance the decision rule 1 is inconsistent whereas ~~the~~ decision rule 3 is consistent. By employing Proposition ~~1~~ 2 we can decompose the decision Table 1 into the following two decision tables:

U_1	*a*	*b*	*c*	*d*	*e*
3	2	0	0	1	1
4	1	1	0	2	2
6	2	2	0	1	1
7	2	1	1	1	2

Table 2

U_2	*a*	*b*	*c*	*d*	*e*
1	1	0	2	2	0
2	0	1	1	1	2
5	1	0	2	0	1
8	0	1	1	0	1

Table 3

The decision Table 2 is consistent, whereas the decision Table 3 is totally inconsistent, which means that all decision rules in Table 2 are consistent, and in decision Table 3 - all decision rules are inconsistent.

■

3. Simplification of Decision Tables

Simplification of decision tables is of primary importance in many applications. Example of simplification is the reduction of condition attributes in a decision table. In the reduced decision table the same decisions can be based on a smaller number of conditions. This kind of simplification eliminates the need for checking unnecessary conditions or, in some applications - performing expensive tests to arrive at a conclusion which eventually could be achieved by simpler means.

Simplification of decision tables has been investigated by many authors (cf.Hurley (1983)), and there is a variety of informal approaches to this problem.

The approach to table simplification presented here consists of the following steps:

1) Computation of reducts of condition attributes which is equivalent to elimination of some column from the decision table.

2) Elimination of duplicate rows.

3) Elimination of superfluous values of attributes.

Remark.

We should note that in contrast to the general notion of knowledge representation system rows do not represent here description of any real objects. Consequently duplicate rows can be eliminated as they correspond to the same decisions. ∎

Thus the proposed method consists in removing superfluous condition attributes (columns), duplicate rows and in addition to that irrelevant values of condition attributes.

In this way we obtain "incomplete" decision table, containing only those values of condition attributes which are necessary to make decisions. According to our definition of a decision table the incomplete table is not a decision table and can be treated as an abbreviation of such table.

From mathematical point of view, removing attributes and removing values of attributes are alike and will be explained in what follows.

For the sake of simplicity we assume that the set of condition attributes is already reduced, i.e. there are not superfluous condition attributes in the decision table.

As we have already mentioned with every subset of attributes $B \subseteq A$ we can associate partition $U/IND(B)$, and consequently set of condition and decision attributes define partitions of objects into *condition* and *decision classes*.

Because we want to discern every decision class using minimal number of conditions - our problem can be reduced now to searching for relative reducts of condition classes with respect to decision classes. To this end we can use similar methods to that of finding reducts of attributes.

Similarly we can reduce superfluous values of condition attributes form a decision table.

From section 5.3 we know that with every subset of attributes $B \subseteq A$ and object x we may associate set $[x]_B$. ($[x]_B$ denotes an equivalence class of the relation $IND(B)$ containing an object x, i.e $[x]_B$ is an abbreviation of $[x]_{IND(B)}$). Thus with any set of condition attributes C in a decision rule d_x we can associate set $[x]_C = \bigcap_{a \in C} [x]_a$. But each set $[x]_a$ is uniquely determined by attribute value $a(x)$, hence in order to remove superfluous values of condition attributes, we have to eliminate all superfluous equivalence classes $[x]_a$ from the equivalence class $[x]_C$ as discussed in section 3.4. Thus problems of elimination of superfluous values of attributes and elimination of corresponding equivalence classes are equivalent.

The example which follows will illustrate the concepts discussed so far.

Example 2

Suppose we are given the following decision table

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	1	0	2	2
7	2	2	2	2	2

Table 4

where a, b, c and d are condition attributes and e is decision attribute.

It is easy to compute that the only e-dispensable condition attribute is c, consequently we can remove column c in Table 4, which yields Table 5 shown 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	1	2	2
7	2	2	2	2

Table 5

In the next step we have to reduce superfluous values of condition attributes, in every decision rules. To this end we have first to compute core values of condition attributes in every decision rule.

For the sake of illustration let us compute the core values of condition attributes for the first decision rule, i.e. the core of the family of sets $\mathbf{F} = \{[1]_a, [1]_b, [1]_d\} = \{\{1, 2, 4, 5\}, \{1, 2, 3\}, \{1, 4\}\}$.

From considerations in sections 3.4 and 5.3 we have $[1]_{\{a,b,d\}} = [1]_a \cap [1]_b \cap [1]_d = \{1, 2, 4, 5\} \cap \{1, 2, 3\} \cap \{1, 4\} = \{1\}$, moreover $a(1) = 1$, $b(1) = 0$ and $d(1) = 1$. In order to find dispensable categories we have to drop one category at a time and check whether the intersection of remaining categories is still included in the decision category $[1]_e = \{1, 2\}$, i.e.

$$[1]_b \cap [1]_d = \{1, 2, 3\} \cap \{1, 4\} = \{1\}$$
$$[1]_a \cap [1]_d = \{1, 2, 4, 5\} \cap \{1, 4\} = \{1, 4\}$$
$$[1]_a \cap [1]_b = \{1, 2, 4, 5\} \cap \{1, 2, 3\} = \{1, 2\}$$

This means that the core value is $b(1) = 0$. In a similar way we can compute remaining core values of condition attributes in every decision rule and final results are presented in Table 6 below.

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 6

Having computed core values of condition attributes we can proceed to compute value reducts.

As an example let us compute value reducts for the first decision rule of the decision table. Accordingly to the definition given in section 3.4, in order to compute reducts of the family $F = \{[1]_a, [1]_b, [1]_d\} = \{\{1, 2, 4, 5\}, \{1, 2, 3\}, \{1, 4\}\}$ we have to find all subfamilies $G \subseteq F$ such that $\bigcap G \subseteq [1]_e = \{1, 2\}$. There are three following subfamilies of F

$$[1]_b \cap [1]_d = \{1, 2, 3\} \cap \{1, 4\} = \{1\}$$

$$[1]_a \cap [1]_d = \{1, 2, 4, 5\} \cap \{1, 4\} = \{1, 4\}$$

$$[1]_a \cap [1]_b = \{1, 2, 4, 5\} \cap \{1, 2, 3\} = \{1, 2\}$$

and only two of them

$$[1]_b \cap [1]_d = \{1, 2, 3\} \cap \{1, 4\} = \{1\} \subseteq [1]_e = \{1, 2\}$$

$$[1]_a \cap [1]_b = \{1, 2, 4, 5\} \cap \{1, 2, 3\} = \{1, 2\} \subseteq [1]_e = \{1, 2\}$$

are reducts of the family F. Hence we have two value reducts: $b(1) = 0$ and $d(1) = 1$ or $a(1) = 1$ and $b(1) = 0$. This means that attribute values of attributes a and b or d and b are characteristic for decision class 1 and do not occur in any other decision classes in the decision table. We see also that the value of attribute b is the intersection of both value reducts, $b(1) = 0$, i.e. it is the core value.

In Table 7 below we list value reducts for all decision rules in Table 1.

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	x	x	2	2
6′	2	x	x	2
7	x	x	2	2
7′	x	2	x	2
7′′	2	x	x	2

Table 7

As we can see from Table 7 for decision rules 1 and 2 we have two value reducts of condition attributes. Decision rules 3,4 and 5 have only one value reduct of condition attributes for each decision rule row. The remaining decision rules 6 and 7 contain two and three value reducts respectively.

Hence there are two reduced forms of decision rule 1 and 2, decision rules 3, 4 and 5 have only one reduced form each, decision rule 6 has two reducts and decision rule 7 have three reducts.

Thus there are 4 x 2 x 3 = 24 (not necessarily different) solutions to our problem. One such solution is presented in the table below.

U	*a*	*b*	*d*	*e*
1	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	x	2	x	2
7	2	x	x	2

Table 8

Another solution is shown in Table 9

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

Table 9

Because decision rules 1 and 2 are identical and so are rules 5, 6 and 7 so we can represent our table in the form

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

Table 10

In fact enumeration of decision rules is not essential so we can enumerate them arbitrary and we get as a final result the table below.

U	*a*	*b*	*d*	*e*
1	1	0	x	1
2	0	x	x	0
3	x	1	1	0
4	x	x	2	2

Table 11

This solution will be referred to as *minimal*, and will be discussed in more detail in the next chapter.

■

The presented method of decision table simplification can be named *semantic*, since it refers to the meaning of the information contained in the table.In the next section we will present another method of decision table simplification, which could be called *syntactic*, because it reefers to purely formal properties of decision tables - which leads to simpler algorithms and programs.

Summary

In order to simplify a decision table we should first find reducts of conditions attributes, remove duplicate rows and then find value-reducts of condition attributes and again, if necessary, remove duplicate rows. This method leads to a simple algorithm for decision table simplification or generation of decision rules (algorithms) from examples, which according to our experiments, out performs other methods, in terms of achievable degree in the number of conditions and what more, gives all possible solutions to the problem.

We conclude this section with the following remark.

As we have seen, a subset of attributes may have more that one reduct (relative reduct), hence the simplification of decision tables does not yield unique results. Thus some decision tables possibly can be optimized according to pre assumed criteria.

Exercises

1. Given a decision table as below.

U	*a*	*b*	*c*	*d*	*e*
1	0	2	0	1	0
2	2	0	1	0	2
3	0	0	1	2	2
4	2	1	1	0	0
5	0	0	1	2	0
6	2	0	0	2	0
7	0	0	2	1	0
8	2	1	1	0	2
9	0	2	1	1	0

Table 12

where *a*, *b* and *c* are condition attributes, whereas *d* and *e* are decision attributes.

a) Decompose the table into consistent and inconsistent part.

b) Compute degree of dependency between the condition and denison attributes.

c) Check whether the set of condition attributes is dependent or not with respect of decision attributes.

d) Compute the core and reducts with of condition attributes with respect to decision attributes.

e) Simplify the decision table, i.e. find all redundant values of condition attributes.

f) Check whether the set of decision attributes is dependent or independent.

e) Compute the significance of condition attributes with respect to decision attributes.

References

Boryczka, M. and Słowiński, R. (1988). Derivation of Optimal Decision Algorithms from Decision Tables Using Rough Sets. *Bull. Polish Acad. Sci. Tech.*, **36**, pp. 143-159.

Hurley, R. B. (1983). Decision Tables in Software Engineering. *Van Nostrand Reinhold Company, New York.*

Montalbano, M. (1974). Decision Tables. *Science Research Associates.* Chicago 1974.

Mrózek, A. (1987). Rough Sets and Some Aspects of Expert Systems Realization. *Proc. 7-th Internal Workshop on Expert Systems and their Applications, Avignon, France,* pp. 597-611.

Mrózek, A. (1989). Rough Sets and Dependency Analysis Among Attributes in Computer Implementation of Expert Inference Models. *International J. of Man-Machine Studies*, **30**, pp. 457-473.

Mrózek, A. (1990). Rough Sets and Decision Rules Generation. *International J. of Man-Machine Studies.* (To appear).

Pawlak, Z. (1985). Rough Sets and Decision Tables. *Lecture Notes in Computer Science*, Springer Verlag, **208**, pp. 186-196.

Pawlak, Z. (1986). On Decision Tables. *Bull. Polish Acad. Sci. Tech.*, **34**, pp. 563-572.

Pawlak, Z. (1987). Decision Tables - A Rough Sets Approach. *Bull. of EATCS*, **33**, pp. 85-96.

Słowinski, K. and Słowinski, R. (1988). Rough Set Approach to Analysis of Data from Peritoneal Lavage in Acute Pancreatitis. *Medical Informatics*, 13, pp. 143-159.

Wong S. K. M. and Ziarko, W. (1986). On Optimal Decision Rules in Decision Tables. *Bull. Polish Acad. Sci. Math.*, 33, pp. 693-696.

7. REASONING ABOUT KNOWLEDGE

1. Introduction

The concept of the rough set 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 (1987), Rasiowa et al.(1985, 1986a,b), Rauszer (1985, 1986), Szczerba (1987), Vakarelov (1981, 1989) [illegible] 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, suited to the ideas presented in the previous chapters of the book, 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, as we have emphasized many times in this book, is reasoning about knowledge concerning certain reality. We have assumed that knowledge is represented as a value-attribute table, called Knowledge Representation System.

Representation of knowledge in tabular form, has great advantages in particular for its clarity. The considered algorithms of knowledge analysis (reduction, dependencies etc.) discussed so far are rather complex.

It turns out that the data table may be also looked at from different angle, namely as a set of propositions about the reality and consequently can be treated by means of logical tools, which will be developed in this chapter. 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, as formulated in Chapters 3 and Chapter 4, and to this end, in contrast to semantic approach employed previously, now 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 $\overset{\text{meet}}{\text{meat}}$ 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 knowledge representation.

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 $\{ \sim , \vee , \wedge , \longrightarrow , \equiv \}$ 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 $v \in V_a$.

Φ

2) If ϕ and Ψ are formulas of the *DL*-language, then so are $\sim\Phi$, $(\Phi \vee \Psi)$, $(\Phi \wedge \Psi)$, $(\Phi \longrightarrow \Psi)$, 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 KR-System $S = (U, A)$. 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) $x \models \phi \longrightarrow \psi$ iff $x \models \sim\phi \vee \psi$

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

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) $|\phi \longrightarrow \Psi|_S = - |\phi|_S \cup |\Psi|_S$

(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, $\models_S \phi$, 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) $\models_S \phi$ iff $|\phi|_S = U$

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

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

(d) $\models_S \phi \equiv \psi$ 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

space

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 with given *A* and *V*, we need some auxiliary notions and denotations.

We will use the following abbreviations:

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

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

Formula of the form

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

where $v_i \in V_{a_i}$, $P = \{a_1, a_2, \dots, a_n\}$, 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 knowledge representation 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 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.

Example 1

Let us consider the following *KR*-system

U	a	b	c
1	1	0	2
2	2	0	3
3	1	1	1
4	1	1	1
5	2	1	3
6	1	0	3

Table 1

The following are all basic formulas (basic knowledge) in the *KR*-system. For simplicity we will omit the symbol of disjunction $\wedge$ in basic formulas: $a_1\ b_0\ c_2$, $a_2\ b_0\ c_3$, $a_1\ b_1\ c_1$, $a_2\ b_1\ c_3$, $a_1\ b_0\ c_3$.

The characteristic formula of the system is

$a_1\ b_0\ c_2 \vee a_2\ b_0\ c_3 \vee a_1\ b_1\ c_1 \vee a_2\ b_1\ c_3 \vee a_1\ b_0\ c_3$.

For the sake of illustration let us give meanings of some formulas in the system.

$| a_1 \vee b_0\ c_2 | = \{1, 3, 4, 6\}$.

$| (a_2\ b_1) | = \{1, 2, 3, 4, 6\}$

$|b_0 \longrightarrow c_2| = \{1, 3, 4, 5\}$

$|a_2 \equiv b_0| = \{1, 2, 3, 4, 5, 6\}$ ■

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) $(a, v) \equiv \bigvee_{\substack{u \in V_a \\ u \neq v}} (a, u)$, 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 *KR*-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 *KR*-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* containSall 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 $\vdash \phi \equiv \psi$.■

Example 2

Below are given normal forms of formulas considered in Example 1.

$$a_1 \vee b_0\, c_2 = a_1\, b_0\, c_2 \vee a_1\, b_1\, c_1 \vee a_1\, b_0\, c_3$$

$$(a_2\, b_1) = a_1\, b_0\, c_2 \vee a_2\, b_0\, c_3 \vee a_1\, b_1\, c_1 \vee a_1\, b_0\, c_3$$

$$b_0 \longrightarrow c_2 = a_1\, b_0\, c_2 \vee a_1\, b_1\, c_1 \vee a_2\, b_1\, c_3$$

$$a_2 \equiv b_0 = a_2\, b_0\, c_3 \vee a_1\, b_1\, c_1$$

Examples of formulas in {*a*,*b*}-normal in Table 1 are given next.

1 $$(\sim (a_2\, b_1) = a_1\, b_0 \vee a_2\, b_0 \vee a_1\, b_1 \vee a_1\, b_0$$

$$a_2 \equiv b_0 = a_2\, b_0 \vee a_1\, b_1$$

The following are examples of formulas in {*b*, *c*}-normal forms in Table 1.

$$a_1 \vee b_0\, c_2 = b_0\, c_2 \vee b_1\, c_1 \vee b_0\, c_3$$

$$b_0 \longrightarrow c_2 = b_0\, c_2 \vee b_1\, c_1 \vee b_1\, c_3$$

Thus in order to compute the normal form of a formula we have to transform the formula by means of propositional calculus axioms and the specific axioms for a given *KR*-system.

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

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

If $\phi \longrightarrow \psi$ is a decision rule and ϕ and ψ are *P*-basic and *Q*-basic formulas respectively, then the decision rule $\phi \longrightarrow \psi$ will be called a *PQ-basic decision rule*, (in short *PQ-rule*), or *basic rule* when *PQ* is known.

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

A *PQ*-rule $\phi \longrightarrow \psi$ 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*.■

Example 3

The rule $b_0 \longrightarrow c_2$ is false in Table 1, because the $\{b, c\}$-normal form of b_0 is $b_0\ c_2 \vee b_0\ c_3$, $\{b, c\}$-normal form of c_2 is $b_0\ c_2$, and the formula $b_0\ c_3$ does not occur in the successor of the rule.

On the other hand the rule $a_2 \longrightarrow c_3$ is true in the table, because the $\{a, c\}$-normal form of a_2 is $a_2\ c_3$, whereas the $\{a, c\}$-normal form of c_3 is $a_2\ c_3 \vee a_1\ c_3$.■

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

Remark

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 $\phi \longrightarrow \psi$ 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*.

Remark

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 - and a decision table with *P* and *Q* as condition and decision attributes respectively. Hence a *PQ*-algorithm and *PQ*-decision table may be considered as equivalent concepts.

Example 4

Consider the *KR*-system shown below.

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

Table 2

Assume that $P = \{a,b,c\}$ and $Q = \{d,e\}$ are condition and decision attributes respectively. Sets *P* and *Q* uniquely associate the following *PQ*-decision algorithm with the table:

$$a_1\, b_0\, c_2 \longrightarrow d_1\, e_1$$

$$a_2\, b_1\, c_0 \longrightarrow d_1\, e_0$$

$$a_2\, b_1\, c_2 \longrightarrow d_0\, e_2$$

$$a_1\, b_2\, c_2 \longrightarrow d_1\, e_1$$

$$a_1\, b_2\, c_0 \longrightarrow d_0\, e_2$$

If we assume that $R = \{a,b\}$ and $T = \{c,d\}$ are condition and decision attributes respectively, the then RT-algorithm determined by Table 2 is the following:

$$a_1\ b_0 \longrightarrow c_2\ d_1$$

$$a_2\ b_1 \longrightarrow c_0\ d_1$$

$$a_2\ b_1 \longrightarrow c_2\ d_0$$

$$a_1\ b_2 \longrightarrow c_2\ d_1$$

$$a_1\ b_2 \longrightarrow c_0\ d_0 \qquad \blacksquare$$

Of course both algorithms are admissible and complete.

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

Remark

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

Let us also remark that in order to check whether a decision rule $\phi \longrightarrow \psi$ 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 depict the above ideas by the following example.

Example 5

Consider again the *KR*-system as in Example 4

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

Table 3

with $P = \{a,b,c\}$ and $Q = \{d,e\}$ as condition and decision attributes respectively. Let us check whether the PQ-algorithm

$$a_1 b_0 c_2 \longrightarrow d_1 e_1$$

$$a_2 b_1 c_0 \longrightarrow d_1 e_0$$

$$a_2 b_1 c_2 \longrightarrow d_0 e_2$$

$$a_1 b_2 c_2 \longrightarrow d_1 e_1$$

$$a_1 b_2 c_0 \longrightarrow d_0 e_2$$

is consistent or not.

Because the predecessors of all decision rules in the algorithm are different, (i.e. all decision classes are discernible by predecessors of all decision rules in the algorithm), then all decision rules in the algorithm are consistent (true) and consequently the algorithm is consistent. This can be also seen directly from Table 4.The RT-algorithm, where $R = \{a,b\}$ and $T = \{c,d\}$

$$a_1\ b_0 \longrightarrow c_2\ d_1$$

$$a_2\ b_1 \longrightarrow c_0\ d_1$$

$$a_2\ b_1 \longrightarrow c_2\ d_0$$

$$a_1\ b_2 \longrightarrow c_2\ d_1$$

$$a_1\ b_2 \longrightarrow c_0\ d_0$$

is inconsistent because the rules

$$a_2\ b_1 \longrightarrow c_0\ d_1$$

$$a_2\ b_1 \longrightarrow c_2\ d_0$$

have the same predecessors and different successors, i.e. we are unable do discern decisions $c_0\ d_1$ and $c_2\ d_0$ by means of conditions $a_2\ b_1$. Thus both rules are inconsistent (false) in the *KR*-system. Similarly, the rules

$$a_1\ b_2 \longrightarrow c_2\ d_1$$

$$a_1\ b_2 \longrightarrow c_0\ d_0$$

are also inconsistent (false).

There is only one consistent rule $a_1\ b_0 \longrightarrow c_2\ d_1$ in the *TR*-algorithm and consequently the algorithm is inconsistent. This is visible much easily when representing the decision algorithm as decision table easier

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

Table 4

For simplicity we rearranged the rows, and separated decision classes by underlining. ■

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 / than 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 $P \Rightarrow Q$.

It can be easily seen that the concept of dependency of attributes corresponds exactly to that introduced in chapter 4.

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~~ that 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 $0 \le k \le 1$. 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.

All these concept correspond exactly to those discussed in chapter 4.

For example the degree of dependency between attributes $\{a, b, c\}$ and $\{d, e\}$ in the algorithm considered in Example 5 in the previous section is 1, whereas the dependency between $\{a, b\}$ and $\{c, d\}$ is 0.2, because there is only one consistent (true) decision rule out of five decision rules in the algorithm.

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. The problem corresponds exactly to that discussed in Section 3.3, however now the question of attribute reduction will be formulated in logical terms. In this section we will discuss the case of a consistent algorithm.

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

PQ 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 $CORE\ (P,Q)$.

Remark

The reader is advised to interpret all the above definitions in terms of indiscernibility. ■

Let us also note that the counterpart of the proposition 2 in chapter 3 is also valid here.

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.

Example 6

Let us consider the following *KR*-system

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

Table 5

and the *PQ*-algorithm in the system shown below, where $P = \{a, b, c\}$ and $Q = \{d, e\}$ are condition and decision attributes respectively.

$$a_1\ b_0\ c_2 \longrightarrow d_1\ e_1$$

$$a_2\ b_1\ c_0 \longrightarrow d_1\ e_0$$

$$a_2\ b_1\ c_2 \longrightarrow d_0\ e_2$$

$$a_1\ b_2\ c_2 \longrightarrow d_1\ e_1$$

$$a_1\ b_2\ c_0 \longrightarrow d_0\ e_2$$

as in **Example 4**, where $P = \{a,\ b,\ c\}$ and $Q = \{d,\ e\}$ are condition and decision attributes respectively.

Let us first compute reducts of condition attributes in the algorithm. It is easy to see that the set of attributes P is dependent in the algorithm, and the core attribute is c. Hence there are two reduct of P, namely $\{a, c\}$ and $\{b,\ c\}$. The PQ-algorithm can be reduced then as

$$a_1\ c_2 \longrightarrow d_1\ e_1$$

$$a_2\ c_0 \longrightarrow d_1\ e_0$$

$$a_2\ c_2 \longrightarrow d_0\ e_2$$

$$a_1\ c_2 \longrightarrow d_1\ e_1$$

$$a_1\ c_0 \longrightarrow d_0\ e_2$$

or

$$b_0\ c_2 \longrightarrow d_1\ e_1$$

$$b_1\ c_0 \longrightarrow d_1\ e_0$$

$$b_1\ c_2 \longrightarrow d_0\ e_2$$

$$b_2\ c_2 \longrightarrow d_1\ e_1$$

$$b_2\ c_0 \longrightarrow d_0\ e_2$$

The above considerations can be easily followed, employing tabular form of representing algorithms, for the basic operations on algorithms can be traced easily, when using this kind of notation.

The *PQ*-algorithm given in this example can be represented ~~as~~ in the following decision table

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

Table 6

in which for simplicity we rearranged the decision rules and the decision classes are separated by underlining.

In order to find core set of attributes we have to drop condition attributes, one by one and see whether ~~thus~~ the obtained decision table (algorithm) is ~~consisted~~ or not.

Removing the attribute *a* we get the table consistent

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

Table 7

which is consistent. Dropping attribute *b* we get again consistent decision table

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

Table 8

However when dropping attribute *c* the result is an incon- sistent table

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

Table 9

Rules 4 and 5 are inconsistent and so are rules 2 and 3, therefore the attribute *c* is the core of the set of condition attributes *{a, b, c}*, and there are two reducts of this set of attributes, *{a, c}* and *{b, c}*. Thus the algorithm has two reduced forms as shown in tabular form in Tables 7 and 8. ■

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~~ an 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 theorem 9 is also valid.

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

where is theorem

Example 7

Consider again *KR*-system as shown in Table 3, and the following set of condition and decision attributes *T* = {*a*, *b*} and *W* = {*c*,*d*}. The corresponding *TW*-algorithm will have the form

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

Table 10

As mentioned before, the only consistent (true) decision rule is the rule number 1, i.e. $a_1\ b_0 \longrightarrow c_2\ d_1$. Hence the positive region of the *TW*-algorithm consists only of this rule. In order to see whether the attribute *a* or *b* is dispensable or not we have to drop each of the attributes and check whether the positive region of the algorithm has changed or not, which is demonstrated below.

Removing attribute *a* we get the same positive region

U	*b*	*c*	*d*
1	0	2	1
4	2	2	1
2	1	0	1
3	1	2	0
5	2	0	0

Table 11

whereas when removing attribute *b* we changed the positive region, which is now the empty set, because all decision rules in the algorithm are inconsistent (false).

U	*a*	*c*	*d*
1	1	2	1
4	1	2	1
2	2	0	1
3	2	2	0
5	1	0	0

Table 12

Hence the attribute *a* is dispensable, whereas the attribute *b* is the core and the reduct of the algorithm and consequently the reduced ~~for~~ from of this algorithm is as follows

U	*b*	*c*	*d*
1	0	2	1
4	2	2	1
2	1	0	1
3	1	2	0
5	2	0	0

Table 13

This means ~~the~~ that the algorithm

$$a_1\, b_0 \longrightarrow c_2\, d_1$$

$$a_1\, b_2 \longrightarrow c_2\, d_1$$

$$a_2\, b_1 \longrightarrow c_0\, d_1$$

$$a_2\, b_1 \longrightarrow c_2\, d_0$$

$$a_1\, b_2 \longrightarrow c_0\, d_0$$

has only one reduced form as shown below

$$b_0 \longrightarrow c_2\, d_1$$

$$b_2 \longrightarrow c_2\, d_1$$

$$b_1 \longrightarrow c_0\, d_1$$

$$b_1 \longrightarrow c_2\, d_0$$

$$b_2 \longrightarrow c_0\, d_0$$

■

11. Reduction of Decision Rules

The purpose of this section is to show how the decision logic can be used ~~to~~ for 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. This idea corresponds exactly to that discussed earlier in section 3.4.

Before we give the necessary definitions, let us first introduce auxiliary denotation s. 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 $\phi \longrightarrow \psi$ be a PQ-rule, and let $a \in P$. We will say that the attribute a is *dispensable* in the rule $\phi \longrightarrow \psi$ if and only if

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

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

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

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

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

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

One can easily verify, that this idea corresponds exactly to that introduced in chapter 3, and that the following theorem is true.

Proposition 8

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

where $RED\ (P \longrightarrow Q)$ is the set of all reducts of $(P \longrightarrow Q)$. ∎

Example 8

As we already mentioned we are going now to eliminate unnecessary conditions in each decision rule of a decision

[illegible]

algorithm separately, i.e. compute the ~~core~~ core and reducts of each decision rule in the algorithm.

There are two possibilities available at the moment. First we may reduce the algorithm, i.e. drop all dispensable condition attributes in the whole algorithm and afterwards reduce each decision rule in the reduced algorithm, i.e. drop all unnecessary conditions in each rule of the algorithm. The second option consists in reduction at the very beginning decision rules, without elimina~~tion~~ ng attributes from the whole algorithm.

Let us ~~first~~ discuss the first option, and as an example consider the *KR*-system

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

Table 14

and the *PQ*-algorithm in the system shown below, where $P = \{a, b, c\}$ and $Q = \{d, e\}$ are condition and decision attributes respectively - as in Example 4.

$$a_1\ b_0\ c_2 \longrightarrow d_1\ e_1$$

$$a_2\ b_1\ c_0 \longrightarrow d_1\ e_0$$

$$a_2\ b_1\ c_2 \longrightarrow d_0\ e_2$$

$$a_1\ b_2\ c_2 \longrightarrow d_1\ e_1$$

$$a_1\ b_2\ c_0 \longrightarrow d_0\ e_2$$

We have to eliminate the unnecessary conditions in each rule of the algorithm separately.

Let us start with the first rule $a_1\ b_0\ c_2 \longrightarrow d_1\ e_1$. The core of this rule is the the empty set, because a, b and c are dispensable in the rule, i.e. the following decision rules $b_0\ c_2 \longrightarrow d_1\ e_1$, $a_1\ c_2 \longrightarrow d_1\ e_1$ and $a_1\ b_0 \longrightarrow d_1\ e_1$ are true. In other words either of the conditions $b_0\ c_2$, $a_1\ c_2$ or $a_1\ b_0$ uniquely determine the decision $d_1\ e_1$. There are two reducts of the rule namely $\{b\}$ and $\{a,c\}$. Hence the rule $a_1\ b_0\ c_2 \longrightarrow d_1\ e_1$ can be replaced by either one of rules $b_0 \longrightarrow d_1\ e_1$ or $a_1\ c_2 \longrightarrow d_1\ e_1$.

Each of the remaining four rules have one core attribute c, and consequently two reducts $\{a,\ c\}$ and $\{b,\ c\}$. Thus, for example the second rule, $a_2\ b_1\ c_0 \longrightarrow d_1\ e_0$ has two reduced forms $a_2\ c_0 \longrightarrow d_1\ e_0$ and $b_1\ c_0 \longrightarrow d_1\ e_0$.

Let us summarize the above considerations in tabular form. Table 15 contains cores of each decision rule.

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

Table 15

In Table 16 all reduced decision rules are listed.

U	*a*	*b*	*c*	*d*	*e*
1	-	0	-	1	1
1′	1	-	2	1	1
2	2	-	0	1	0
2′	-	1	0	1	0
3	2	-	2	0	2
3′	-	1	2	0	2
4	1	-	2	1	1
4′	-	2	2	1	1
5	1	-	0	0	2
5′	-	2	0	0	2

Table 16

Because each decision rule in the algorithm have two reduced forms, hence in order to simplify the algorithm we have to choose one of them, and as a result we get the algorithm with reduced decision rules as shown for example in Table 17.

U	a	b	c	d	e
1	-	0	-	1	1
2′	-	1	0	1	0
3	2	-	2	0	2
4′	-	2	2	1	1
5	1	-	0	0	2

Table 17

which can be also presented in the *DL*-language as

$$b_0 \longrightarrow d_1\ e_1$$

$$b_2\ c_2 \longrightarrow d_1\ e_1$$

$$b_1\ c_0 \longrightarrow d_1\ e_0$$

$$a_2\ c_2 \longrightarrow d_0\ e_2$$

$$a_1\ c_0 \longrightarrow d_0\ e_2$$

Note that rules 1′ and 4 are identical, hence choosing one ~~on~~ of the rules we obtain decision algorithm with smaller number of decision rules, as shown for example in Table 18

U	a	b	c	d	e
1′	1	-	2	1	1
2′	-	1	0	1	0
3	2	-	2	0	2
5	1	-	0	0	2

Table 18

or in decision logic notation

$$a_1\ c_2 \longrightarrow d_1\ e_1$$

$$b_1\ c_0 \longrightarrow d_1\ e_0$$

$$a_2\ c_2 \longrightarrow d_0\ e_2$$

$$a_1\ c_0 \longrightarrow d_0\ e_2$$

We may also first reduce the algorithm and then reduce further decision rules in the reduced algorithm. In this case the above example of decision algorithm would be reduced as follows:

As already shown in Example 6 the algorithm has two reduced forms

$$a_1\ c_2 \longrightarrow d_1\ e_1$$

$$a_2\ c_0 \longrightarrow d_1\ e_0$$

$$a_2\ c_2 \longrightarrow d_0\ e_2$$

$$a_1\ c_0 \longrightarrow d_0\ e_2$$

and

$$b_0\ c_2 \longrightarrow d_1\ e_1$$

$$b_1\ c_0 \longrightarrow d_1\ e_0$$

$$b_1\ c_2 \longrightarrow d_0\ e_2$$

$$b_2\ c_2 \longrightarrow d_1\ e_1$$

$$b_2\ c_0 \longrightarrow d_0\ e_2$$

which can be presented in tabular form

U	a	c	d	e
4	1	2	1	1
2	2	0	1	0
3	2	2	0	2
5	1	0	0	2

Table 19

and

U	~~a~~ b	c	d	e
1	0	2	1	1
4	2	2	1	1
2	1	0	1	0
3	1	2	0	2
5	2	0	0	2

Table 20

All decision rules in the decision algorithm shown in Table 19 are already reduced, hence the algorithm cannot be simplified further.✓In the decision algorithm in Table 20 rules 2, 3, 4 and 5 are are already reduced, whereas in the rule 1 the condition c_2 can be eliminated and the rule will have the form $b_0 \longrightarrow d_1\ e_1$. Thus the algorithm takes the form

$$b_0 \longrightarrow d_1\ e_1$$

$$b_1\ c_0 \longrightarrow d_1\ e_0$$

$$b_1\ c_2 \longrightarrow d_0\ e_2$$

$$b_2\ c_2 \longrightarrow d_1\ e_1$$

$$b_2\ c_0 \longrightarrow d_0\ e_2$$

■

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 $\phi \longrightarrow \psi$ 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. itself.

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 $\phi \longrightarrow \psi$ 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~~ last step removes all superfluous decision rules from the algorithm.

The example which follows will depict the above defined concepts.

Example 9

Spouse 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 21

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 22 can be simplified as shown in Table 23 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 23

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 24.

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 24

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 25

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) $a_1b_0 \longrightarrow e_1$

2) $b_0d_1 \longrightarrow e_1$

$a_1d_0 \longrightarrow e_1$

or

$b_0d_1 \vee a_1d_0 \longrightarrow e_1$

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

$a_0 \longrightarrow e_0$

$b_1d_1 \longrightarrow e_0$

or

$a_0 \vee b_1d_1 \longrightarrow e_0$

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

$d_2 \longrightarrow e_2$

Finally we get two minimal decision algorithms

$a_1b_0 \longrightarrow e_1$

$a_0 \longrightarrow e_0$

$b_1d_1 \longrightarrow e_0$

$d_2 \longrightarrow e_2$

and

$b_0d_1 \longrightarrow e_1$

$a_1d_0 \longrightarrow e_1$

$a_0 \longrightarrow e_0$

$b_1d_1 \longrightarrow e_0$

$d_2 \longrightarrow e_2$

The combined form of these algorithms are

$$a_1 b_0 \longrightarrow e_1$$

$$a_0 \vee b_1 d_1 \longrightarrow e_0$$

$$d_2 \longrightarrow e_2$$

and

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

$$a_0 \vee b_1 d_1 \longrightarrow e_0$$

$$d_2 \longrightarrow e_2$$

■

Summary

All the concepts introduced in previous chapters can be also worded in terms of decision logic. As a result we obtain much simpler computation algorithms for knowledge reduction, dependencies, etc.

It is worthwhile to stress that the basic role is played here by a relationship between truth and indiscernibiliy, strictly speaking the concept of truth is replaced in our approach by the concept of indiscernibility. The relationship between truth and indiscernibility, is of principal importance from algorithmic point of view, because it allows to replace checking whether some implications (decision rules) are true or not - by investigation of indiscernibility of some elements of the universe. Note also the highly parallel structure of algorithms in this setting.

Excercise

1. Give meaning of the following formulas

$$a_1 \vee (b_0 \longrightarrow c_1)$$

$$a_2 \vee (b_0 \wedge c_0)$$

$$a_1 \longrightarrow (b_2 \longrightarrow c_1)$$

$$a_1 \equiv (b_1 \wedge c_0)$$

in the *KR*-systems below

U	a	b	c
1	0	1	2
2	2	0	1
3	0	1	1
4	2	2	1
5	0	1	0

Table 26

U	a	b	c
1	2	0	2
2	1	2	1
3	0	2	1
4	0	1	1
5	0	2	0

Table 27

2. Compute for the formulas given in Exercise 1, and Tables 26 and 27, P-normal forms for $P = \{a, b, c\}$, $\{b, c\}$, $\{a, c\}$, $\{a, b\}$, $\{a\}$, $\{b\}$ and $\{c\}$.

3. Give PQ-decision algorithms for P and Q as listed below

P	Q
$\{a, b\}$	$\{c\}$
$\{c\}$	$\{a, b\}$
$\{a\}$	$\{b\}$
$\{a, c\}$	$\{b\}$
$\{a, c\}$	$\{b, c\}$
$\{a, b, c\}$	$\{a, b, c\}$

which are admissible in Table 26 (27).

4. Check which algorithm are consistent and inconsistent. For inconsistent algorithms give the degree of inconsistency. (Degree of dependence between condition and decision algorithms).

5. Compute reduced forms for each algorithm in Exercise 3.

6. Compute minimal forms for each algorithm in Exercise 3.

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. and Tuschnik, H. P. (1990). An Aximatisation of the Logic with Rough Quantifiers. *Zeitschrift feur Grundlagen der Mathematik und 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. (1987). 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 Scme Modal Knowledge Representation Systems and the Logic NIM of Non-deterministic 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.

P A R T II

A P P L I C A T I O N S

8. DECISION MAKING

1. Introduction

In this chapter we are going to illustrate in detail some of the basic ideas discussed in the previous chapters.

Many problems related with decision making are of the following nature: given a decision table - find all minimal decision algorithms associated with the table.

It is worthwhile to mention that this problem can be also seen as learning from examples, or as it is often called in the AI folklore - as decision rule generation from examples. We shall however stick to our terminology, since it seems more consistent and self explanatory.

There are many methods available to solve this problem (cf. Quinlan (1979), Cendrowska (1987)). Recently several methods based on rough set philosophy ~~has~~ have been proposed (cf. Mrózek (1987), Grzymała-Busse (1988), Pawlak (1988), Boryczka et al. (1988), Słowinski (1988), Ziarko (1987)), which seem to be superior to other methods.

The rough set approach offers all solutions to the problem of decision table simplification and yields fast computer algorithms, and has found real life application in various fields (cf.for example Fibak et al. (1986), Krysiński (1990), Pawlak et al. (1986), Słowiński et al. (1989)).

2. Optician's Decisions Table

The example, which follows (cf. Cendrowska (1987)), concerns optician's decisions as to whether or not a patient is suitable for contact lenses wear. The set of all possible decisions is listed in Table 1.

U	*a*	*b*	c	*d*	*e*
1	1	1	2	2	1
2	1	2	2	2	1
3	2	1	2	2	1
4	3	1	2	2	1
5	1	1	1	2	2
6	1	2	1	2	2
7	2	1	1	2	2
8	2	2	1	2	2
9	3	2	1	2	2
10	1	1	1	1	3
11	1	1	2	1	3
12	1	2	1	1	3
13	1	2	2	1	3
14	2	1	1	1	3
15	2	1	2	1	3
16	2	2	1	1	3
17	2	2	2	1	3
18	2	2	2	2	3
19	3	1	1	1	3
20	3	1	1	2	3
21	3	1	2	1	3
22	3	2	1	1	3
23	3	2	2	1	3
24	3	2	2	2	3

Table 1

The table is in fact a decision table in which *a*, *b*, *c*, and *d* are condition attributes whereas *e* is a decision attribute.

The attribute *e* represents optician's decisions, which are the following:

1. Hard contact lenses
2. Soft contact lenses
3. No contact lenses

These decisions are based on some facts concerning the patient, which are expressed by the condition attributes given below together with corresponding attribute values.

a - Age

1 - *young*

2 - *pre-presbyopic*

3 - *presbyopic*

b - spectacle

1 - *myope*

2 - *hypermetrope*

c - astigmatic

1 - *no*

2 - *yes*

d - tear production rate

1 - *reduced*

2 - *normal*

The problem we are going to discuss here is the elimination of conditions from a decision table, which are unnecessary to make decisions specified in the table.

In our approach the problem can be expressed as simplification of the decision table (described in Chapter 6) or finding of minimal decision algorithms associated with the decision table (described in Chapter 7).

The first method is based on some operations on indiscernibility (equivalence) relations and has algebraic flavor, whereas the second - is embedded in logic. The algebraic approach to decision table simplification is straightforward, however, algorithms based on this method are sometimes not very efficient. The logical approach is easier to implement and gives faster algorithms then the algebraic method, but its intuitive meaning seems to be not so obvious as in the previous case. Therefore, for the sake of clarity we recommend combination of both methods, employing decision tables notation and logical algorithms. This means that we treat decision table as a set of formulas, which can be treated in logical way. For example row 1 in Table 1 can be considered as the formula $a_1\ b_1\ c_2\ d_2 \longrightarrow e_1$. In order to see whether this formula is true or not we could check the inclusion $|a_1\, b_1\, c_2\, d_2| \le |e_1|$, however this procedure is rather inefficient. It is better to employ in this case Proposition 7.6, i.e.,to check whether there is in the table another implication with the same predecessor and different successor. Because this is not the case, the implication is true. The above presented approach will be used in the forthcoming chapters.

3. Simplification of Decision Table

In view of what has been said so far our problem consists ~~in~~ of :

a) checking whether elementary categories of decision attributes (decision categories) can be defined (expressed) in terms of basic categories defined by the set of all condition attributes (condition categories).

b) reduction of the set of condition categories necessary to define decision categories.

Obviously problem a) reduces in our case to the question whether decision attribute e depends on condition attributes a, b, c and d, i.e., whether the dependency $\{a, b, c, d\} \Rightarrow \{e\}$ holds. Because all conditions in the table are different, hence according to Propositions 7.6 and 7.7, there is total dependency between condition and decision attributes, i.e. the dependency $\{a, b, c, d\} \Rightarrow \{e\}$~~,~~ is valid. ~~That~~ This means that elementary categories of the attribute e (decision categories) can be uniquely defined by the categories of condition attributes (condition categories), or in other words, ~~that~~ the values of decision attribute e are uniquely determined by means of values of condition attributes a, b, c, and d.

As to the problem b) one can compute ~~that~~ the set of condition attributes is e-independent, ~~what~~ means that none of the condition attribute can be eliminated, as a whole, without destroying the classificatory properties of the decision table, that is removing any of the condition attributes from Table 1 makes this decision table inconsistent. Let us analyze this in detail.

Removing attribute *a* in Table 1 we get the following decision table

U	*b*	c	*d*	*e*
1	1	2	2	1
2	2	2	2	1
3	1	2	2	1
4	1	2	2	1
5	1	1	2	2
6	2	1	2	2
7	1	1	2	2
8	2	1	2	2
9	2	1	2	2
10	1	1	1	3
11	1	2	1	3
12	2	1	1	3
13	2	2	1	3
14	1	1	1	3
15	1	2	1	3
16	2	1	1	3
17	2	2	1	3
18	2	2	2	3
19	1	1	1	3
20	1	1	2	3
21	1	2	1	3
22	2	1	1	3
23	2	2	1	3
24	2	2	2	3

Table 2

which is inconsistent because we have in Table 2 the following pairs of inconsistent decision rules

$b_2\ c_2\ d_2 \longrightarrow e_1$ (row 2)

$b_2\ c_2\ d_2 \longrightarrow e_3$ (rows 18 and 24)

This means that the condition b_2 c_2 d_2 does not determine uniquely ~~n~~either decision e_1 ~~nor~~ or e_3. Thus the attribute a ~~can not~~ cannot be dropped.

Similarly removing attribute attribute b we get the following decision table

U	a	c	d	e
1	1	2	2	1
2	1	2	2	1
3	2	2	2	1
4	3	2	2	1
5	1	1	2	2
6	1	1	2	2
7	2	1	2	2
8	2	1	2	2
9	3	1	2	2
10	1	1	1	3
11	1	2	1	3
12	1	1	1	3
13	1	2	1	3
14	2	1	1	3
15	2	2	1	3
16	2	1	1	3
17	2	2	1	3
18	2	2	2	3
19	3	1	1	3
20	3	1	2	3
21	3	2	1	3
22	3	1	1	3
23	3	2	1	3
24	3	2	2	3

Table 3

In this table the following pairs of decision rules are inconsistent

(i) $a_2\ c_2\ d_2 \longrightarrow e_1$ (rule 3)

$a_2\ c_2\ d_2 \longrightarrow e_3$ (rule 18)

(ii) $a_3\ c_2\ d_2 \longrightarrow e_1$ (rule 4)

$a_3\ c_2\ d_2 \longrightarrow e_3$ (rule 24)

(iii) $a_3\ c_1\ d_2 \longrightarrow e_2$ (rule 9)

$a_3\ c_1\ d_2 \longrightarrow e_3$ (rule 20).

This means that pairs of decisions e_1, e_3 and e_2,e_3 can not be distinguished by means of conditions $a_3\ c_1\ d_2$.

Similarly removing the attribute *c* from Table 1 we get Table 4

U	*a*	*b*	*d*	*e*
1	1	1	2	1
2	1	2	2	1
3	2	1	2	1
4	3	1	2	1
5	1	1	2	2
6	1	2	2	2
7	2	1	2	2
8	2	2	2	2
9	3	2	2	2
10	1	1	1	3
11	1	1	1	3
12	1	2	1	3
13	1	2	1	3
14	2	1	1	3
15	2	1	1	3
16	2	2	1	3
17	2	2	1	3
18	2	2	2	3
19	3	1	1	3
20	3	1	2	3
21	3	1	1	3
22	3	2	1	3
23	3	2	1	3
24	3	2	2	3

Table 4

in which the following pairs of rules are inconsistent

(i) $a_1\ b_1\ d_2 \longrightarrow e_1$ (rule 1)

$a_1\ b_1\ d_2 \longrightarrow e_2$ (rule 5)

(ii) $a_1\ b_2\ d_1 \longrightarrow e_1$ (rule 2)

$a_1\ b_2\ d_1 \longrightarrow e_2$ (rule 6)

(iii) $a_2\ b_1\ d_2 \longrightarrow e_1$ (rule 3)

$a_2\ b_1\ d_2 \longrightarrow e_2$ (rule 7)

(iv) $a_3\ b_1\ d_2 \longrightarrow e_1$ (rule 4)

$a_3\ b_1\ d_2 \longrightarrow e_3$ (rule 20)

(v) $a_2\ b_2\ d_2 \longrightarrow e_2$ (rule 8)

$a_3\ b_2\ d_2 \longrightarrow e_3$ (rule 18)

(vi) $a_3\ b_2\ d_2 \longrightarrow e_2$ (rule 9)

$a_3\ b_2\ d_2 \longrightarrow e_3$ (rule 24)

Finally dropping the attribute d in Table 1 we get ~~another~~ Table 5

U	a	b	c	e
1	1	1	2	1
2	1	2	2	1
3	2	1	2	1
4	3	1	2	1
5	1	1	1	2
6	1	2	1	2
7	2	1	1	2
8	2	2	1	2
9	3	2	1	2
10	1	1	1	3
11	1	1	2	3
12	1	2	1	3
13	1	2	2	3
14	2	1	1	3
15	2	1	2	3
16	2	2	1	3
17	2	2	2	3
18	2	2	2	3
19	3	1	1	3
20	3	1	1	3
21	3	1	2	3
22	3	2	1	3
23	3	2	2	3
24	3	2	2	3

Table 5

in which there are the following pairs of inconsistent rules

(i) $a_1\ b_1\ c_2 \longrightarrow e_1$ (rule 1)

$a_1\ b_1\ c_2 \longrightarrow e_{\cancel{2}3}$ (rule ~~3~~ 11)

(ii) $a_1\ b_2\ c_2 \longrightarrow e_1$ (rule 2)

$a_1\ b_2\ c_2 \longrightarrow e_3$ (rule 13)

(iii) $a_2\ b_1\ c_2 \longrightarrow e_1$ (rule 3)

$a_2\ b_1\ c_2 \longrightarrow e_1$ (rule 15)

(iv) $a_3\ b_1\ c_2 \longrightarrow e_1$ (rule 4)

$a_3\ b_1\ c_2 \longrightarrow e_3$ (rule 21)

(v) $a_1\ b_1\ c_1 \longrightarrow e_2$ (rule 5)

$a_1\ b_1\ c_1 \longrightarrow e_3$ (rule 10)

(vi) $a_1\ b_2\ c_1 \longrightarrow e_2$ (rule 6)

$a_1\ b_2\ c_1 \longrightarrow e_3$ (rule 12)

(vii) $a_2\ b_1\ c_1 \longrightarrow e_2$ (rule 7)

$a_2\ b_1\ c_1 \longrightarrow e_3$ (rule 14)

(viii) $a_2\ b_2\ c_1 \longrightarrow e_2$ (rule 8)

$a_2\ b_2\ c_1 \longrightarrow e_3$ (rule 16)

(ix) $a_3\ b_2\ c_1 \longrightarrow e_2$ (rule 9)

$a_3\ b_2\ c_1 \longrightarrow e_3$ (rule 22)

As we can see from the analysis none of the condition attributes can be removed from Table 1. Hence the set of condition attribute~~s~~ is e-independent.

It is interesting to note, ~~that~~ dropping of various condition attributes we introduce~~s~~ inconsistency of different "depth". We shall discuss this problem in more detail in the last ~~least~~ section of this chapter.

Coming back to our initial problem of simplification of decision table we have now to check whether we can eliminate some elementary condition categories, i.e., some superfluous values of condition attributes in Table 1. To this end first we have to compute *e*-cores of each elementary condition category. In other words we have to check which values of condition attributes are indispensable in order to discern values of decision attribute. Because value core is the set of all indispensable values with respect to *e*, hence in order to check whether a specific value is dispensable or not (i.e. whether it belongs to the core) we have to remove the value from the table and see if the remaining values in the same row uniquely determine decision attribute value in this row. If not this value belongs to the core. Speaking in logical terms we have to compute core values of each decision rule in the decision table, i.e. find all those condition attribute values in the decision rule which make the decision rule false.

For example in the first decision rule $a_1\ b_1\ c_2\ d_2 \longrightarrow e_1$ values c_2 and d_2 are core values because the rules

$$b_1\ c_2\ d_2 \longrightarrow e_1$$

$$a_1\ c_2\ d_2 \longrightarrow e_1$$

are true, whereas the rules

$$a_1\ b_1\ d_2 \longrightarrow e_1$$

$$a_1\ b_1\ c_2 \longrightarrow e_1$$

are false. (see Proposition 7.6).

All core values of each decision rule in Table 1 are given in Table 6.

U	a	b	c	d	e
1	-	-	2	2	1
2	1	-	2	2	1
3	-	1	2	2	1
4	-	1	2	2	1
5	-	-	1	2	2
6	-	-	1	2	2
7	2	-	1	2	2
8	-	-	1	2	2
9	-	2	1	2	2
10	-	-	-	1	3
11	-	-	-	1	3
12	-	-	-	1	3
13	-	-	-	1	3
14	-	-	-	1	3
15	-	-	-	1	3
16	-	-	-	1	3
17	-	-	-	-	3
18	2	2	2	-	3
19	-	-	-	-	3
20	3	1	1	-	3
21	-	-	-	1	3
22	-	-	-	1	3
23	-	-	-	-	3
24	3	2	2	-	3

Table 6

Now we can easily compute all *e*-reducts of condition elementary categories, or what is the same, reduct values of condition attributes of each decision rule. From Proposition 8 follows that in order to find reducts of decision rules we have to add to the core values of each decision rule such values of condition attributes of the rule, that the predecessor of the rule is independent and the whole rule is true.

For example the first rule $a_1\ b_1\ c_2\ d_2 \longrightarrow e_1$ has two reducts $a_1\ c_2\ d_2 \longrightarrow e_1$ and $b_1\ c_2\ d_2 \longrightarrow e_1$, since both decision rules are true and predecessor of each decision rule is independent.

Reduts of each decision rule in Table 1 are listed in Table 7 which follows.

U	*a*	*b*	*c*	*d*	*e*
1	x	1	2	2	1
1′	1	x	2	2	1
2	1	x	2	2	1
3	x	1	2	2	1
4	x	1	2	2	1
5	1	x	1	2	2
6	1	x	1	2	2
6′	x	2	1	2	2
7	2	x	1	2	2
8	2	x	1	2	2
8′	x	2	1	2	2
9	x	2	1	2	2
10	x	x	x	1	3
11	x	x	x	1	3
12	x	x	x	1	3
13	x	x	x	1	3
14	x	x	x	1	3
15	x	x	x	1	3
16	x	x	x	1	3
17	x	x	x	1	3
17′	2	2	2	x	3
18	2	2	2	x	3
19	3	1	1	x	3
19′	x	x	x	1	3
20	3	1	1	x	3
21	x	x	x	1	3
22	x	x	x	1	3
23	x	x	x	1	3
23′	3	2	2	x	3
24	3	2	2	x	3

Table 7

It can be seen from the table that decision rules 1, 6, 8, 17, 19 and 23 have two reducts, and the remaining decision rules have one reduct each.

In order to find the minimal decision algorithm we have to remove from the table all superfluous decision rules (see section 7.10).

From Table 7 we see that in the first decision class decision rules 1 and 1′ are superfluous, because these rule are identical ~~with~~ to rules 2, 3 and 4 respectively. In the second decision class rules 6 and 5 are identical, rules 6′, 8′ and 9 are also identical and so are rules 8 and 7, hence it is enough to have one representative rule from each group of identical rules. Similarly we can remove superfluous decision rules from decision class three and we obtain the following set of minimal decision rules (minimal decision algorithm) as shown in Table 8.

U	*a*	*b*	*c*	*d*	*e*
1′,2	1	x	2	2	1
1,3,4	x	1	2	2	1
5,6	1	x	1	2	2
7,8	2	x	1	2	2
6′,8′,9	x	2	1	2	2
10-17,19′					
21,22,23	x	x	x	1	3
17′,18	2	2	2	x	3
19,20	3	1	1	x	3
23′,24	3	2	2	x	3

Table 8

Let us note that there is only one minimal solution to this problem.

Because enumeration of the decision rules is not essential we can get from Table 8 the following Table 9.

U	*a*	*b*	*c*	*d*	*e*
1	1	x	2	2	1
2	x	1	2	2	1
3	1	x	1	2	2
4	2	x	1	2	2
5	x	2	1	2	2
6	x	x	x	1	3
7	2	2	2	x	3
8	3	1	1	x	3
9	3	2	2	x	3

Table 9

Crosses in the table denote "don't care" values of attributes. What we have obtained finally is the minimal set of decision rules (minimal decision algorithm) which is equivalent to the original table, as far as the decisions are concerned. That means that in the simplified table only the minimal set of conditions, necessary to make decisions specified in the table, are included.

It is worthwhile to stress once more that there is only one solution to the discussed problem. In general many solutions are possible, as we will see in the next chapter.

4. Decision Algorithm

As we already stated before we are not interested in decision tables but ~~rather~~ in decision algorithms, and tables are used only as a convenient way of notation for decision algorithms. Tabular ~~form~~ for of presentation of decision algorithms, allows us easy ~~computation~~ of truth of decision rules ~~and~~ provides us consequently ~~easy~~ way of decision algorithms simplification. We could for example start simplification of Table 1, by replacing all rows in the table by appropriate decision rules and by application of Proposition 7.6 obtain final form of minimal decision algorithm. The tabular notation of decision algorithms is however more "transparent" and leads to simpler computer algorithms (which can be also implemented in parallel).

Thus our final result presented in Table 9 can be rewritten as minimal decision algorithm in normal form:

$a_1\ c_2\ d_2 \longrightarrow e_1$

$b_1\ c_2\ d_2 \longrightarrow e_1$

$a_1\ c_1\ d_2 \longrightarrow e_2$

$a_2\ c_1\ d_2 \longrightarrow e_2$

$b_{\cancel{1}\,2}\ c_1\ d_2 \longrightarrow e_2$

$d_1 \longrightarrow e_3$

$a_2\ b_2\ c_2 \longrightarrow e_3$

$a_3\ b_1\ c_1 \longrightarrow e_3$

$a_3\ b_2\ c_2 \longrightarrow e_3$

Combining all decision rules for one decision class we get the following decision algorithm:

$(a_1 \vee b_1)\ c_2\ d_2 \longrightarrow e_1$

$(a_1 \vee a_2 \vee b_{\cancel{1}\,2})\ c_1\ d_2 \longrightarrow e_2$

$d_1 \vee (a_3\ b_1\ c_1) \vee ((a_2 \vee a_3)\ b_2\ c_2) \longrightarrow e_3$

5. The Case of Incomplete Information

It may happen that for some reason values of some attributes are not available while making decisions. The problem arises how this lack of information would affect the ability to make decisions. In order to answer this question let us come back to the problem of elimination of attributes considered in section 3.

Lack of attribute *a* makes only two decision rules inconsistent

$b_2\ c_2\ d_2 \longrightarrow e_1$ (row 2)
$b_2\ c_2\ d_2 \longrightarrow e_3$ (rows 18 and 24)

Leaving ~~remaining~~ the rest of decision rules unaffected (true) and constitute positive region of the decision algorithm. Hence the degree of dependency $\{bcd\} \Rightarrow \{e\}$ is $21/24 = 7/8$. The attribute *b* is more significant for its removal causes six decision rules to be inconsistent thus produces the degree of dependency $\{acd\} \Rightarrow \{e\}$ $18/24 = 3/4$. Removing attribute *c* makes twelve rules inconsistent and consequently the dependency $\{abd\} \Rightarrow \{e\}$ is valid in the degree $12/24 = 1/2$. The last ~~least~~ attribute *d*, when dropped causes eighteen rules to be inconsistent, which yields a degree of dependency $\{abc\} \Rightarrow \{e\}$ equal to $6/24 = 1/4$; thus the attribute *d* is the most significant one, since without the attribute only six out of twenty four decision rules remain true, i.e., enable us to make consistent decisions. Whereas without the attribute *a* twenty one ~~two~~ out of twenty four decision rules are consistent, so lack of this attribute affects rather not very much the positive region of the decision algorithm, i.e., its ability to make consistent decisions.

Summary

The example considered in this chapter shows how knowledge about certain domain can be analyzed and reduced using rough sets approach. The main problem in this case was to reduce the original knowledge (decision Table 1) in such a way that decisions specified in Table 1 can be made using minimal set of conditions.

In summary, to simplify a decision table we should first find reducts of condition attributes, remove duplicate rows (decision rules) and then find value-reducts of condition attributes and again, if necessary, remove redundant decision rules. This method leads to a simple algorithm for decision table simplification or generation of minimal decision algorithm from examples, which, according to our experiments, out performs other methods, in ~~terms~~ of achievable degree in the number of conditions and what's more , + gives all possible solutions to the problem.

Let us stress once more that minimal decision algorithm corresponding to Table 1 could be also obtained without decision tables notation used in the presented example - employing directly the logical tools introduced in Chapter 7, however as we mentioned in the beginning of ~~this~~ chapter the tabular notation ~~seems~~ to be more clear than the pure logical notation and what's ~~is~~ more important - is easer to implement, because tables are more convenient data structures for computer implementation then logical formulas~~,~~ and also the "inference rule " (proposition 7.6) & seems to be much more useful when data have tabular form, than the ~~classical~~ rules of inference.

Exercises

1.Check whether the following decision rules are consistent

$$b_1\ c_2 \longrightarrow e_1$$

$$a_2\ c_2\ d_2 \longrightarrow e_1$$

$$a_3\ d_2 \longrightarrow e_2$$

$$b_1 \longrightarrow e_3$$

$$a_3\ b_1\ d_1 \longrightarrow e_3$$

or not in Table 1. Use both semantic and syntactic methods, i.e. compute meanings of decision rules according to Proposition 7.1 (e) and Proposition 7.6.

2. Compute degree of dependency for the following

$a \Rightarrow e$	$ab \Rightarrow e$	$ac \Rightarrow e$
$b \Rightarrow e$	$bc \Rightarrow e$	$bc \Rightarrow e$
$c \Rightarrow e$	$ad \Rightarrow e$	$cd \Rightarrow e$
$d \Rightarrow e$	$cd \Rightarrow e$	$bd \Rightarrow e$

For simplicity we omitted parenthesis {, } in decision rules.

References

Boryczka, M. and Słowiński, R. (1988). Derivation of Optimal Decision Algorithms from Decision Tables Using Rough Sets. *Bull. Polish Acad. Sci. Tech.*, 36, pp. 252-260.

Cendrowska, J. (1987). PRISM: An Algorithm for Inducing Modular Rules. *Int. J. Man-Machine Studies*, 27, pp. 349-370.

Fibak, J., Slowinski, K. and Słowiński, R. (1986). Rough Sets Based Decision Algorithm for Treatment of Duodenal Ulcer by HSV. *Proc. 6th Internal Workshop on Expert Systems and their Applications.* Avignon, France, pp. 587-599.

Grzymała-Busse, J. (1988). Knowledge Acquisition under Uncertainty - a Rough Set Approach. *Journal of Intelligent and Robotic Systems*, 1, pp. 3-16.

Krysiński, J. (1990). Rough Sets Approach to Analysis of Relationship Between Structure and Activity of Quaternary Imidazolium Compounds. *Arzeneimittel-Forschung/Drug Research.* (to appear).

Mrózek, A. (1987). Information Systems and some Aspects of Expert Systems Realization. *Proc. 7th International Workshop on Expert Systems and their Applications, Avignon, France*, pp. 597-611.

Pawlak, Z., Słowiński, K. and Słowiński, R. (1986). Rough Classification of Patients after Highly Selective Vagotomy for Duodenal Ulcer. *International Journal of Man-Machine Studies*, 24, pp. 413-433.

Pawlak, Z. (1988). Rough Sets and their Applications. *Workshop "Mathematics and AI", Schloss Reiseburg, W. Germany.* II, pp. 543-572.

Słowiński, R., Słowiński, K. and Stefanowski, J. (1988). Rough set Approach to Analysis of Data from Pertioneal Lavage in Acute Pancreatitis. *Medical Information*, 13, pp. 143-159.

Słowiński, R. and Słowiński, K. (1989). Sensitivity Analysis of Rough Classification. *Internal Journal of Man-Machine Studies*, 30, pp. 245-248.

Quinlan, J. R. (1979). Discovering Rules from Large Collection of Examples: a Case Study. In Michie,D., Ed., *Expert Systems in the Micro-Electronic Age.* Edinburgh: Edinburgh University Press. pp. 165-201.

Ziarko, W. (1987). Reduction of Knowledge Representation. *Proc. 2nd ACN SIGART International Sym. on Methodologies for Intelligent Systems.* Charlotte, NC.(Colloquia Series). pp.~~???~~ ?

9. DATA ANALYSIS

1. Introduction

In this chapter we shall consider several examples of decision tables, which can be obtained as a result of observations or measurements, in contrast to decision tables considered in Chapter 8, which represented *explici~~te~~* knowledge of an agent or group of agents.

First we shall analyze generation of control algorithm for a rotary clinker kiln from observation of a stoker decisions who controls the kiln (cf. Mrózek (1988)). Our aim is to derive from this observations computer control algorithms, which ~~is~~ are able to mimic stoker's performance as kiln controller. We will see that this problem reduces to decision (control) algorithm derivation from observational data given in a form of a decision table.

This example illustrates several vital points connected with the application of rough set concept to the analysis of knowledge of a human operator and may serve as an illustration of wider class of similar tasks.

The next example considered in this chapter concerns data about patients suffering from a certain disease. This simple example is suppose to depict~~s~~ another situation when data is collected by observation or measurement. In reality, the application of rough set philosophy to this kind of problems is much more complicated and the reader is advised to see the papers of Pawlak et al.(1986) ~~et~~ and Słowiński et al.(1988) and also references given in the previous Chapter.

2. Decision Table as Protocol of Obsevations

In this section we will analyze stoker's decisions, while controlling the clinker kiln.

For our purpose, the details of cement kiln operation are not important and we give only brief information necessary to understand the forthcoming considerations. For much greater detail the reader is advised to consult the paper of Mrózek.

The aim of the stoker is to keep the kiln in a "proper" state, and what is the "proper" state is basically determined by his experience. To this end he has to control values of two parameters of the kiln, namely the kiln revolutions (KR) and the coal worm revolutions (CWR). The values of these parameters are set by the stoker upon observations of four parameters of the kiln, namely burning zone temperature (BZT), burning zone color (BZC), clinker granulation (CG), and kiln inside color (KIC). Thus actions of the stoker can be described by a decision table, where BZT, BZC, CG and KIC are condition attributes, and KR and CWR are decision attributes.

Each combination of condition attributes values corresponds to specific kiln state, and in each state of the kiln, appropriate action must be taken in order to obtain cement of required quality. The stoker has control of the rotation speed of the kiln, which can be set to two values, and the temperature of the kiln, which is set indirectly by control of heating coal consumption measured in revolution speed of the coal worm.

All attributes and their values are listed below. For the sake of simplicity we will denote attributes by lower case italic letters, as shown in what follows.

Thus we have the following condition attributes:

a - Burning Zone Temperature (BZT)
b - Burning Zone Color (BZC)
c - Clinker Granulation (in burning zone) (CG)
d - Kiln Inside Color (KIC)

The domains of these attributes are coded as follows:

Burning Zone Temperature

1 - (1380 - 1420°C)
2 - (1421 - 1440°C)
3 - (1441 - 1480°C)
4 - (1481 - 1500°C)

Burning Zone Color

1 - *scarlet*
2 - *dark pink*
3 - *bright pink*
4 - *very bright pink*
5 - *rose white*

Clinker Granulation

1 - *fines*
2 - *fines with small lumps*
3 - *granulation*
4 - *lumps*

Kiln Inside Color

1 - *dark streaks*
2 - *Indistinct dark streaks*
3 - *lack of dark streaks*

The decision attributes

e - Kiln Revolutions (KR)
f - Coal Worm Revolutions (CWR)

have the following values of attributes:

Kiln Revolutions

1 - 0,9 rpm
2 - 1,22 rpm

Coal Worm Revolutions

1 - 0 rpm
2 - 15 rpm
3 - 30 rpm
4 - 40 rpm

Let us note that in fact all attribute values are qualitative.

An exemplary list of decisions (protocol) taken by a stoker is shown in Table 1.

TIME	BZT *a*	BZC *b*	CG *c*	KIC *d*	KR *e*	CWR *f*
1	3	2	2	2	2	4
2	3	2	2	1	2	4
3	3	2	2	1	2	4
4	2	2	2	1	1	4
5	2	2	2	1	1	4
6	2	2	2	1	1	4
7	2	2	2	1	1	4
8	2	2	2	1	1	4
9	2	2	2	2	1	4
10	2	2	2	2	1	4
11	2	2	2	2	1	4
12	3	2	2	2	2	4
13	3	2	2	2	2	4
14	3	2	2	3	2	3
15	3	2	2	3	2	3
16	3	2	2	3	2	3
17	3	3	2	3	2	3
18	3	3	2	3	2	3
18	3	3	2	3	2	3
19	3	3	2	3	2	3
20	4	3	2	3	2	3
21	4	3	2	3	2	3
22	4	3	2	3	2	3
23	4	3	3	3	2	2
24	4	3	3	3	2	2
25	4	3	3	3	2	2
26	4	4	3	3	2	2
27	4	4	3	3	2	2
28	4	4	3	3	2	2
29	4	4	3	3	2	2
30	4	4	3	3	2	2
31	4	4	3	2	2	2

32	4	4	3	2	2	2
33	4	3	3	2	2	2
34	4	3	3	2	2	2
35	4	3	3	2	2	2
36	4	2	3	2	2	2
37	4	2	3	2	2	2
38	3	2	2	2	2	4
39	3	2	2	2	2	4
40	3	2	2	2	2	4
41	3	3	2	2	2	4
42	3	3	2	2	2	4
43	3	3	2	2	2	4
44	3	3	2	3	2	3
45	3	3	2	3	2	3
46	4	3	2	3	2	3
47	4	3	2	3	2	3
48	4	3	3	3	2	2
49	4	3	3	3	2	2
50	4	4	3	3	2	2
51	4	4	3	3	2	2
52	4	4	3	3	2	2

Table 1

As we mentioned already this table (protocol of stoker behavior) can be treated as a decision table in which *a*, *b*, *c* and *d* are condition attributes, whereas *e* and *f* are decision attributes.

The table describes actions undertaken by a sto~~c~~ker, during one shift, when specific conditions observed by the stoker have been fulfilled. Numbers given in the column TIME are in fact ID labels of moments of time, when the decisions took place, and form the universe of the decision table.

Because many decisions are the same, hence identical decision rules can be removed from the table and we obtain in this way Table 2.

U	*a*	*b*	*c*	*d*	*e*	*f*
1	3	3	2	2	2	4
2	3	2	2	2	2	4
3	3	2	2	1	2	4
4	2	2	2	1	1	4
5	2	2	2	2	1	4
6	3	2	2	3	2	3
7	3	3	2	3	2	3
8	4	3	2	3	2	3
9	4	3	3	3	2	2
10	4	4	3	3	2	2
11	4	4	3	2	2	2
12	4	3	3	2	2	2
13	4	2	3	2	2	2

Table 2

In Table 2 decisions are identified not by moments of time but by arbitrary integers (or could be labelled by any kind of identifiers), which form the new universe of the decision table.

The table describes knowledge of an experienced human operator of the kiln, which has been obtained by observation of stoker actions. Thus the knowledge of the operator has been obtained not by interviewing him, as it is usually the case in expert systems philosophy, but by observing his actions when controlling the kiln.

3. Derivation of Control Algorithms from Observation

We pass next to the analysis of the observed data describing stoker decisions, in order to obtain control algorithm of the kiln - based on stoker's experience and able to use his knowledge. More precisely the problems we are going to discuss in this section are consistency of stoker knowledge, reduction of his knowledge and control algorithm generation, from the observed stoker's behavior.

By consistency of knowledge in this particular case we mean functional dependency between conditions and actions, i.e., whether actions are uniquely determined by conditions. Reduction of knowledge consists in dropping all dispensable condition attributes and condition attribute values from the table and thus obtained minimal decision table is in fact the control (decision) algorithm, which can be simply implemented in any programming language, or directly in hardware.

Let us first consider the consistency problem. For this purpose we need to check whether the table is consistent or not, or in other words - whether $C \Rightarrow D$. To this end we have to check whether $\gamma_C(D) = 1$ or not. Because all conditions are distinct, then according to Proposition 7.7 the dependency $C \Rightarrow D$ holds and the table is consistent ~~or not~~, i.e., actions are uniquely determined by conditions in the table.

It is rather straightforward to show that attributes *a*, *c* and *d* are *D*-indispensable, whereas the attribute *b* is *D*-dispensable.

Removing attribute *a* in Table 2 we get the following table

U	*b*	*c*	*d*	*e*	*f*
1	3	2	2	2	4
2	2	2	2	2	4
3	2	2	1	2	4
4	2	2	1	1	4
5	2	2	2	1	4
6	2	2	3	2	3
7	3	2	3	2	3
8	3	2	3	2	3
9	3	3	3	2	2
10	4	3	3	2	2
11	4	3	2	2	2
12	3	3	2	2	2
13	2	3	2	2	2

Table 3

Table 3 is inconsistent ~~since~~ because the following pairs of decision rules

(i) $b_2\ c_2\ d_1 \rightarrow e_{\cancel{1}2}\ f_4$ (rule 3)

$b_2\ c_2\ d_1 \rightarrow e_{\cancel{2}1}\ f_4$ (rule 4)

(ii) $b_2\ c_2\ d_2 \rightarrow e_{\cancel{1}2}\ f_4$ (rule 2)

$b_2\ c_2\ d_2 \rightarrow e_{\cancel{2}1}\ f_4$ (rule 5)

are inconsistent.

Removing attribute *b* from Table 2 we get the next table

U	*a*	*c*	*d*	*e*	*f*
1	3	2	2	2	4
2	3	2	2	2	4
3	3	2	1	2	4
4	2	2	1	1	4
5	2	2	2	1	4
6	3	2	3	2	3
7	3	2	3	2	3
8	4	2	3	2	3
9	4	3	3	2	2
10	4	3	3	2	2
11	4	3	2	2	2
12	4	3	2	2	2
13	4	3	2	2	2

Table 4

It is easly seen that all decison ~~ruthe~~ rules in the table are consistent, hence the attribute *b* is superfluous.

Without attribute *c* Table 2 yields Table 5, as shown below.

U	a	b	d	e	f
1	3	3	2	2	4
2	3	2	2	2	4
3	3	2	1	2	4
4	2	2	1	1	4
5	2	2	2	1	4
6	3	2	3	2	3
7	3	3	3	2	3
8	4	3	3	2	3
9	4	3	3	2	2
10	4	4	3	2	2
11	4	4	2	2	2
12	4	3	2	2	2
13	4	2	2	2	2

Table 5

in which the following pair of decision rules is inconsistent

$a_4\ b_3\ d_3 \longrightarrow e_2\ f_3$ (rule 8)

$a_4\ b_3\ d_3 \longrightarrow e_2\ f_2$ (rule 9)

Finally removing attribute d we get Table 6

U	a	b	c	e	f
1	3	3	2	2	4
2	3	2	2	2	4
3	3	2	2	2	4
4	2	2	2	1	4
5	2	2	2	1	4
6	3	2	2	2	3
7	3	3	2	2	3
8	4	3	2	2	3
9	4	3	3	2	2
10	4	4	3	2	2
11	4	4	3	2	2
12	4	3	3	2	2
13	4	2	3	2	2

Table 6

in which the following pairs of decision rules

(i) $a_3\ b_3\ c_2 \longrightarrow e_2\ f_4$ (rule 1)

$a_3\ b_3\ c_2 \longrightarrow e_2\ f_3$ (rule 7)

(ii) $a_3\ b_2\ c_2 \longrightarrow e_2\ f_4$ (rule 3)

$a_3\ b_2\ c_2 \longrightarrow e_2\ f_3$ (rule 6)

are inconsistent

Thus without the attribute *a*, *c* or *d* Table 2 becomes inconsistent, and without the attribute *b* the table remains consistent. Consequently the set {*a*,*b*,*c*} is *D*-core of *C* and at the same time the only *D*-reduct of *C*. Hence the set of condition attributes *C* is *D*-dependent and the attribute *b* can be dropped from the table.

Thus after removing superfluous attribute *b* and duplicate decision rules, which occurred after removing the attribute *b*, and new numeration of decision rules, Table 2 can be simplified as shown in Table 7 below. We recall that, if there are two or more identical decision rules in a table we should drop all but one, arbitrary representative.

U	*a*	*c*	*d*	*e*	*f*
1	3	2	2	2	4
2	3	2	1	2	4
3	2	2	1	1	4
4	2	2	2	1	4
5	3	2	3	2	3
6	4	2	3	2	3
7	4	3	3	2	2
8	4	3	2	2	2

Table 7

Let us note that in this decision table there are four kinds of possible decisions, which are specified by the following pairs of values of decision attributes e and f : (e_2, f_4), (e_1, f_4), (e_2, f_3) and (e_2, f_2), denoted in what follows by I, II, III and IV, respectively. For easy of notation we will replace values of decision attributes in the table by numbers of corresponding decision classes as shown in Table 8.

U	a	c	d	e	f
1	3	2	2	I	
2	3	2	1		
3	2	2	1	II	
4	2	2	2		
5	3	2	3	III	
6	4	2	3		
7	4	3	3	IV	
8	4	3	2		

Table 8

Let us now turn our attention to the problem of removing superfluous values of condition attributes from the table.

For this purpose we have to compute which attribute values are dispensable or indispensable with respect to each decision class and find out core values and reduct values for each decision rule. That means we are looking only for those attribute values which are necessary to distinguish all decision classes (see section 7.9), i.e., preserving consistency of the table.

As an example let us compute core values and and reduct values for the first decision rule $a_3\, c_2\, d_2 \longrightarrow e_2\, f_4$ in Table 8.

Values a_3 and d_2 are indispensable in the rule, since the following pairs of rules are inconsistent

(i) $c_2\, d_2 \longrightarrow e_2\, f_4$ (rule 1)

$c_2\, d_2 \longrightarrow e_1\, f_4$ (rule 4)

(ii) $a_3\ c_2 \longrightarrow e_2\ f_4$ (rule 1)

$a_3\ c_2 \longrightarrow e_2\ f_3$ (rule 5),

whereas the attribute value c_2 is dispensable, since the decision rule $a_3\ d_2 \longrightarrow e_2\ f_4$ is consistent. Thus a_3 and d_2 are core values of the decision rule $a_3\ c_2\ d_2 \longrightarrow e_2\ f_4$.

For comparison let us also compute the core values of the decision rule using Proposition 7.1.

To this end we have to check whether the following inclusions $|c_2\ d_2| \subseteq |e_2\ f_4|$, $|a_3\ d_2| \subseteq |e_2\ f_4|$ and $|a_3\ c_2| \subseteq |e_2\ f_4|$ are valid or not. Because we have $|c_2\ d_2| = \{1, 4\}$, $|a_3\ d_2| = \{1\}$, $|a_3\ c_2| = \{1, 2, 5\}$ and $|e_2\ f_4| = \{1, 2\}$, hence only the decision rule $a_3\ d_2 \longrightarrow e_2\ f_4$ is true, and consequently the core values of the first decision rule are a_3 and d_2. It is interesting to compare these both methods, however we leave this to the interested reader.

Computing core values for the remaining decision rules we get the results as shown in Table 9.

U	a	c	d	e	f
1	3	-	2	I	
2	3	-	1		
3	2	-	-	II	
4	2	-	-		
5	-	-	3	III	
6	-	2	-		
7	-	3	-	IV	
8	-	-	-		

Table 9

It can be easily seen that in the decision classes I and II sets of core values of each decision rule are also reducts, because rules

$a_3\ d_2 \longrightarrow e_2\ f_4$

$a_3\ d_1 \longrightarrow e_2\ f_4$

$a_2 \longrightarrow e_1\ f_4$

are true. For the decision classes III and IV however core values do not form value reducts. For example decision rules

$d_3 \longrightarrow e_2\ f_3$ (rule 5)

$d_3 \longrightarrow e_2\ f_2$ (rule 7)

are inconsistent and so are ~~also~~ decision rules

$c_2 \longrightarrow e_2\ f_3$ (rule 6)

$c_2 \longrightarrow e_1\ f_4$ (rule 4)

hence, according to the definition, they do not form reducts.

All possible value reducts for in Table 8 are listed in Table 10 below:

U	*a*	*c*	*d*	*e* *f*
1	3	x	2	I
2	3	x	1	
3	2	x	x	II
4	2	x	x	
5	x	2	3	III
5′	3	x	3	
6	4	2	x	
6′	x	2	3	
7	x	3	x	IV
8	4	3	x	
8′	x	3	2	
8′′	4	x	2	

Table 10

In order to find minimal decision algorithm we have to remove superfluous decision rules from each decision class.

(see Chapter 7.10). It is easy to see that there are not superfluous decision rules in classes I and II. For decision class III we have two minimal solutions

$$c_2\, d_3 \longrightarrow e_2\, f_3$$

and

$$a_4\, c_2 \longrightarrow e_2\, f_3$$

$$a_3\, d_3 \longrightarrow e_2\, f_3$$

and for class IV we have one minimal solution

$$c_3 \longrightarrow e_2\, f_2.$$

hence we have the following two minimal decision algorithms

$$a_3\, d_2 \longrightarrow e_2\, f_4$$

$$a_3\, d_{\cancel{2}\,1} \longrightarrow e_2\, f_4$$

$$a_2 \longrightarrow e_1\, f_4$$

$$c_2\, d_3 \longrightarrow e_2\, f_3$$

$$c_3 \longrightarrow e_2\, f_2$$

and

$$a_3\, d_2 \longrightarrow e_2\, f_4$$

$$a_3\, d_{\cancel{2}\,1} \longrightarrow e_2\, f_4$$

$$a_2 \longrightarrow e_1\, f_4$$

$$a_3\, d_3 \longrightarrow e_2\, f_3$$

$$a_4\, c_2 \longrightarrow e_2\, f_3$$

$$c_3 \longrightarrow e_2\, f_2$$

The combined form of these algorithms are

$$a_3\ d_2 \vee a_3\ d_{\cancel{2}\,1} \longrightarrow e_2\ f_4$$

$$a_2 \longrightarrow e_1\ f_4$$

$$c_2\ d_3 \longrightarrow e_2\ f_3$$

$$c_3 \longrightarrow e_2\ f_2$$

and

$$a_3\ d_2 \vee a_3\ d_{\cancel{2}\,1} \longrightarrow e_2\ f_4$$

$$a_2 \longrightarrow e_1\ f_4$$

$$a_3\ d_3 \vee a_4\ c_2 \longrightarrow e_2\ f_3$$

$$c_3 \longrightarrow e_2\ f_2$$

4. Another Approach

It seems to be interesting to discuss briefly another example of cement kiln control (cf. Sandness (1986)) in which actions of a stoker are based not on the kiln state but on the quality of the cement produced, described by the following attributes:

a - Granularity
b - Viscosity
c - Color
d - pH level

which are assumed to be the condition attributes.

Again there are two decision (actions) attributes

e - Rotation Speed
f - Temperature

Below the corresponding decision table is given.

U	*a*	*b*	*c*	*d*	*e*	*f*
1	2	1	1	1	1	3
2	2	1	1	0	1	3
3	2	2	1	1	1	3
4	1	1	1	0	0	3
5	1	1	1	1	0	3
6	2	1	1	2	1	2
7	2	2	1	2	1	2
8	3	2	1	2	1	2
9	3	2	2	2	1	1
10	3	3	2	2	1	1
11	3	3	2	1	1	1
12	3	2	2	1	1	1
13	3	0	2	1	1	1

Table 11

We skip the discussion of attribute values meaning, but one general remark concerning the table seems to be interesting. Note that now the table is not obtained as a result of the stoker's actions observation, and do not represent the stoker knowledge, but it cont~~ains~~ the prescription which the stoker should follow in order to produce cement of required quality. This example is ~~meant~~ used as an illustration that the meaning of a decision table may be arbitrary, and is not connected with formal properties of the table.

Proceeding as in the previous example we find out that the attribute *b* is again dispensable with respect to the decision attributes, which means that the viscosity is superfluous condition, which can be dropped without affecting the decision procedure. Thus Table 11 after renumeration of decision rules can be simplified as shown below

U	*a*	*c*	*d*	*e*	*f*
1	2	1	0	I	
2	2	1	1		
3	1	1	0	II	
4	1	1	1		
5	2	1	2	III	
6	3	1	2		
7	3	2	2	IV	
8	3	2	1		

Table 12

The core values of attributes are computed next as shown in Table 13

U	*a*	*c*	*d*	*e*	*f*
1	2	-	0	I	
2	2	-	1		
3	1	-	-	II	
4	1	-	-		
5	-	-	2	III	
6	-	1	-		
7	-	2	-	IV	
8	-	-	-		

Table 13

Note that in ~~the~~ row 8 the core value is the empty set. For class I and II the set of core values are at the same time the reducts, however this is not the case for classes III and IV.

Computing now reduct values for each decision class we obtain the following results.

U	a	c	d	e f
1	2	x	0	I
2	2	x	1	
3	1	x	x	II
4	1	x	x	
5	2	x	2	III
5′	x	1	2	
6	3	1	x	
6′	x	1	2	
7	x	2	x	IV
8	3	2	x	
8′	3	x	1	
8′′	x	2	1	

Table 14

Removing now redundant decision rules from the table we obtain the following two minimal decision tables

U	a	c	d	e f
1	2	x	0	I
2	2	x	1	
3	1	x	x	II
4	x	1	2	III
5	x	2	x	IV

Table 15

and

U	a	c	d	e f
1	2	x	0	I
2	2	x	1	
3	1	x	x	II
4	2	x	2	III
5	3	1	x	
6	x	2	x	IV

Table 16

From this tables we have the following minimal decision algorithms in normal forms

$$a_2\ d_0 \longrightarrow e_1\ f_3$$

$$a_2\ d_1 \longrightarrow e_1\ f_3$$

$$a_1 \longrightarrow e_0\ f_3$$

$$c_1\ d_2 \longrightarrow e_1\ f_2$$

$$c_2 \longrightarrow e_1\ f_1$$

and

$$a_2\ d_0 \longrightarrow e_1\ f_3$$

$$a_2\ d_1 \longrightarrow e_1\ f_3$$

$$a_1 \longrightarrow e_0\ f_3$$

$$a_2\ d_2 \longrightarrow e_0\ f_3$$

$$a_3\ c_1 \longrightarrow e_0\ f_3$$

$$c_2 \longrightarrow e_1\ f_1$$

or in the more compact form

$$a_2\ (d_0 \vee d_1) \longrightarrow e_1\ f_3$$

$$a_1 \longrightarrow e_0\ f_3$$

$$c_1\ d_2 \longrightarrow e_1\ f_2$$

$$c_2 \longrightarrow e_1\ f_1$$

and

$$a_2\ (d_0 \vee d_1) \longrightarrow e_1\ f_3$$

$$a_1 \longrightarrow e_0\ f_3$$

$$a_2\ d_2 \vee a_3\ c_1 \longrightarrow e_{\cancel{1}0}\ f_{\cancel{2}3}$$

$$c_2 \longrightarrow e_1\ f_1$$

It is interesting to compare solutions obtained in sections 3 and 4. Although they are based on completely different approaches the results are almost the same.

5. The Case of Inconsistent Data

In general, when a decision table is the result of observations or measurements it may happen that ~~the~~ table is inconsistent, since some observed or measured data can be conflicting. This leads to partial dependency of decision and condition attributes. Basically we are interest ed in conclusions which can be derived from nonconflicting (consistent) data, but sometimes ~~also~~ the inconsistent part of data could also be of interest.

To illustrate this idea let us consider an example of decision table, taken from Grzymała-Busse (1989a). Table 17 contains some observed symptoms of patients, denoted by numbers 1,2, ...,9, suffering from certain disease. The question arises whether there are some characteristic symptoms for influenza. In order to answer this question we have to compute whether there is a dependency between decision and condition attributes in the table.

U	*a*	*b*	*c*	*d*	*e*
1	normal	absent	absent	absent	absent
2	normal	absent	present	present	absent
3	subfeb.	absent	present	present	present
4	subfeb.	present	absent	absent	absent
5	subfeb.	present	absent	absent	present
6	high	absent	absent	absent	absent
7	high	present	absent	absent	absent
8	high	present	absent	absent	present
9	high	present	present	present	present

Table 17

where

a - *Temperature*
b - *Dry-cough*
c - *Headache*
d - *Muscle pain*

are condition attributes, and

e - *Influenza*

is decision attribute.

It is easily seen that decision rules 4 and 5 are inconsistent.

Rule 4:

```
If
    (Temperature, subfeb.) and
    (Dry-cough, absent) and
    (Headache, absent) and
    (Muscle pain, absent)
then
    (Influenza, present)
```

Rule 5:

```
If
    (Temperature, subfeb.) and
    (Dry-cough, absent) and
    (Headache, absent) and
    (Muscle pain, absent)
then
    (Influenza, absent)
```

Similarly decision rules 7 and 8 are inconsistent.

Rule 7:

If
    (*Temperature*, high) and
    (*Dry-cough*, present) and
    (*Headache*, absent) and
    (*Muscle pain*, absent)
then
    (*Influenza*, absent)

Rule 8:

If
    (*Temperature*, high) and
    (*Dry-cough*, present) and
    (*Headache*, absent) and
    (*Muscle pain*, absent)
then
    (*Influenza*, present)

The remaining five decision rules are true, so the dependency between decision and condition attribute is 5/9. This means that the condition attributes are not sufficient to decide whether a patient has influenza or not. Of course patients displaying symptoms occurring in the consistent (true) decision rules can be classified as having influenza. Thus we can decompose the decision table (algorithm) into two decision tables (algorithms), consistent and totally inconsistent. The inconsistent part consists of rules 4, 5, 7 and 8, and the rest of decision rules constitute the consistent part of the algorithm, as shown below:

Rule 1:
If
    (*Temperature*, normal) and
    (*Dry-cough*, absent) and
    (*Headache*, absent) and
    (*Muscle pain*, absent)

```
    then
        (Influenza, absent)


Rule 2:
    If
        (Temperature, normal) and
        (Dry-cough, absent) and
        (Headache, present) and
        (Muscle pain, present)
    then
        (Influenza, absent)

Rule 3:
    If
        (Temperature, subfeb.) and
        (Dry-cough, absent) and
        (Headache, present) and
        (Muscle pain, present)
    then
        (Influenza, present)

Rule 6:
    If
        (Temperature, high) and
        (Dry-cough, absent) and
        (Headache, absent) and
        (Muscle pain, absent)
    then
        (Influenza, absent)

Rule 9:
    If
        (Temperature, high) and
        (Dry-cough, present) and
        (Headache, present) and
        (Muscle pain, present)
    then
        (Influenza, present)
```

Now we can reduce superfluous condition attributes. To this end first we have to compute core of condition attributes. Because the extended notation for decision rules is rather obscure when computing core and reducts, for the sake of simplicity we shall use tabular notation for decision algorithm and values of attributes will be shortened by their first letters. Thus we shall present the table in abbreviated form

U	*a*	*b*	*c*	*d*	*e*
1	n	a	a	a	a
2	n	a	p	p	a
3	s	a	p	p	p
4	s	p	a	a	a
5	s	p	a	a	p
6	h	a	a	a	a
7	h	p	a	a	a
8	h	p	a	a	p
9	h	p	p	p	p

Table 18

The inconsistent decision rules are underlined in the table.

Note that attributes *c* and *d* are equivalent, so we can drop one of them, say *d*, and as a result we get Table 19.

U	*a*	*b*	*c*	*e*
1	n	a	a	a
2	n	a	p	a
3	s	a	p	p
4	s	p	a	a
5	s	p	a	p
6	h	a	a	a
7	h	p	a	a
8	h	p	a	p
9	h	p	p	p

Table 19

Now we are about to compute the core of condition attributes. Removing the attribute *a* we obtain the table

U	*b*	*c*	*e*
1	a	a	a
2	a	p	a
3	a	p	p
4	p	a	a
5	p	a	p
6	a	a	a
7	p	a	a
8	p	a	p
9	p	p	p

Table 20

in which rules 2 and 3 are inconsistent, and the positive region (the set of consistent decision rules) of the algorithm will be changed, thus the attribute *a* is indispensable and belongs to the core.

Removing attribute *b* we obtain Table 21

U	*a*	*c*	*e*
1	n	a	a
2	n	p	a
3	s	p	p
4	s	a	a
5	s	a	p
6	h	a	a
7	h	a	a
8	h	a	p
9	h	p	p

Table 21

in which rules 6 and 8 are inconsistent, hence rule 6 is false and the positive region of the algorithm changes, i.e. the attribute *b* is indispensable, and belongs to the core.

Removing attributes *c* and we will arrive at the Table22

U	*a*	*b*	*e*
1	n	a	a
2	n	a	a
3	s	a	p
4	s.	p	a
5	s	p	p
6	h	a	a
7	h	p	a
8	h	p	p
9	h	p	p

Table 22

In this table rules 7 and 9 are inconsistent, so after removing attribute *c* rule 9 will be inconsistent (false), and the positive region of the algorithm changes, hence the attribute is indispensable and belongs to the core. In other words the set of condition attributes *a*, *b* and *c* is independent and it forms a reduct of condition attributes. In order to simplify further the decision table and find minimal solutions we have to eliminate superfluous values of attributes in all consistent decision rules in the decision table. First we have to compute core values of condition attributes in all consistent decision rules, and the results are shown in Table 23.

U	*a*	*b*	*c*	*e*
1	-	-	-	a
2	n	-	-	a
3	s	-	-	p
4	s	p	a	a
5	s	p	a	p
6	-	a	-	a
7	h	p	a	a
8	h	p	a	p
9	-	-	p	p

Table 23

We recall that core values of condition attributes are those values of condition attributes, which do not preserve consistency (truth) of consistent decision rules, when removed from the rule. Now we are able to compute all reduct of consistent decision rules, and the result is shown below.

U	*a*	*b*	*c*	*d*
1	n	a	x	a
1′	n	x	a	a
1′′	x	a	a	a
2	s	x	x	a
4	s	p	a	a
5	s	p	a	p
6	h	a	x	a
6′	x	a	a	a
7	h	p	a	a
8	h	p	a	p
9	h	x	p	p
9′	x	p	p	p

3 s a x p
3' s x p p

Table 24

We see from this table that rule 1 has three reducts, rule 2 one reduct and remaining inconsistent rules have two reducts each. It is easily seen all reducts of rule 1 are superfluous rules and consequently we have 2 x 2 x 2 = 8 minimal equivalent decision algorithms. One exemplary minimal deterministic decision algorithm is shown in what follows:

$$a_n \longrightarrow e_a$$

$$a_h\, b_a \longrightarrow e_a$$

$$a_s\, b_a \longrightarrow e_p$$

$$a_h\, c_p \longrightarrow e_p$$

or in an extended form

Rule 2:

```
    If
        (Temperature, normal) and
    then
        (Influenza, absent)
```

Rule 6:

```
    If
        (Temperature, high) and
        (Dry-cough, absent) and
    then
        (Influenza, absent)
```

Rule 3:

```
    If
        (Temperature, subfeb.) and
        (Dry-cough, absent) and
    then
        (Influenza, present)
```

```
Rule 9:
     If
          (Temperature, high) and
          (Muscle pain, present)
     then
          (Influenza, present)
```

The algorithm can be also presented in the following form

```
Rule 1:
     If
          (Temperature, normal) or
          ((Temperature, high) ~~and~~
          (Dry-cough, absent))
     then
          (Influenza, absent)

Rule 2:
     If
          ((Temperature, subfeb.) and
          (Dry-cough, absent)) or
          ((Temperature, high) and
          (Muscle pain, present))
     then
          (Influenza, present)
```

Summary

The basic idea of ~~this~~ these examples consist~~s~~ in that we retrieve knowledge of an agent by observ~~ation~~ing ~~of~~ ~~his~~ its behavior, and this recorded behavior is used ~~as a starting point~~ to ~~the~~ generat~~ion~~e ~~of~~ an algorithm~~, which is able~~ to simulate ~~his~~ its actions~~.~~ in the future.

Let us also note that in general, many minimal solutions are possible, the simplification of decision tables does not yield unique results, and the decision algorithm can be optimized according to pre assumed criteria. For example, in our case the minimal number of decision rules can be used as a criterion for optimization.

Exercise

1. Compute all core values and value reducts for all decision rules in Table 7 and Table 12.

2. Check whether attributes *e* and *f* are independent or not in both Table 7 and Table 12.

3. Compute minimal decision algorithms (i.e.dependencies, reducts, cores and also value reducts and core reducts) for the following decision tables:

a) Students admission

U	*a*	*b*	*c*	*d*	*e*
1	3.8-3.99	Average	No	Poor	Reject
2	4.0	Average	Yes	Poor	Accept
3	3.8-3.99	Good	No	Extensive	Accept
4	Below 3.8	Good	No	Extensive	Reject
5	4.0	Poor	Yes	Average	Accept
6	Below 3.8	Average	No	Average	Reject
7	4.0	Average	Yes	Poor	Reject
8	4.0	Poor	Yes	Average	Reject

Table 25

The meaning of the attributes is as follows:

a - *High-School GPA*
b - *Extracurricular-activities*
c - *Alumni-relatives*
d - *Honor-awards*
e - *Decision*

Attributes *a*, *b*, *c*, *d* and *e* are condition attributes, whereas *e* is decision attribute. The example is taken from Grzymała-Busse (1989b) and the reader is advised to read the cited paper. (Coping of this example is by permission of the Association of Computing Machinery).

b) Infants attitude to toys

U	*a*	*b*	*c*	*d*	*e*	*f*
1	Blue	Big	Hard	Indefinite	Plastic	Negative
2	Red	Medium	Moderate	Smooth	Wood	Neutral
3	Yellow	Small	Soft	Furry	Plush	Positive
4	Blue	Medium	Moderate	Fuzzy	Plastic	Negative
5	Yellow	Small	Soft	Indefinite	Plastic	Neutral
6	Green	Big	Hard	Smooth	Wood	Positive
7	Yellow	Small	Hard	Indefinite	Metal	Positive
8	Yellow	Small	Soft	Indefinite	Plastic	Positive
9	Green	Big	Hard	Smooth	Wood	Neutral
10	Green	Medium	Moderate	Smooth	Plastic	Neutral

Table 26

The table taken from Grzymała-Busse (1989c) describes observed infant attitude to toys, characterized by the following (condition) attributes:

a - *Color*
b - *Size*
c - *Feel*
d - *Cuddliness*
e - *Material*

and decision attribute

f - *Attitude*

c) Car performance (Grzymała-Busse (1989c)

U	*a*	*b*	*c*	*d*
1	poor	low	short	< 30
2	poor	low	short	< 30
3	good	low	medium	< 30
4	good	medium	short	30..50
5	poor	low	short	< 30
6	poor	high	long	> 50

Table 27

where

a - Terrain familiarity
b - Gasoline level
c - Distance

are condition attributes and

e - Speed m.p.h.]

is a decision attribute.

g) Table 28 contains the characterisation of six stores in terms of six factors *E, Q, S, R, L* and *P* (cf. Kick et al.).

Store	*E*	*Q*	*S*	*R*	*L*	*P*
1	high	good	yes	yes	no	500
2	high	good	no	yes	no	-100
3	medium	good	yes	yes	yes	200
4	low	avg	yes	yes	yes	70
5	low	good	yes	yes	yes	100
6	high	avg	no	no	yes	-20

Table 28

The meaning of the attributes is the following:

E - empowerment of sales personel
Q - perceived quality of merchandise
S - segmented customer-base (specific market domain)
R - good refund policy
L - high trafic location
p - store profit or loss (in Millions of US dollars)

Detremine what makes a given store successful as measured by the net profit generated by the store (attribute *P*).

4. Compute all minimal decision algorithms for the decision table considered in section 5 (Table 19).

5. Compute uncertainty factors for all inconsistent decision rules in Table 19.

References

Grzymała-Busse, J. (1989a). Managing Uncertainty in Expert Systems. *Department of Computer Science, The University of Kansas*, pp. 1-197.

Grzymała-Busse, J.(1989b). An Overview of the LERS1 Learning System. *Proceedings of the Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE - 89,* at The University of Tennessee Space Institute (UTSI)Tullahoma, Tennessee, June 6-9, 1989, 2, pp. 838-844.

Grzymała-Busse, J.(1989c). Learning Minimal Discriminant Rules from Examples with Inconsistencies - A Rough Set Approach. *Department of Computer Science, The University of Kansas,* November 1989, Report TR-89-4, pp. 1-17.

Kick, R. C. and Koczkodaj, W. W. (1990). Business Data Analysis by Rough Set Theory. *Technological University of Tennessee.* Manuscript.

Mrózek, A. (1985). Information Systems and Control Algorithm. *Bull. Polish Acad. Sci. Tech.,.* 33, pp. 195-?

Mrózek, A. (1986). Use of Rough Sets and Decision Tables for Implementing Rule-based Control of Industrial Processes. *Bull. Polish Acad. Sci. Tech.*, 34, pp. 357-?

Mrózek, A. (1987). Rough Sets and Some Aspects of Expert Systems Realization. *7-th International Workshop on Expert Systems and their Applications,* Avignon, pp. 597-?

Mrózek, A. (1989). Rough Sets and Dependency Analysis Among Attributes in Computer Implementation of Expert's Inference Models. *Int. J. Man-Machine Studies*, 30, pp. 457-473.

Pawlak, Z., Słowiński, K. and Słowiński, R. (1986). Rough Classification of Patients after HSV for Duodenal Ulcer.*International Journal of Man-Machine Studies*, **24**, pp. 413-433.

Sandness, G.D. (1986). A Parallel Learning System. *CS 890 Class Paper*, Carnegie-Mellon University, pp. 1-12.

Słowiński, K., Słowiński, R. and Stefanowski J. (1988). Rough Set Approach to Analysis of Data from Peritoneal Lavage in Acute Pancreatitis. *Med. Inform.*, **13**, pp. 143-159.

10. DISSIMILARITY ANALYSIS

1. Introduction

In Chapters 8, 9 and 10 we confined our discussion of examples of Knowledge Representation Systems in which condition and decision attributes were distinguished, i.e. to decision tables.

In this chapter we are going to discuss Knowledge Representation Systems in which neither condition nor decision attributes are distinguished. Therefore we are basically not interested in dependencies among attributes, but in description of some objects in terms of available attributes, in order to find essential differences between objects of interest. The problem of differentiation of various options is often of crucial importance in decision making.

The formal tools developed so far in this book can be also well used to this purpose, and many problems can be reduced to this schema. Some of them, given in the next sections, will be used to depict the idea.

2. The Middel East Situation

In this section we shall investigate a political situation in the Middle East. The example is taken from Casti (1988), but our approach is different ~~to~~ from that of Casti and is based on the rough set philosophy. It seems that the rough set approach enables us to investigate greater ~~verity~~ variety of problems and yields more straightforward result in comparison to method used by Casti.

Let us consider six parties

1 - Israel
2 - Egypt
3 - Palestinians
4 - Jordan
5 - Syria
6 - Saudi Arabia

The relations between those parties are determined by the following issues

a - autonomous Palestinian state on the West Bank and Gaza

b - return of the West Band and Gaza to Arab rule

c - Israeli military outpost along the Jordan River

d - Israeli retains East Jerusalem

e - free access to all religious centers

f - return of Sinai to Egypt

g - dismantle Israeli Sinai settlements

h - return of Golan Height to Syria

i - Israeli military outposts on the Golan Heights

j - Arab countries grant citizenship to Palestinians who choose to remain within their borders

The table below represents the way the participants of the Middle East region interact with the above issues; 0 means, that the participant is against and 1 - neutral or favorable toward the issue.

U	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
1	0	1	1	1	1	1	0	0	1	1
2	1	1	1	0	1	1	1	1	1	0
3	1	1	0	0	1	1	1	1	1	1
4	1	1	0	0	1	1	1	1	1	0
5	1	1	0	0	1	1	1	1	0	0
6	1	1	1	0	1	1	1	1	1	1

Table 1

Participants in the approach can be treated as objects and issues as attributes, however there are no~~t~~ condition and decision attributes distinguished in the table. Formally it is convenient to consider all the attributes both as condition and decision attributes (why?), and carry out all computation under this assumption.

Our task is to find essential differences between participant to the debate, so that we can clearly distinguish their attitudes to the issues being discussed.

Before entering more specific considerations let us remark that attributes *a*, *g* and *h* are equivalent, hence all of them are not necessary in the table and we can retain only one, say *a*. Attributes *b*,*e* and *f* do not discern participants to the debate, since they have the same values for all negotiating partners, and therefore they are irrelevant in this specific situation, so they can be dropped from the table too. Thus, for the sake of simplicity instead of Table 1 as a departure point of our analysis we can assume the following table (of course we could start our considerations assuming Table 1):

U	*a*	*c*	*d*	*i*	*j*
1	0	1	1	1	1
2	1	1	0	1	0
3	1	0	0	1	1
4	1	0	0	1	0
5	1	0	0	0	0
6	1	1	0	1	1

Table 2

Because all rows in the table are distinct, that means ~~that~~ the table is consistent, which means that all parties engaged in the dispute have different views and all participants can be recognized (discern) by their attitude to the issues discussed.

Let us now compute core issues (attributes).

As we already mentioned we may assume that all attributes are at the same time condition and decision attributes, hence with each row in the table we will associate decision rules (implications) with the same predecessor and successor. For example, with row 1 in Table 2 we associate decision rule a_0 c_1 d_1 i_1 j_1 -> a_0 c_1 d_1 i_1 j_1. As we already have mentioned all decision rules in the table are consistent, i.e. all decision rules have different predecessors, and consequently the decision table is consistent.

It is a simple task to check that *a* and *d* are dispensable, because dropping attribute *a* or *d* we get still consistent tables as shown below.

U	*c*	*d*	*i*	*j*
1	1	1	1	1
2	1	0	1	0
3	0	0	1	1
4	0	0	1	0
5	0	0	0	0
6	1	0	1	1

Table 3

U	*a*	*c*	*i*	*j*
1	0	1	1	1
2	1	1	1	0
3	1	0	1	1
4	1	0	1	0
5	1	0	0	0
6	1	1	1	1

Table 4

On the contrary attributes *c*, *i* and *j* are indispensable, since without these attributes some decision rules in the table are inconsistent (false), e.g. rules

$a_1\ d_0\ i_1\ j_0 \longrightarrow a_1\ c_1\ d_0\ i_1\ j_0$ (rule 2)

$a_1\ d_0\ i_1\ j_0 \longrightarrow a_1\ c_0\ d_0\ i_1\ j_0$ (rule 4)

are inconsistent, also rules

$a_1\ c_0\ d_0\ j_0 \longrightarrow a_1\ c_0\ d_0\ i_1\ j_0$ (rule 4)

$a_1\ c_0\ d_0\ j_0 \longrightarrow a_1\ c_0\ d_0\ i_0\ j_0$ (rule 5)

are also inconsistent, and so are rules

$a_1\ c_1\ d_0\ i_1 \longrightarrow a_1\ c_1\ d_0\ i_1\ j_0$ (rule 2)

$a_1\ c_1\ d_0\ i_1 \longrightarrow a_1\ c_1\ d_0\ i_1\ j_1$ (rule 6).

In other words if we drop *c*, *i* or *j* in Table 2 we obtain tables with duplicate rows, as shown below.

U	*a*	*d*	*i*	*j*
1	0	1	1	1
2	1	0	1	0
3	1	0	1	1
4	1	0	1	0
5	1	0	0	0
6	1	0	1	1

Table 5

U	*a*	*c*	*d*	*j*
1	0	1	1	1
2	1	1	0	0
3	1	0	0	1
4	1	0	0	0
5	1	0	0	0
6	1	1	0	1

Table 6

U	a	c	d	i
1	0	1	1	1
2	1	1	0	1
3	1	0	0	1
4	1	0	0	1
5	1	0	0	0
6	1	1	0	1

Table 7

Identical rows are underlined in these tables. Thus Tables 5, 6 and 7 are inconsistent and consequently the set {*c*, *i*, *j*} is the core and there are two reducts in the table: {*a*, *c*, *i*, *j*} and {*d*, *c*, *i*, *j*}. Hence either of the reducts can be assumed as a basis for negotiations, for they provide two equivalent sets of issues for discussion. Hence instead of Table 2 we can consider Table ~~2~~ 3 or Table ~~3~~ 4 given below:

Computing now core values of attributes for each table we obtain the following tables.

U	*a*	*c*	*i*	*j*
1	0	-	-	-
2	-	1	-	0
3	-	0	-	1
4	-	0	1	0
5	-	-	0	-
6	1	1	-	1

Table 8

U	*c*	*d*	*i*	*j*
1	-	1	-	-
2	1	-	-	0
3	0	-	-	1
4	0	-	1	0
5	-	-	0	-
6	1	0	-	1

Table 9

What we have achieved so far is the following. Not all issues are of equal importance in the debate. Some issues can be eliminated form the negotiations without affecting the position of the parties involved in the negotiation process, some can not be eliminated without changing balance of opinion, (core issues *c*, *i* and *j*) and some issues can be mutually replaced, without changing essentially views of the negotiating parties (issues *a* and *d*).

It is also easy to see that the core values given in Table 8 and Table 9 are also the reduct values. Examining the tables we get minimal characteristic of each participant of the discussion with respect to the discussed issues.

For example for Table 8 the minimal characteristics of each participants are:

Israel
Against:
-*Autonomous Palestinian state*

Egipt
Favorable:
-*Israeli military outposts along the Jordan River*
Against:
-*Arab countries grant citizenship to Palestinian*

Palestinians
Against:
-*Israeli military outposts along the Jordan River*
Favorable:
-*Toward Arab countries grant citizenship to Palestinians*

Jordan
Against:
-*Israeli military outposts along the Jordan River*
-*Arab countries grant citizenship to Palestinians*
Favorable:
-*Israeli military outposts on the Golan Highs*

Syria

Against:

-Israeli military outposts on the Golan Highs

Saudi Arabia

Favorable:

-Autonomous Palestinians state

-Israeli military outposts along the Jordan River

-Arab countries grant citizenship to Palestinians

The above conditions can be presented in symbolic form

Israel	- a_0
Egipt	- $c_1\ j_0$
Palestinians	- $c_0\ j_1$
Jordan	- $c_0\ i_1\ j_0$
Syria	- i_0
Saudi Arabia	- $a_1\ c_1\ j_1$

or in the form

$a_0 \longrightarrow$ **Israel**

Y

$c_1\ j_0 \longrightarrow$ **Eg~~i~~pt**

$c_0\ j_1 \longrightarrow$ **Palestinians**

$c_0\ i_1\ j_0 \longrightarrow$ **Jordan**

$i_0 \longrightarrow$ **Syria**

$a_1\ c_1\ j_1 \longrightarrow$ **Saudi Arabia**

Remark

Note that the above description, formally is not ~~a~~ decision algorithm, since the expressions like **Israel**, **Egipt** etc. do not belong to our formal language. Nevertheless, for simplicity we will treat this as a decision algorithm. ■

Similar characteristic can be obtained for Table 9.

The essence of the example illustrates several vital points about the rough set analysis of this kind of situations. What we have done consists $\overset{of}{in}$, as we already mentioned, first of elimination of superfluous issues, which can be skipped from the negotiations, without disturbing participants' positions (elimination of dispensable attributes), and second, elimination of unessential opinions within the remaining issues (elimination of superfluous values of attributes).

Using the proposed approach we can answer many more questions. For example we might be interest^{ed} in knowing how far apart are the views of participants. To this end we can define *a distance graph* as follows: with each object in the table we associate a node labelled by this object and two nodes *x* and *y* are connected by a vertex labelled *α* if, removing the attribute *α* places objects *x* and *y* in the same equivalence class (i.e. they have the same description in terms of remaining attributes values).

The distance graph for Table 8 has the form

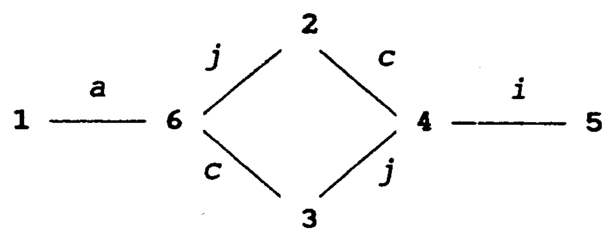


Fig.1

and for Table 9 we have the following graph

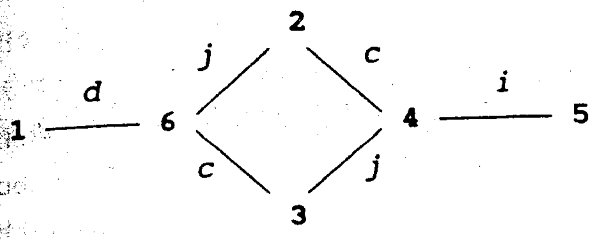


Fig.2

Distance between two objects *x* and *y* is the length of the shortest path between *x* and *y* in the distance graph, i.e. the smallest number of vertices which connect them. For example distance between Israel (1) and Syria (5) is 3, between Israel (1) and Saudi Arabia (6)- is 1, between Saudi Arabia (6) and Jordan (4) is 2 etc. Thus distance between participants in this example has quite clear intuitive explanation: it is the number of issues that separates them in their views. Removing of specific issue would ~~deeries~~ decrease the distance between some participants by 1 bringing their views closer. Thus the distance graph gives interesting insight in to the structure of the analyzed situation and enable us to better understand the nature of the conflict analyzed.

3. Beauty Contest

In this section we shall analyze again Table 2 with the difference that now we will address rather different problems.

Suppose objects in Table 2 are girls Jane, Marry, Ann, Barbara, Jackie and Dorothy taking part in a beauty contest, and there are five referees *a*, *c*, *d*, *i* and *j*. The table entries are judgment of referees, meaning 1 for *beautiful* and 0 for *average*. (Of course arbitrary number of grades could be assumed by referees, but for simplicity we will stick to two values as in the table already analyzed). We assume now that the values of attributes are now ordered by the strict ordering relation, $(0 < 1)$ what in the previous example was not necessary, and any two not ordered values of attributes could be assumed. Thus if two girls *x* and *y* scored 1 and 0 respectively by the referee *a*, that means that according to the referee *a* the girl *x* is more beautiful than the girl *y*.

There are many theoretical approaches to analyze this kind of voting schemes, (cf.Nurmi 1987) however we will not discuss them here, but we would like to ~~rise~~ raise only one point, which seems to be of basic importance when speaking about voting - namely the independence of judgments. We claim that the independence of judgments is the basic assumption which should be fulfilled before applying any specific voting scheme.

In our setting the problem of independence of judgments reduces to the investigation of independence of attributes.

To check whether the judgment in the beauty contest are independent or not we have to investigate whether the set of attributes $\{a, c, d, i, j\}$ is independent or not. As we showed in the previous section this set is in dependent and there are two reducts in this set

$$\{a, c, i, j\} \text{ and } \{c, d, i, j\}$$

i.e. two subsets of independent referees and consequently, according to the property that if $B \subseteq A$ is a reduct of A, then $B \Rightarrow A - B$, (cf. Proposition 4.2) - we have exactly two dependencies

$$\{a, c, i, j\} \Rightarrow d \text{ and } \{d, c, i, j\} \Rightarrow a.$$

Employing now the concept of the distance graph we can see differences between various judgments, however we are unable to say which girl is the most beautiful one. So far we can say only by inspecting the distance graph (Fig. 1, Fig. 2) that, according to referees opinions, Marry and Ann are equally beautiful. What more the distance graph yields information on how the differences are related to referees opinions~~.~~ For example the difference between Jackie and ~~Dorothy~~ Barbara is due to the judgment of referee i. Without his opinion they would be indistinguishable.

In this way we include some kind of semantic information about the differences between girls, which can give deeper insight in the mechanisms of judgments.

Let us also mention that the essence of opinions delivered by the group of referees $\{a, c, i, j\}$ can be expressed by the formulas given below

Jane	$- a_0$
Mary	$- c_1\ j_0$
Ann	$- c_0\ j_1$
Barbara	$- c_0\ i_1\ j_0$
Jackie	$- i_0$
Dorothy	$- a_1\ c_1\ j_1$

which can be also presented as decision algorithm

$a_0 \longrightarrow$ Jane

$c_1\ j_0 \longrightarrow$ Mary

$c_0\ j_1 \longrightarrow$ Ann

$c_0\ i_1\ j_0 \longrightarrow$ Barbara

$i_0 \longrightarrow$ Jackie

$a_1\ c_1\ j_1 \longrightarrow$ Dorothy

Of course similar set of decision rules can be given also for the set $\{c, d, i, j\}$.

Observe that these formulas (or decision rules) uniquely characterize each girl in terms of referees judgments and can be used to analyze differences between the girls and applied in final evaluation. Note also that this kind of characteristic can be given only for independent set of referees.

The presented method is obviously of a general nature and can be applied to various problems, like for example marketing analysis, performance evaluation, etc., and lies in fact in the scope of group decision making area.

4. Pattern Recognition

Consider example 5.4 of digits display unit in a calculator, but now the table will be assumed to represent a characterization of "hand written " digits, consisting of horizontal and vertical strokes. Certainly real life character recognition algorithm must admit arbitrary written characters, not only combinations of horizontal and verticals strokes, as we propose here, however for the sake of illustration the assumed representation seems to be sufficient and simplifies the discussion. Table 10 describes each digit in terms of elements *a*, *b*, *c*, *d*, *e*, *f* and *g* as shown in Example 5.4.

U	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Table 10

Or task is to find minimal description of each digit and the corresponding decision algorithm.

As in the previous examples ~~first~~ we have to compute the core first and next find all reducts of attributes. Note first that the table is consistent, i.e. each digit is uniquely characterized in terms of the attributes. It is easy to check that the core is the set {*a*, *b*, *e*, *f*, *g* }, which is also the only ~~one~~ reduct. Hence instead of the whole set of attributes

$\{a, b, c, d, e, f, g\}$ it is sufficient to consider only the attributes $\{a, b, e, f, g\}$ as a basis for the decision algorithm, which means that there is the following dependency between the attributes

$$\{a, b, e, f, g\} \Rightarrow \{c, d\}$$

i.e. attributes *c* and *d* are dependent upon the reduct and they are not necessary to digits recognition. As a result Table 10 can be replaced by Table 11.

U	*a*	*b*	*e*	*f*	*g*
0	1	1	1	1	0
1	0	1	0	0	0
2	1	1	1	0	1
3	1	1	0	0	1
4	0	1	0	1	1
5	1	0	0	1	1
6	1	0	1	1	1
7	1	1	0	0	0
8	1	1	1	1	1
9	1	1	0	1	1

Table 11

Now we have to reduce each decision rule in the table separately, by computing first the core of each rule. Let us once more compute, for the sake of illustration, all core values for each decision rule. We remind the reader that dropping the core value makes the consistent rule inconsistent. Thus removing the attribute *a* makes rules 1, 7 and 4, 9 inconsistent, i.e. we unable to discern digits 1 and 7, and 4 and 9 without the attribute *a*, as shown below (inconsistent rules are underlined). Hence a_0 is the core value in rules 1 and 4, whereas a_1 is the core value in rules 7 and 9.

U	*b*	*e*	*f*	*g*
0	1	1	1	0
1	1	0	0	0
2	1	1	0	1
3	1	0	0	1
4	1	0	1	1
5	0	0	1	1
6	0	1	1	1
7	1	0	0	0
8	1	1	1	1
9	1	0	1	1

Table 12

Removing attribute *b* we get Table 13

U	*a*	*e*	*f*	*g*
0	1	1	1	0
1	0	0	0	0
2	1	1	0	1
3	1	0	0	1
4	0	0	1	1
5	1	0	1	1
6	1	1	1	1
7	1	0	0	0
8	1	1	1	1
9	1	0	1	1

Table 13

in which rules (6, 8) and (5, 9) are indiscernible, thus values b_0 and b_1 are core values in these rules respectively.

Similarly removing the attribute *e* we get the table

U	*a*	*b*	*f*	*g*
0	1	1	1	0
1	0	1	0	0
2	1	1	0	1
3	1	1	0	1
4	0	1	1	1
5	1	0	1	1
6	1	0	1	1
7	1	1	0	0
8	1	1	1	1
9	1	1	1	1

Table 14

we get three pairs of indiscernible rule, (2, 3), (5, 6) and (8,9), which yields core values for corresponding rules (e_1, e_0), (e_0, e_1) and (e_1, e_0).

Without the attribute *f* we get the table

U	*a*	*b*	*e*	*g*
0	1	1	1	0
1	0	1	0	0
2	1	1	1	1
3	1	1	0	1
4	0	1	0	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	0
8	1	1	1	1
9	1	1	0	1

Table 15

yields indiscernible pairs (2, 8) and (3, 9) and core values for each pair are (f_0, f_1).

Finally the ~~least~~ last attribute g removed gives indiscernibility shown in Table 16.

U	a	b	e	f
0	1	1	1	1
1	0	1	0	0
2	1	1	1	0
3	1	1	0	0
4	0	1	0	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	0
8	1	1	1	1
9	1	1	0	1

Table 16

where pairs of rules (0, 8) and (3, 7) are indiscernible, and the values (g_0, g_1) and (g_1, g_0) are cores of corresponding decision rules.

Core values for all decision rules are listed in Table 17.

U	a	b	e	f	g
0	-	-	-	-	0
1	0	-	-	-	-
2	-	-	1	0	-
3	-	-	0	0	1
4	0	-	-	-	-
5	-	0	0	-	-
6	-	0	1	-	-
7	1	-	-	-	0
8	-	1	1	1	1
9	1	1	0	1	-

Table 17

Rules 2, 3, 5, 6, 8 and 9 are already reduced, since ~~this~~ these core values discern these rules from the remaining ~~once~~ ones, i.e. the rules with the core values only are consistent (true). For the remaining rules core values make them inconsistent (false), so they do not form reducts and reducts must be computed by adding proper additional attributes to make the rules consistent. The results are shown in Table 18.

U	a	b	e	f	g
0	x	x	1	x	0
0′	x	x	x	1	0
1	0	x	x	0	x
1′	0	x	x	x	0
2	x	x	1	0	x
3	x	x	0	0	1
4	0	x	x	1	x
4′	0	x	x	x	1
5	x	0	0	x	x
6	x	0	1	x	x
7	1	x	0	x	0
7′	1	x	x	0	0
8	x	1	1	1	1
9	1	1	0	1	x

Table 18

Because the four decision rules 0, 1, 4 and 7 have two reduced forms hence we have all together 16 minimal decision algorithms. One of possible reduced algorithm is shown below.

$e_1\, g_0\ (f_1\, g_0) \longrightarrow 0$

$a_0\, f_0\ (a_0\, g_0) \longrightarrow 1$

$e_1\, f_0 \longrightarrow 2$

$e_0\, f_0\, g_1 \longrightarrow 3$

$a_0\, f_1\ (a_0\, g_1) \longrightarrow 4$

$b_0 \; e_0 \longrightarrow 5$

$b_0 \; e_1 \longrightarrow 6$

$a_1 \; e_0 \; g_0 \; (a_1 \; f_0 \; g_0) \longrightarrow 7$

$b_1 \; e_1 \; f_1 \; g_1 \longrightarrow 8$

$a_1 \; b_1 \; e_0 \; f_1 \longrightarrow 9$

In parenthesis alternative reducts are given.

This algorithm can be also presented in usual boolean notation as

$e \; g' \; (f' \; g) \longrightarrow 0$

$a' \; f' \; (a' \; g') \longrightarrow 1$

$e \; f' \longrightarrow 2$

$e' \; f' \; g \longrightarrow 3$

$a' \; f \; (a' \; g) \longrightarrow 4$

$b' \; e' \longrightarrow 5$

$b' \; e \longrightarrow 6$

$a \; e' \; g' \; (a \; f' \; g') \longrightarrow 7$

$b \; e \; f \; g \longrightarrow 8$

$a \; b \; e' \; f \longrightarrow 9$

where x and x' denotes variable and its negation.

The algorithm can be implemented in software or in hardware. The latter case will be discussed in more detail in the next chapter.

To make the example more realistic assume that instead of seven elements we are given a grid of sensitive pixels, say 9 x 12, in which seven areas marked *a*, *b*, *c*, *d*, *e*, *f* and *g* as shown in Fig.1 are distinguished.

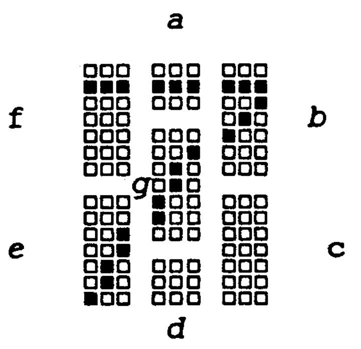


Fig. 1

After slight modification the algorithm will recognize now much more realistic handwritten digits. Alternatively we can also play with the shape of the grid to get more adequate algorithms for character recognition.

The algorithm for digits recognition can be also obtained as a result of learning from examples. This approach will be considered in Chapter 12.

5. Buying a Car

Suppose we are going to buy a car (house, TV set etc.) and several cars are available, having parameters as shown in Table 19.

Car	*Price*	*Mileage*	*Size*	*Max-Speed*	*Acceleration*
1	*low*	*medium*	*full*	*low*	*good*
2	*medium*	*medium*	*compact*	*high*	*poor*
3	*high*	*medium*	*full*	*low*	*good*
4	*medium*	*medium*	*full*	*high*	*excellent*
5	*low*	*high*	*full*	*low*	*good*

Table 19

In order to make the proper choice first investigation of essential differences between the cars seems reasonable.

To this end we need the minimal description of each car in terms of available features, i.e. we have to find minimal decision algorithm as discussed previously.

For simplification of notation we replace Table 19 ~~ba~~ by Table 20

U	*a*	*b*	*c*	*d*	*e*
1	-	0	+	-	0
2	0	0	-	+	-
3	+	0	+	-	0
4	0	0	+	+	+
5	-	+	+	-	0

Table 20

where

a - *Price*
b - *Mileage*
c - *Size*
d - *Max-Speed*
e - *Acceleration*

and values of attributes are coded as follows:

$$V_{Price} = \{low\ (-),\ medium\ (0),\ high\ (+)\}$$

$$V_{Mileage} = \{low\ (-),\ medium\ (0),\ high\ (+)\}$$

$$V_{Size} = \{compact\ (-),\ medium\ (0),\ full (+)\}$$

$$V_{Max\text{-}Speed} = \{low\ (-),\ medium\ (0),\ high\ (+)\}$$

$$V_{Acceleration} = \{poor\ (-),\ good\ (0),\ excellent\ (+)\}$$

The attribute values are coded by symbols in parenthesis.

Now we are ready to proceed to our analysis.

First note that all rows in the table are different. That means that each car has unique characterization in terms of the given features, and the attributes give consistent (in this case we could also say - deterministic) description of each car, or that the corresponding decision rules are true. We would like to know whether features of these cars are independent or not, in order to eliminate from our considerations features which are not necessary. To this end we have to compute first the core and reducts. Compute first the core of attribute. Removing the attribute *a* we get Table 21

U	*b*	*c*	*d*	*e*
1	0	+	–	0
2	0	–	+	–
3	0	+	–	0
4	0	+	+	+
5	+	+	–	0

Table 21

which is inconsistent, because of two identical rows 1 and 3 (underlined). Similarly dropping the attribute *b* we get inconsistent Table 22

U	*a*	*c*	*d*	*e*
1	–	+	–	0
2	0	–	+	–
3	+	+	–	0
4	0	+	+	+
5	–	+	–	0

Table 22

in which rows 1 and 5 are identical. Removing attributes *c*, *d* or *e* we get Tables 23, 24 and 25

U	a	b	d	e
1	-	0	-	0
2	0	0	+	-
3	+	0	-	0
4	0	0	+	+
5	-	+	-	0

Table 23

U	a	b	c	e
1	-	0	+	0
2	0	0	-	-
3	+	0	+	0
4	0	0	+	+
5	-	+	+	0

Table 24

U	a	b	c	d
1	-	0	+	-
2	0	0	-	+
3	+	0	+	-
4	0	0	+	+
5	-	+	+	-

Table 25

which are consistent.

Thus the core of attributes is the set $\{a, b\}$.

There are two reducts $\{a, b, c\}$ and $\{a, b, e\}$ of the set of attributes, i.e. there are exactly two consistent and independent tables

U	a	b	c
1	-	0	+
2	0	0	-
3	+	0	+
4	0	0	+
5	-	+	+

Table 26

U	a	b	e
1	-	0	0
2	0	0	-
3	+	0	0
4	0	0	+
5	-	+	0

Table 27

Thus we have the following dependencies

$\{a, b, c\} \Rightarrow \{d, e\}$ *and* $\{a, b, e\} \Rightarrow \{d, c\}$

Intuitively ~~that~~ this means that the attributes *a* (*Price*) and *b* (*Mileage*) must be taken into the account when discussing the differences between the cars, and attributes *c* (*Size*) and *e* (*Max-Speed*) can be mutually replaced. *The attribute d* (*Acceleration*) depends on the remaining set of attributes and is not a essential issue in this consideration.

Now we are ready to compute minimal decision algorithms which will give description of essential differences between the cars. To this end we have to reduce each decision rule, i.e. to eliminate unnecessary attribute values from the reduced tables. Because there are two reducts we have to repeat the reduction for Tables 26 and 27.

Let us first analyze Table 26.

Removing the attribute *a* from Table 26 we get Table 28

U	*b*	*c*
$\underline{1}$	$\underline{0}$	$\underline{+}$
2	0	−
$\underline{3}$	$\underline{0}$	$\underline{+}$
$\underline{4}$	$\underline{0}$	$\underline{+}$
5	+	+

Table 28

in which the underlined rules are inconsistent (false), i.e. a_-, a_0, $a_+$are core values for rules 1, 4 and 3 respectively. Removing attribute *b* we obtain Table 29

U	*a*	*c*
$\underline{1}$	$\underline{-}$	$\underline{+}$
2	0	−
3	+	+
4	0	+
$\underline{5}$	$\underline{-}$	$\underline{+}$

Table 29

in which rule 1 and 5 are inconsistent, so the attribute values b_0, b_+ are core values for rules 1 and 5. Similarly dropping the attribute *c* we have Table 30

U	*a*	*b*
1	−	0
$\underline{2}$	$\underline{0}$	$\underline{0}$
3	+	0
$\underline{4}$	$\underline{0}$	$\underline{0}$
5	−	+

Table 30

in which rules 2 and 5 are inconsistent, and consequently c_-, are core values for these rules respectively. In Table 31 the core values for Table 26 are listed.

U	a	b	c
1	-	0	
2			-
3	+		
4	0		+
5		+	

Table 31

It turns out the the core values are reduts of the decision rules, so we have the following decision algorithm

$a_- \, b_0 \longrightarrow 1$

$c_- \longrightarrow 2$

$a_+ \longrightarrow 3$

$a_0 \, c_+ \longrightarrow 4$

$b_+ \longrightarrow 5$

or

(Prize, low) $\wedge$ *(Mileage, medium)* $\longrightarrow 1$
(Size, compact) $\longrightarrow 2$
(Prize, high) $\longrightarrow 3$
(Prize, medium) $\wedge$ *(Size, full)* $\longrightarrow 4$
(Mileage, high) $\longrightarrow 5$

Thus each car is uniquely characterized by a proper decision rule and this characterization can serve as a basis for car evaluation. Detailed discussion of the result is left for the reader.

Because there are two reducts we have also another opportunity, using instead of attribute c - attribute e. In this case we get the following reducts

U	a	b	e
1	-	0	
2			-
3	+		
4			+
5		+	

Table 32

and the decision algorithm is

(*Prize*, *low*) ∧ (*Mileage*, *medium*) ⟶ 1
(*Acceleration*, *poor*) ⟶ 2
(*Prize*, *high*) ⟶ 3
(*Acceleration*, *excellent*) ⟶ 4
(*Mileage*, *high*) ⟶ 5

Because we have two reducts, so if we prefer the attribute *Acceleration* to attribute *Size* , we should use the second algorithm, otherwise the first attribute should be employed. Detail discussion of the obtained results are left to the interested reader.

Summary

Knowing differences between various options is often the departure point in decision making. The rough set approach seems to be useful tool to trace the dissimilarities between objects, states, opinions, processes etc.

Excercises

1. Assume in sections 2 and 3 that the attributes have three (or more) values (for example, favorable, neutral and against) and carry out the same analysis, as for the two valued attributes case. What are the consequences of introducing multivalued attributes?

2. Assume that values of attributes in the Exercise 1 are strictly ordered, i.e. that they form a linear scale. For example in the beauty contest the girls may be rated by each referee 0, 1, 2, 3, 4 and 5, meaning that the girl receiving higher score is more beautiful. Strict ordering of values of attributes induces partial order on the set of objects. (In partial ordering some objects can be not comparable, e.g. girls having the same rating can not be ordered according to their beauty). Compare judgments of referees *a*, *c*, *d*, *i* and *j*, as shown in Table 2. Can referees opinion be aggregated so that the group opinion as a whole will not violate individual opinions? (Hint: ReadChapter 8, Group Decision and Social Choice, in French 1986]).

3. Compute core, reducts, all dependencies and minimal decision rules for Table 27.

References

Casti, J. L. (1989). Alternate Realities - Mathematical Models of Nature and man. *John Wiley and Sons.*

Nurmi, H. (1987). Comparing Voting Schemes. *D. Reidel,* Dordrecht.

French, S. (1986). Decision Theory: An Introduction to the Mathematics of Rationality. *Ellis Horwood Limited,* Chichester.

1. Introduction

Logical design of switching circuits has a long and rich history, and there is a variety of methods for digital circuits analysis and simplification (see the enclosed references, and for overview see for example Muroga (1979)). Our claim is that the methods we have just reported in the preceding several chapters can be successfully employed in this area. The best way to begin the discussion is to note that switching circuits performance can be presented as truth tables, which are in fact decision tables with two valued attributes, where condition attributes represent input variables, and decision attributes are to represent output variables of the circuit. Hence truth tables can be simplified in terms of our approach, exactly in the same way as any kind of decision tables, and consequently our method can be viewed as a generalization of switching algebra.

The proposed method seems to be best suited for synthesis of Programmable Logic Array (PLA) (cf. Muroga (1979) pp. 549-559)).

It is interesting to compare the proposed method with alternative approaches, like prime-implicants method (cf. Quine (1955)) or Karnaugh maps (cf. Karnaugh (1953)), but we leave this comparison to the interested reader.

2. Minimization of Partially Defined Switching Functions

We are going to show in this section an example illustrating the basic ideas underlying the application of rough sets philosophy to switching function minimization (cf. Rybnik (1990)) the

Consider partially defined switching function as shown in Table 1,

U	*a*	*b*	*c*	*d*	*e*	*f*
1	0	0	0	1	1	1
2	0	1	0	0	0	1
3	0	1	1	0	0	1
4	1	1	0	0	0	1
5	1	1	0	1	0	1
6	1	1	1	1	0	1
7	1	1	1	0	0	1
8	1	0	0	1	1	1
9	1	0	0	1	0	1
10	0	0	0	0	0	0
11	0	0	0	1	0	0
12	0	1	1	1	0	0
13	0	1	1	0	1	0
14	1	1	1	1	1	0
15	1	0	0	0	1	0

Table 1

where *a*, *b*, *c*, *d* and *e* are input variables, whereas *f* is output variable, as shown in Fig. 1.

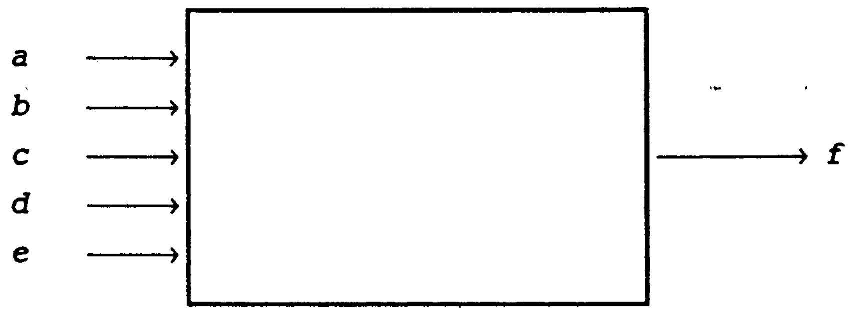


Fig. 1

We assume that combinations of input variable values not present in the table will never occur, and we are interested in designing a "minimal" digital circuit functioning of which is described by the table.

To this end we have to simplify ~~simplified~~ the truth table following the pattern we have described in the earlier chapters, i.e. in the present case we have to reduce the input variables and afterwards drop all unnecessary values of variables.

Observe first that Table 1 is consistent (all combinations of input variable values are different), hence the corresponding switching function is well defined.

Let us start with reduction of input variables.

The variable *a* is *f*-indispensable, because removing variable *a* gives Table 2

U	*b*	*c*	*d*	*e*	*f*
1	0	0	1	1	1
2	1	0	0	0	1
3	1	1	0	0	1
4	1	0	0	0	1
5	1	0	1	0	1
6	1	1	1	0	1
7	1	1	0	0	1
8	0	0	1	1	1
9	0	0	1	0	1
10	0	0	0	0	0
11	0	0	1	0	0
12	1	1	1	0	0
13	1	1	0	1	0
14	1	1	1	1	0
15	0	0	0	1	0

Table 2

which is inconsistent, because in the table pairs of decision rules 6, 12 and 9, 11 are inconsistent, thus input variable *a* is *f*-indispensable.

Removing variable *b* we get Table 3

U	*a*	*c*	*d*	*e*	*f*
1	0	0	1	1	1
2	0	0	0	0	1
3	0	1	0	0	1
4	1	0	0	0	1
5	1	0	1	0	1
6	1	1	1	0	1
7	1	1	0	0	1
8	1	0	1	1	1
9	1	0	1	0	1
10	0	0	0	0	0
11	0	0	1	0	0
12	0	1	1	0	0
13	0	1	0	1	0
14	1	1	1	1	0
15	1	0	0	1	0

Table 3

which is inconsistent because rules 2 and 10 are inconsistent, hence variable *b* is *f*-indispensable.

Without input variable *c* we have Table 4

U	*a*	*b*	*d*	*e*	*f*
1	0	0	1	1	1
2	0	1	0	0	1
3	0	1	0	0	1
4	1	1	0	0	1
5	1	1	1	0	1
6	1	1	1	0	1
7	1	1	0	0	1
8	1	0	1	1	1
9	1	0	1	0	1
10	0	0	0	0	0
11	0	0	1	0	0
12	0	1	1	0	0
13	0	1	0	1	0
14	1	1	1	1	0
15	1	0	0	1	0

Table 4

which is consistent because all decision rules in the table are consistent, Thus the input variable *c* is *f*-dispensable.

Dropping variable *d* we obtain Table 5

U	*a*	*b*	*c*	*e*	*f*
1	0	0	0	1	1
2	0	1	0	0	1
3	0	1	1	0	1
4	1	1	0	0	1
5	1	1	0	0	1
6	1	1	1	0	1
7	1	1	1	0	1
8	1	0	0	1	1
9	1	0	0	0	1
10	0	0	0	0	0
11	0	0	0	0	0
12	0	1	1	0	0
13	0	1	1	1	0
14	1	1	1	1	0
15	1	0	0	1	0

Table 5

which is inconsistent because rules 3 and 12 as well rules 8 and 15 are inconsistent, thus the variable *d* is *f*-indispensable.

Finally removing input variable *e* we have inconsistent Table 6

U	*a*	*b*	*c*	*d*	*f*
1	0	0	0	1	1
2	0	1	0	0	1
3	0	1	1	0	1
4	1	1	0	0	1
5	1	1	0	1	1
6	1	1	1	1	1
7	1	1	1	0	1
8	1	0	0	1	1
9	1	0	0	1	1
10	0	0	0	0	0
11	0	0	0	1	0
12	0	1	1	1	0
13	0	1	1	0	0
14	1	1	1	1	0
15	1	0	0	0	0

Table 6

because pairs of decision rules 1 and 11, 3 and 13 as well and 6 and 14 are inconsistent, thus the input variable *e* is *f*-indispensable.

From this analysis follows that the only *f*-dispensable input variable is *c*, which means that this variable is superfluous in the definition of the switching function described by Table 1. Hence Table 1 can be replaced by Table 4, which is presented in compact form in Table 7.

U	*a*	*b*	*d*	*e*	*f*
1	0	0	1	1	1
2,3	0	1	0	0	1
4,7	1	1	0	0	1
5,6	1	1	1	0	1
8	1	0	1	1	1
9	1	0	1	0	1
10	0	0	0	0	0
11	0	0	1	0	0
12	0	1	1	0	0
13	0	1	0	1	0
14	1	1	1	1	0
15	1	0	0	1	0

Table 7

in which identical rules are combined together.

Now we have to eliminate unnecessary values of input variables, i.e. compute reducts of each decision rule in Table 7, which requires first computation of core values of each decision rule. The result is shown in Table 8.

U	*a*	*b*	*d*	*e*	*f*
1	-	-	-	1	1
2,3	-	1	0	0	1
4,7	-	-	-	-	1
5,6	1	-	-	0	1
8	-	0	1	-	1
9	1	-	-	-	1
10	-	0	-	-	0
11	0	-	-	0	0
12	0	-	1	-	0
13	-	-	-	1	0
14	-	1	-	1	0
15	-	-	0	-	0

Table 8

We recall that removing core values from the table makes the table inconsistent. For example without core values e_1 in rule 1 and e_0 in rule 11 Table 7 becomes inconsistent because we have then in the table the following inconsistent decision rules

$$a_0\ b_0\ d_1 \longrightarrow f_1 \text{ and } a_0\ b_0\ d_1 \longrightarrow f_0.$$

One can check that core values of rules 2 and 3, 5 and 6 as well as rule 14 are also reduced values, i.e. the decision rules

$$b_1\ d_0\ e_0 \longrightarrow f_1$$

$$a_1\ e_0 \longrightarrow f_1$$

$$b_1\ e_1 \longrightarrow f_0$$

are reduced, whereas the remaining core values do not form reducts. According to ideas given in chapter 3.4 we have to add to the above decision rules minimal set of such reduced decision rules that union of its all condition for each decision class covers the whole decision class. Because

$$|b_1\ d_0\ e_0| = \{2, 3, 4, 7\}$$

$$|a_1\ e_0| = \{4, 5, 6, 7, 9\}$$

we need for decision class "1" rules covering set {1, 8}. From Table 7 we find the the missing rules are

$$a_0\ d_1\ e_1 \longrightarrow f_1$$

$$b_0\ d_1\ e_1 \longrightarrow f_1$$

$$a_0\ b_0\ e_1 \longrightarrow f_1$$

$$a_1\ b_0\ d_1 \longrightarrow f_1$$

for which we have ,

$$|a_0\ d_1\ e_1| = \{1\}$$

$$|b_0\ d_1\ e_1| = \{1,\ 8\}$$

$$|a_0\ b_0\ e_1| = \{1\}$$

$$|a_1\ b_0\ d_1| = \{8,\ 9\}$$

That means that we have the following three minimal solution to our problem

(i) $b_1\ d_0\ e_0 \longrightarrow f_1$

$a_1\ e_0 \longrightarrow f_1$

$\overset{b}{\cancel{a}}_0\ d_1\ e_1 \longrightarrow f_1$

(ii) $b_1\ d_0\ e_0 \longrightarrow f_1$

$a_1\ e_0 \longrightarrow f_1$

$\overset{a}{\cancel{b}}_0\ d_1\ e_1 \longrightarrow f_1$

$a_1\ b_0\ d_1 \longrightarrow f_1$

(iii) $b_1\ d_0\ e_0 \longrightarrow f_1$

$a_1\ e_0 \longrightarrow f_1$

$a_0\ b_0\ e_1 \longrightarrow f_1$

$a_1\ b_0\ d_1 \longrightarrow f_1$

or

(i) $b_1 d_0 e_0 \vee a_1 e_0 \vee b_0 d_1 e_1 \longrightarrow f_1$

(ii) $b_1 d_0 e_0 \vee a_1 e_0 \vee a_0 d_1 e_1 \vee a_1 b_0 d_1 \longrightarrow f_1$

(iii) $b_1 d_0 e_0 \vee a_1 e_0 \vee a_0 b_0 e_1 \vee a_1 b_0 d_1 \longrightarrow f_1$

Proceeding in the same way for decision class "0" we obtain the following four minimal solutions:

(i) $b_1 e_1 \vee b_0 d_0 \vee a_0 d_1 e_0 \longrightarrow f_0$

(ii) $b_1 e_1 \vee b_0 d_0 \vee a_0 b_0 e_0 \vee a_0 b_1 d_0 \longrightarrow f_0$

(iii) $b_1 e_1 \vee a_0 b_0 e_0 \vee a_0 d_1 e_0 \vee d_0 e_1 \longrightarrow f_0$

(iv) $b_1 e_1 \vee a_0 b_0 e_0 \vee a_0 b_1 d_1 \vee d_0 e_1 \longrightarrow f_0$

As a consequence we have 12 minimal solution to the problem. One of them requiring minimal number of gates is given below

$$b_1 d_0 e_0 \vee a_1 e_0 \vee b_0 d_1 e_1 \longrightarrow f_1$$

$$b_1 e_1 \vee b_0 d_0 \vee a_0 d_1 e_0 \longrightarrow f_0$$

which can written in more convenient form

$$b\ d'\ e' \vee a\ e' \vee b'\ d\ e \longrightarrow f$$

$$b\ e \vee b'\ d' \vee a'\ d\ e' \longrightarrow f'$$

where x and x' denote variables without and with negation respectively.

The corresponding switching circuit is shown in Fig. 2. ? (no Fig 2. !)

3. Multiple-Output Switching Functions

In the previous section we have discussed partially defined switching function with a single output. The same method can be employed for synthesis of multiple output switching functions, by treating this kind of functions as a set of single output switching functions, and applying to these functions exactly the same procedures as before. However in many cases we can get better results, if we treat the set of all binary output variables as a single multivalued variable, as it will be shown in the next example.

Consider switching circuit with four input variables *a*, *b*, *c*, *d*, and two output variables *e*, *f*, as shown in Fig. 3.

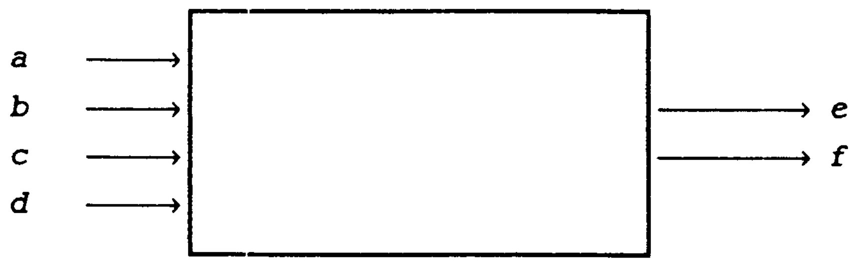


Fig. ~~2~~ 3

Corresponding truth table is given in Table 9.

U	*a*	*b*	*c*	*d*	*e*	*f*
1	0	0	0	1	1	1
2	1	1	1	1	0	1
3	1	1	1	0	0	1
4	1	0	0	1	1	1
5	1	0	1	1	0	1
6	0	0	0	0	0	0
7	0	1	0	1	0	0
8	0	1	1	1	0	0
9	0	1	1	0	1	0
10	1	0	0	0	1	0

Table 9

Replace variables *e* and *f* by a single four valued variable *y*, with values (0, 0), (0, 1) (1, 0) and (1, 1). Table 1 can be presented now as

U	*a*	*b*	*c*	*d*	*y*	
1	0	0	0	1	(1	1)
2	1	1	1	1	(0	1)
3	1	1	1	0	(0	1)
4	1	0	0	1	(1	1)
5	1	0	1	1	(0	1)
6	0	0	0	0	(0	0)
7	0	1	0	1	(0	0)
8	0	1	1	1	(0	0)
9	0	1	1	0	(1	0)
10	1	0	0	0	(1	0)

Table 10

For the sake of simplicity let us denote the four values of output variable *y* by i, ii, iii and iv respectively and let us also group and renumerate rows of the table with respect to values of the output variable, as shown below.

U	*a*	*b*	*c*	*d*	*y*
1	0	0	0	0	i
2	0	1	0	1	i
3	0	1	1	1	i
4	1	1	1	1	ii
5	1	1	1	0	ii
6	1	0	1	1	ii
7	0	1	1	0	iii
8	1	0	0	0	iii
9	0	0	0	1	iv
10	1	0	0	1	iv

Table 11

It is easy to see that the table is consistent, thus the switching function is well defined. Now let us compute *y*-reducts of input variables. To this end we have to compute the *y*-dispensable input variables.

Removing input variable *a* we get Table 12

U	*b*	*c*	*d*	*y*
1	0	0	0	i
2	1	0	1	i
3	1	1	1	i
4	1	1	1	ii
5	1	1	0	ii
6	0	1	1	ii
7	1	1	0	iii
8	0	0	0	iii
9	0	0	1	iv
10	0	0	1	iv

Table 12

which is inconsistent,because the following pairs of rules (1, 8), (3, 4) and (5, 7) are inconsistent. Thus the input variable *a* is *y*-indispensable.

Removing input variable *b* we get Table 13

U	*a*	*c*	*d*	*y*
1	0	0	0	i
2	0	0	1	i
3	0	1	1	i
4	1	1	1	ii
5	1	1	0	ii
6	1	1	1	ii
7	0	1	0	iii
8	1	0	0	iii
9	0	0	1	iv
10	1	0	1	iv

Table 13

we get again inconsistent table because the rules 2 and 9 are inconsistent.

Next we drop the input variable *c* and we obtain Table 14

U	*a*	*b*	*d*	*y*
1	0	0	0	i
2	0	1	1	i
3	0	1	1	i
4	1	1	1	ii
5	1	1	0	ii
6	1	0	1	ii
7	0	1	0	iii
8	1	0	0	iii
9	0	0	1	iv
10	1	0	1	iv

Table 14

which is also inconsistent, because rules 6 and 10 are inconsistent.

Finally removing input variable *d* we have Table 15

U	*a*	*b*	*c*	*y*
1	0	0	0	i
2	0	1	0	i
3	0	1	1	i
4	1	1	1	ii
5	1	1	1	ii
6	1	0	1	ii
7	0	1	1	iii
8	1	0	0	iii
9	0	0	0	iv
10	1	0	0	iv

Table 15

we get also an inconsistent table, since the pairs of rules (1, 9), (3, 7) and (8, 10) are inconsistent.

Hence all input variables are independent and none of them can be eliminated.

Now we are to eliminate unnecessary values of variables from the truth table, which requires first computation of core values of input variables and they are listed in Table 16.

U	*a*	*b*	*c*	*d*	*y*
1	0	-	-	0	i
2	-	1	-	-	i
3	0	-	-	1	i
4	1	-	-	-	ii
5	1	-	-	-	ii
6	-	-	1	-	ii
7	0	-	-	0	iii
8	1	-	-	0	iii
9	-	0	-	1	iv
10	-	-	0	1	iv

Table 16

and list of all reduced rules are ~~gibe~~ given in the table below

U	*a*	*b*	*c*	*d*	*y*
1	0	-	0	0	i
1′	0	0	-	0	i
2	-	1	0	-	i
2′	0	1	-	1	i
3	0	-	1	1	i
3′	0	1	-	1	i
4	1	1	-	-	ii
4′	1	-	1	-	ii
5	1	1	-	-	ii
5′	1	-	1	-	ii
6	-	0	1	-	ii
6′	1	-	1	-	ii
7	0	1	-	0	iii
7′	0	-	1	0	iii
8	1	0	-	0	iii
8′	1	-	0	0	iii
9	0	0	-	1	iv
9′	-	0	0	1	iv
10	1	-	0	1	iv
10′	-	0	0	1	iv

Table 17

There are several minimal solutions to our problem, and that requiring minimal number of gates is given in Table 18.

U	a	b	c	d	y
1′	0	0	-	0	i
3′	0	1	-	1	i
4′,5′,6′	1	-	1	-	ii
7	0	1	-	0	iii
8	1	0	-	0	iii
9′,10′	-	0	0	1	iv

Table 18

which is equivalent to the following decision algorithm

$a_0\, b_0\, d_0 \longrightarrow$ i

$a_0\, b_1\, d_1 \longrightarrow$ i

$a_1\, c_1 \longrightarrow$ ii

$a_0\, b_1\, d_0 \longrightarrow$ iii

$a_1\, b_0\, d_0 \longrightarrow$ iii

$b_0\, c_0\, d_1 \longrightarrow$ iv

or

$a_0\, b_0\, d_0 \vee \cancel{a_0}\, b_1\, d_1 \longrightarrow$ i

$a_1\, c_1 \longrightarrow$ ii

$a_0\, b_1\, d_0 \vee a_1\, b_0\, d_0 \longrightarrow$ iii

$b_0\, c_0\, d_1 \longrightarrow$ iv

The algorithm can also be presented using boolean algebra notation

$a' \ b' \ d' \longrightarrow$ i

$a' \ b \ d \longrightarrow$ i

$a \ c \ \longrightarrow$ ii

$a' \ b \ d' \longrightarrow$ iii

$a \ b' \ d' \longrightarrow$ iii

$b' \ c' \ d \longrightarrow$ iv

or

$a' \ b' \ d' \vee a' \ b \ d \longrightarrow$ i

$a \ c \ \longrightarrow$ ii

$a' \ b \ d' \vee a \ b' \ d' \longrightarrow$ iii

$b' \ c' \ d \longrightarrow$ iv

The corresponding switching circuit is shown in Fig. 4. We have to bear in mind however that our aim is to design switching circuit with two binary output variables *e* and *f*. To this end it is enough to add simple circuit, which will translate four valued output to the required binary outputs, but we left this task to the interested reader.

Summary

We hope we have shown that the rough set approach can be used to switching circuits synthesis. The method seem to be best suited to two-level single as well multiple-output circuits and Programmable ~~L~~ogic Array synthesis, however further research in this area is necessary.

Excercises

1. Find all minimal solutions of Table 9.

2. Find minimal truth tables of Table 9 assuming single output switching circuits with output variables *e* and *f*, and compare the obtained results with the multiple-output solution presented in section 3.

3. Find all minimal solution of truth table given below.

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

Table 19

a) Assume that *a*, *b*, *c*, *d* and *e* are input variable and *f* is output variable.

b) Assume *a*, *b*, *c* and *d* as input variables and *e*, *f* as output variables.

4. Find all minimal solution for Table 20 (cf. Crama et al. (1988).

U	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
1	0	1	0	1	0	1	1	0	1
2	1	1	0	1	1	0	0	1	1
3	0	1	1	0	1	0	0	1	1
4	1	0	1	0	1	0	1	0	0
5	0	0	0	1	1	1	0	0	0
6	1	1	0	1	0	1	0	1	0
7	0	0	1	0	1	0	1	0	0

Table 20

Compare the number of solutions for Table 19 and Table 20. Explain the difference.

Hint: Compute the core of input variables in both cases. Can the number of solutions be attributed the the "size" of the core?

References

Caldwell, S. H. (1958). Switching Circuits and Logical Design. *John Wiley and Sons.*

Crama, Y., Hammer, P. L. and Ibaraki, T. (1988). Cause-Effect Relationship and Partially Defined Boolean Functions. *Rutcor Research Report*, RRR # 39-88, August.

Cutler, R. B. and Muroga, S. (1980). Useless Prime Implicants of Incompletely Specified Multiple-Output Switching Functions. *International Journal of Computer and Information sciences*, **4**, pp. 337-350.

Dunham, B. and Friedshal, R. (1959). The Problem of Simplifying Logical Expressions. *J. Symb. Logic*, pp. 17-19.

Grasseli, A. and Luccio, F. (1966). A method of for the Combined Row-Column Reduction of Flow Tables. *IEEE Conference Record, Seventh Annual Symposium on Switching Theory and Automata Theory*, pp. 136-147.

Hill, F. J. and Peterson, G. R. (1974). Introduction to Switching Theory and Logical Design. 2nd ed., *John Wiley and Sons.*

Karnaugh, M. (1953). The Map Method for Synthesis of Combinational Logic Circuits.*Commun. Electron.*, November, pp. 593-599.

Lawer, E. L. (1964). An Approach to Multi-Level Boolean Minimization. *J. ACM., July*. pp. 283-295.

Lee, C. Y. (1959). Representation of Switching Function by Binary Decision Programs. *Bell Syst. Tech. J.*, pp. 985-999.

Lee, S. C. (1976). Digital Circuits and Logical Design. *Prentice-Hall.*

Maley, G. A. (1970). Manual of Logical Design. *Prentice-Hall.*

McClusky, E. J. ((1965). Introduction to the Theory of Switching Circuits. *McGraw-Hill.*

Muroga, M. (1979). Logic Design and Switching Theory. *John Wiley and Sons.*

Nelson, R. J. (1954). Simplest Normal Truth Functions. *J. Symb. Logic, June*, pp. 105-108.

Paul, M. C. and Unger, S. H. (1959). Minimizing the Number of State Incompletely Specified Switching Functions. *IRE Trans. Electron. Com., September*, pp. 356-367.

Quine, W. V. (1955). A way of Simplify Truth Functions. *Amer. Math. Mon., November*, pp. 627-631.

Reush, B. (1975). Generation of Prime Implicants from Subfunctions and a Unifying Approach to the Covering problem. *IEEE Trans. Comput., September*, pp. 924-930.

Roth, J. P. (1958). Algebraic Topological methods for the Synthesis of Switching Systems, I. *Trans. Amer. Math. Soc., July*, pp. 301-326.

Rybnik, J. (1990). Minimization of Partially Defined Switching Functions Using Rough Sets. *Ph. D. Dissertation.*

12. MACHINE LEARNING

1. Introduction

Machine learning is one of the most important fields of artificial intelligence, and growing number of researchers is involved in this area. There is a variety of approaches to machine learning, however by now no commonly accepted theoretical foundations has been developed. It seems that the rough set approach can be used as a theoretical basis for some problems in machine learning. In this section we are going to outline briefly some ideas showing how the concepts introduced so far can be used in this area. We do not pretend however to give full account for machine learning in the rough set perspective.

We will not go into details about machine learning, and in order to avoid unnecessary technicalities we will skip considerations on the present state and current research in this area, and the reader unfamiliar with the subject is advised to consult, for example, the book of Forsyth and Rada (1986), where fundamentals about machine learning and the relevant literature can be found.

More about rough set view on various aspects of machine learning can be found in the enclosed literature.

2. Learning from Examples

In our setting the process of learning from examples can be formulated as follows.

Assume that there are two agents: a "knower" (a teacher, an expert, an environment, etc) and a "learner" (a student, a robot, etc).

We assume that the knower has knowledge about certain universe of discourse *U*, that is, according to the ideas given in the previous chapters, he is able to classify elements of the universe *U*, and classes of the knower's classification form concepts to be learned by the learner. Moreover, we assume in this section that the knower has *complete* knowledge about the universe *U*, that means he is able to classify every object of the universe *U*. Furthermore we assume that the universe *U* is *closed* that is nothing else besides *U* exists, i.e. *U* is the whole world of the knower. This will be called the *Closed World Assumption* (*CWA*).

The task of a learner is to learn knower's knowledge. For the sake of illustration we might assume ~~the~~ that the learning process runs as follows. Assume that the learner is able to distinguish some features (attributes) of objects of the universe *U*, for example shape, color etc. The knower exhibits step by step each object to the learner together with the information ~~which~~ concept it represents. The learner is suppose to find characteristic features of each concept on the basis of features of all instances of the concept, or in other words the learning consists in finding description of the knower's concepts in terms of attributes of objects available to the learner. More precisely the learner has to derive decision rules from examples delivered by the knower - which enable proper classification of objects on the basis of theirs features. For example the knower can demonstrate to the learner various kind of hand written characters and the task of the learner is to find characteristic features of each character. Thus learning from examples in the case of *CWA* can be reduced to the derivation of decision algorithms from decision tables, considered in the previous chapters, in which condition attributes are identical with learner's attributes and the knower's knowledge is represented by decision attribute in the decision table.

The question arises whether always the learner's knowledge can match the knower's knowledge or whether the learner is always able to learn concepts demonstrated by the knower. In our terminology it can be expressed as, whether the knower's knowledge can be expressed (exactly or approximately) in terms of learner's knowledge, or in other words whether the knower's knowledge (attribute) depends on learner's knowledge (attributes).

From our considerations follows that this may be not always the case, which means that some concepts can not be learned, since the set of learner's attributes can not permit to express some concepts. This follows directly from the topological classification of rough sets given in Chapter 2

Remark

It seems that in the present literature on machine learning the impossibility of learning of some concepts has been missed. ■

As a consequence the degree of dependency between the set of knower's and learner's attributes (see Chapter 4, Section 3), can be used as a numerical measure of how exactly the knower's knowledge can be learned. More exactly, if B and C are learner's and knower's attributes respectively, and $B \Rightarrow_k C$, where

$$k = \gamma_B(C) = \frac{card\ POS_B(C)}{card\ U}$$

then k will be called the *quality of learning*. This number expresses what percentage of knower's knowledge can be learned by the learner.

We will illustrate this idea by the following example.

Example 1.

Consider the following *KR*-System

U	*a*	*b*	*c*	*d*	*e*
1	1	2	0	1	1
2	1	2	0	1	1
3	2	0	0	1	0
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
7	2	0	0	1	0
8	0	1	2	2	1
9	2	1	0	2	2
10	2	0	0	1	0

Table 1

Suppose that $B = \{a, b, c, d\}$ is the set of learner's attributes and assume that e is a knower's attribute.

Thus the knower knowledge consists of three concepts $|e_0| = \{3, 7, 10\}$, $|e_1| = \{1, 2, 4, 5, 8\}$ and $|e_2| = \{6, 9\}$, which for simplicity will be denoted as X_0, X_1 and X_2 respectively.

The learner's knowledge consists of the following basic concepts

$$Y_0 = |a_1\ b_2\ c_0\ d_1| = \{1, 2\}$$

$$Y_1 = |a_2\ b_0\ c_0\ d_1| = \{3, 7, 10\}$$

$$Y_2 = |a_0\ b_0\ c_1\ d_2| = \{4, 6\}$$

$$Y_3 = |a_2\ b_1\ c_0\ d_2| = \{5, 9\}$$

$$Y_4 = |a_0\ b_1\ c_2\ d_2| = \{8\}$$

which are equivalence classes of the relation *IND* (B).

To learn knower's knowledge means to express each knower's basic concept by means of learner~~'s~~s basic concepts, i.e. we have to check whether or not each equivalence class of the relation *IND* $\{e\}$ is union of some equivalence classes of the relation *IND* (B), or in other words whether $B \Rightarrow \{e\}$ or not. To this end we have to compute approximations of knower's basic concepts, in terms of learner's basic concepts, which are listed below.

$$\underline{B}X_0 = \bar{B}X_0 = X_0 = Y_0 = \{3, 7, 10\}$$

$$\underline{B}X_1 = Y_0 \cup Y_4 = \{1, 2, 8\}$$

$$\bar{B}X_1 = Y_0 \cup Y_2 \cup Y_3 \cup Y_4 = \{1, 2, 4, 5, 6, 8, 9\}$$

$$\underline{B}X_2 = \emptyset$$

$$\bar{B}X_2 = Y_2 \cup Y_3 = \{4, 5, 6, 9\}$$

Hence concept X_0 is exact and can be learned fully. Concept X_1 is roughly B-definable, which means that the concept can be learned only with some approximation, i.e. the learner can learn that instances 1,2 and 8 belong to the concept X_1 (are positive instances of $X_{\underset{1}{\cancel{0}}}$), instances 3, 7, 10 do not belong to the concept (are negative instances of the concept), instances 4, 5, 6 and 9 can not be decided by the learner whether they belong to X_1 or not (are border-line examples). Concept X_2 is internally B-undefinable, since there are no~~t~~ positive instances of the concept. The following 1, 2, 3, 7, 8 and 10, are negative instances of the concept, and the remaining examples 4, 5, 6 and 9 are border-line cases. Let us also note that according to Theorem 5 in Chapter 2 each knower's concepts in this case must have negative examples, which are

{1, 2, 4, 5, 6, 8, 9}

{3, 7, 10}
~~{3, 5, 7, 8, 9, 10}~~

{1, 2, 3, 7, 8, 10}

for X_0, X_1, X_2 respectively.

Because $POS_B\{e\} = \{1, 2, 3, 7, 8, 10\}$ only those instances can be properly classified by the learner thus the quality of learning in this case will be

$$\gamma\ \{e\} = \frac{card\ \{1, 2, 3, 7, 8, 10\}}{card\ \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}} \cancel{=}$$

$$= \frac{6}{10}$$

In other words, instances 4, 5, 6 and 9 can not be classified by the learner and concepts containing these instances can not be learned exactly. That means that only instances belonging to the positive region can be classified properly and consequently no~~n~~ empty lower approximation of concepts can be learned.

In order to find the minimal description of each knower's concept (i.e. minimal decision algorithms) we have to proceed as shown in the previous chapters but we will omit the detailed discussion of this task and leave it to the interested reader. Exemplary decision algorithm for Table 1 is shown below:

$$a_2\, b_0 \longrightarrow e_0$$

$$a_1 \longrightarrow e_1$$

$$a_0\, b_1 \longrightarrow e_1$$

Another exemplary decision algorithm is given next.

$$b_0\, c_0 \longrightarrow e_0$$

$$b_2 \longrightarrow e_1$$

$$b_1\, c_2 \longrightarrow e_1$$

Because the concept X_2 has no~~t~~ positive examples there are not decision rules corresponding to this concept. ∎

The question arises whether all instances are necessary to learn the knower's knowledge or not. The following example will illustrate the idea more closely.

Example 2.

Consider Table 1 given in Example 1.

U	*a*	*b*	*c*	*d*	*e*
1	1	2	0	1	1
2	1	2	0	1	1
3	2	0	0	1	0
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
7	2	0	0	1	0
8	0	1	2	2	1
9	2	1	0	2	2
10	2	0	0	1	0

Table 2

It is easily seen that instances given in Table 3 will provide the same decision rules as Table 2.

U	a	b	c	d	e
1	1	2	0	1	1
3	2	0	0	1	0
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
8	0	1	2	2	1
9	2	1	0	2	2

Table 3

Similarly Table 4 will give the same decision rules as Table 2

U	a	b	c	d	e
2	1	2	0	1	1
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
8	0	1	2	2	1
9	2	1	0	2	2
10	2	0	0	1	0

Table 4

However if we drop instances 4 and 8 from Table 2 we obtain Table 5 which yields different decision rules to that obtained from Table 2, as shown below.

U	a	b	c	d	e
1	1	2	0	1	1
2	1	2	0	1	1
3	2	0	0	1	0
5	2	1	0	2	1
6	0	0	1	2	2
7	2	0	0	1	0
9	2	1	0	2	2
10	2	0	0	1	0

Table 5

We miss of course decision rule generated from instance 8, but moreover because of removing instance 4 we obtain a new decision rule corresponding to instance 6

$$a_0\ b_0 \longrightarrow e_2$$

and the whole new decision algorithm will now have the form

$$a_2\ b_0 \longrightarrow e_0$$

$$a_1 \longrightarrow e_1$$

$$a_0\ b_0 \xrightarrow{} e_2$$

which means that ~~that~~ the concept X_2 is now internally definable. Similarly if we drop instance 9 from Table 2 we get Table 6

U	a	b	c	d	e
1	1	2	0	1	1
2	1	2	0	1	1
3	2	0	0	1	0
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
7	2	0	0	1	0
8	0	1	2	2	1
10	2	0	0	1	0

Table 6

and the new decision rule will result

$$a_2 b_1 \longrightarrow e_2$$

which yields new decision algorithm

$$a_2 b_0 \longrightarrow e_0$$

$$a_1 \longrightarrow e_1$$

$$a_0 b_1 \longrightarrow e_1$$

$$a_2 b_1 \longrightarrow e_2$$

Again concept X_2 is now internally definable. ■

One can see from the above examples that some instances are crucial for concept learning but some are not. The general formulation of this problem is rather easy and is left to the interested reader.

3. The Case of an Imperfect Teacher

In this section we are going to discuss briefly how lack of knowledge by the knower (i.e. his impossibility to classify some objects) would affect learner's ability to learn and in particular whether the learner is able to discover the knower's deficiency.

Let us first give some illustrative examples.

Example 3.

Suppose we are given the following *KR*-system

U	*a*	*b*	*c*
1	0	2	+
2	0	1	+
3	1	0	+
4	0	1	0
5	1	0	0
6	1	1	-
7	2	1	-
8	0	1	-
9	1	0	-

Table 7

where $B = \{a, b\}$ is the set of learner's attributes and c is knower's attribute. There are two concepts in the table, X_+ and X_-, denoted by + and - values of knower's attribute c respectively. Value 0 of attribute c denotes that the knower is unable to classify corresponding objects, and the set X_0 of all objects which cannot be classified by the knower will be called the *knower's ignorance region* (in short *ignorance region*). We will also need the set $X^* = X_+ \cup X_-$ which will be refereed to as the *knower's competence region* (in short *competence region*). Thus we have in the universe U three sets:

$X_+ = \{1, 2, 3\}$
$X_- = \{6, 7, 8, 9\}$
$X_0 = \{4, 5\}$

Let us compute whether sets X_+, X_- and X_0 are definable in terms of attributes a and b, i.e. whether they can be learned employing these attributes. To this end we have to compute the lower and the upper approximations of the above sets which are given below:

$\underline{B}\ (X_+) = \{1\}$

$\bar{B}\ (X_+) = \{1, 2, 3, 4, 5, 8, 9\}$

$\underline{B}\ (X_-) = \{6, 7\}$

$\bar{B}\ (X_-) = \{2, 3, 4, 5, 6, 7, 8, 9\}$

$\underline{B}(X_0) = \emptyset$

$\bar{B}(X_0) = \{2, 3, 4, 5, 8, 9\}$

Because $\underline{B}(X_0) = \emptyset$ hence the ignorance region X_0 is internally B-undefinable, which means that the learner is unable to characterize the set X_0 in terms of his attributes, cannot
i.e. discover the knower's ignorance.

As to the second question, whether knower's deficiency is important and influences the learner's ability to learn, let us compute the boundary region of the competence region $X^* = X_+$ and X_-.

$$BN_B(X^*) = BN_B(X_+) \underset{\cap}{\cancel{\cup}} BN_B(X_-) = \{2, 3, 4, 5, 8, 9\}$$

It is easy to check that by every substitution for value 0 in attribute c, values + or - the approximations, and consequently, the boundary region of the competence region X^* remain unchanged. That means that the knower's lack of knowledge is unessential and the fact that he failed in classifying examples 4 and 5 does not disturb the learning process. This is due to the fact that both instances 4 and 5 are in the boundary region of both concepts (i.e. of the competence region) and as such cannot be learned. Thus it does not matter whether the knower knows how to classify them or not.

Example 4.

Consider now another decision table given below

U	a	b	c
1	0	2	+
2	0	1	+
3	1	0	+
4	0	1	+
5	1	0	0
6	1	1	0
7	2	1	-
8	0	1	-
9	1	0	-

Table 8

In this table we have

$X_+ = \{1, 2, 3, 4\}$

$X_- = \{7, 8, 9\}$

$X_0 = \{5, 6\}$

The approximations of these sets are given next.

$\underline{B}(X_+) = \{1\}$

$\bar{B}(X_+) = \{1, 2, 3, 4, 5, 8, 9\}$

$\underline{B}(X_-) = \{7\}$

$\bar{B}(X_-) = \{2, 3, 4, 5, \overset{7}{\swarrow} 8, 9\}$

$\underline{B}(X_0) = \{6\}$

$\bar{B}(X_0) = \{3, 5, 6, 9\}$

Because $\underline{B}(X_0) = \{6\}$ the learner can discover that the knower is unable to classify object 6.

Again, if we replace value 0 in the attribute c by value + or - instance 5 would still remain in the boundary region, whereas instance 6 would belong to positive region of the competence region X^*. That means that the knower's disability to classify instance 6 is important and lack of its classification affects essentially the learning process.

■

The general formulation of problems pointed out in these examples is rather straightforward and is left for the interested reader.

4. Inductive Learning

In learning considered in the previous sections we have assumed that the set of instances U is constant and unchanged during the learning process. In many real life situations however this is not the case and new instances can be added to the set U. The problem arises how these new instances can change the already acquired learner's knowledge.

We assume that every new instance is classified by the knower and the learner is suppose to classify it too on the basis of his actual knowledge.

The initial set *U* will be called the *training set*, and the problem arises whether knowledge acquired on the basis of the training set suffices to classify by the learner every new instance.

The training set may be conside~~r~~ed as a sample of a unknown universe and the problem arises whether the sample suffices to learn the whole knowledge, i.e. to learn the ability to classify every new instance. This is usually called the generalization of knowledge, and is a basis of inductive learning. Many questions arise in connection with this kind of learning and a variety of research has been done in this area (cf. Forsyth et al. 1986) but we are not going to discuss the problem in details and only point out some ideas which may be of interest in connection with the rough set view of this area.

Generalization can be called the *Open World Assumption* (*OWA*) that is we assume in this case that the whole concept is unknown to the knower and only certain instances of the concept are known and the question arises whether it is possible to generalize the sample in such a way that every new instance which satisfies the conditions induced from the sample belongs to the concept in question. An example of white and black swans is the commonly given illustration in this context.

should be [illegible]

Let us consider now the case of *OWA*. Suppose we are given a *KR*-system $S = (U, S)$ and a new instance x is added to U (i.e. $x \notin U$). The set U will be called the *training set*.

We will say that the training set is *consistent* if the *KR*-system is consistent and the training set is *inconsistent* otherwise.

The problem we are going to discuss now is what happens when a new instance is added to the training set U, or to be more specific how concepts learned so far are influenced by new instances. Let us first consider some examples which will give some intuitive background to the problem.

Example 3

Let us consider again *KR*-system given in Example 1.

U	*a*	*b*	*c*	*d*	*e*
1	1	2	0	1	1
2	1	2	0	1	1
3	2	0	0	1	0
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
7	2	0	0	1	0
8	0	1	2	2	1
9	2	1	0	2	2
10	2	0	0	1	0

Table 7

and suppose we add a new instance as shown in Table 8.

U	*a*	*b*	*c*	*d*	*e*
1	1	2	0	1	1
2	1	2	0	1	1
3	2	0	0	1	0
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
7	2	0	0	1	0
8	0	1	2	2	1
9	2	1	0	2	2
10	2	0	0	1	0
11	0	1	2	2	1

Table 8

It is obvious that the new instance does not change the decision algorithm, that means that the learned concepts will remain the same. However if we add instance as shown in Table 9

U	a	b	c	d	e
1	1	2	0	1	1
2	1	2	0	1	1
3	2	0	0	1	0
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
7	2	0	0	1	0
8	0	1	2	2	1
9	2	1	0	2	2
10	2	0	0	1	0
11	1	2	0	1	0

Table 9

the decision algorithm would have the form

$$a_2\ b_0 \longrightarrow e_0$$

$$a_0\ b_1 \longrightarrow e_1$$

In this case the new instance changes the decision algorithm. Finally let us add instance as shown in Table 10

U	a	b	c	d	e
1	1	2	0	1	1
2	1	2	0	1	1
3	2	0	0	1	0
4	0	0	1	2	1
5	2	1	0	2	1
6	0	0	1	2	2
7	2	0	0	1	0
8	0	1	2	2	1
9	2	1	0	2	2
10	2	0	0	1	0
11	1	0	0	1	3

Table 10

Again we obtain new decision algorithm, as shown below:

$a_2\ b_0 \longrightarrow e_0$

$a_1 \longrightarrow e_1$

$a_0\ b_1 \longrightarrow e_1$

$a_1\ b_0\ c_0\ d_1 \longrightarrow e_3$

■

As we have seen from the above examples adding a new instance to the universe we face three possibilities:

1) The new instance *confirms* actual knowledge

2) The new instance *contradicts* actual knowledge

3) The new instance is completely *new case*

In order to show how a new instance affects the actual knowledge gained by the learner from examples considered so far we have to compute how the quality of learning would change in each of the above situations.

If the new instance match one of the existing classes, or form completely new class, then the quality of learning would be

$$\gamma_B(C) = \frac{card\ POS_B(C) + 1}{card\ U + 1}$$

If the new instance contradicts the knowledge already acquired we get

$$\gamma_B(C) = \frac{card\ POS_B(C) - card\ \ x]_{IND\ (B)}}{card\ U + 1}$$

where $x]_{IND(B)}$ denotes the equivalence class containing the new instance x.

Let us briefly comment the above formulas.

If the new instance confirms the acquired knowledge and $U = POS_B(C)$, i.e there are no~~t~~ border-line instances th~~a~~en the new example does not extend learner's knowledge in th~~at~~e sense that the quality of learning is equal to 1 and does not change~~s~~d with new confirming examples. In other words if the training set ha~~ve~~s all possible kinds of objects, adding new object does not increase~~s~~ our knowledge, and every new example can be properly classified.

If however $U \neq POS_B(C)$, i.e. there are border-line examples then the quality of learning increases "slightly" with every new confirmation. If however the new instance will belong to the boundary region then the quality of learning will be

$$\gamma_B(C) = \frac{card\ POS_B(C)}{card\ U + 1}$$

which means that quality of learning will decrease.

The most interesting case is when the new instance will contradict knowledge acquired so far. That means that the new instance x has the same description as some other object y however x and y belong to different classes according to the knower's opinion. In other words these objects are different according to knower's opinion, while the learner is unable to distinguish them employing his attributes. This leads to contradiction and improper classification of the new instance by the learner. This cause that all instances with the same description as x in terms of learner's attributes must be now reclassified, which cau~~se~~ses ~~that~~ the quality of learning may decrease drastically, as shown by the corresponding formula.

To sum up, if the training set is in a certain sense complete, i.e. the corresponding decision table is consistent it provides the highest quality of learning and the learners' knowledge can not be improved by means of new instances. If however the training set is inconsistent every new confirming instance increases learners knowledge and any new border-line instance decreases his knowledge. That means that if the training set is inconsistent the learner's knowledge can be increased by properly chosen new instances. However new instances which contradict actual learner's knowledge decreases quality of learning and the larger is the "loss" of learner's knowledge, the larger is the amount of improperly classified examples before.

Summary

Rough set philosophy seems to be useful as a theoretical basis for some problems in machine learning. Exemplary ideas has been briefly outlined in this chapter.

Excercises

1. Explain examples considered in the previous chapters in terms of machine learning. Assume the Closed and Open Word Assumption.

2. Give general formulation of problems discussed in the previous sections of this Chapter.

References

Arciszewski, T. and Ziarko, W. (1986). Adaptive Expert System for Preliminary Engineering Design. *Proc. 6th International Workshop on Expert Systems and their Applications, April 28-30, 1986, Avignon, France,* **1**, pp. 696-712.

Arciszewski, T., Mustafa, M. and Ziarko, W. (1987). A Methodology of Design Knowledge Acquisition for Use in Learning Expert Systems. *Int. J. Man-Machine Study,* **27**, pp. 23-32.

Arciszewski, T. and Mustafa, M. (1988). Inductive Learning: the user's Perspective. *In Machine Learning, Principles and Techniques. Ed., Richard Forsyth, Chapman and Hall.* London New York. pp. 39-61.

Chien-Chung Chan and Grzymala-Busse, J. (1988). Rough Set Boundaries as a Tool for Learning from Examples. *Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045. Manuscript.

Chien-Chung Chan and Grzymala-Busse, J. (1989). On the Attribute Redundancy and the Learning Programs ID3, Prism and LEM2. *Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045. Manuscript

Chien-Chung Chan and Grzymala-Busse, J. (1989). On the Lower Boundaries of in Learning Rules from Examples. *Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045. Manuscript

Chen, D. and Grzymala-Busse, J. (1989). An Overview of the Learning programs LEM1.1 and LERSS1.1.*TR-89-9, Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045.

Dean, J. S. and Grzymala-Busse, J. (1989). An Overview of the Learning from Examples Module LEM1.*TR-88-2 Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045.

Grzymala-Busse, J. (1988). Learning from Examples Based on Rough Multisets. *Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045. Manuscript.

Grzymala-Busse, J. (1989a). On the Learning of Minimal Discriminant Rules from Examples. *TR-89-3, Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045.

Grzymala-Busse, J. (1989b). On the Reduction of Instance Space in Learning from Examples. *Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045. Manuscript.

Grzymala-Busse, J. (1989c). Learning Minimal Discriminant Rules from Examples with Inconsistencies - A rough Set Approach. *TR-89-4, Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045.

Grzymala-Busse, J. (1989d). An Overview of the LERS1 Learning System. *2nd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, June 6 - 9, 1989,* Tullahoma, TN 37 388. pp. 838-844.

Grzymala-Busse, J. and Mithal, S. (1989). A Comparison of Four Tests for Attribute Dependency in the LEM and LERS Systems for Learning from Examples. *TR-89-5, Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045.

Grzymala-Busse, J. and Yang, C. Y. (1989). LERS-LB an Implementation of Lower Boundaries for Learning Rules from Examples.*TR-89-10, Department of Computer Science, The University of Kansas.* Lawrence, Kansas, 66045.

Forsyth, R. and Rada, R. (1986). Machine Learning: Applications in Expert Systems and Information Retrieval. John Wiley & Sons, New York, Chichester, Brisbane, Toronto.

Hadjimichael, M. (1989). Conditions Suggestion Algorithm for Knowledge Representation Systems. *Proceedings of the Fourth International Symposium on Methodologies for Intelligent Systems: Poster Session Program, Charlotte, NC, ORNL/DSRD-24.*

Hadjimichael, M. and Wasilewska, A. (1990). Rule Reduction for Knowledge Representation Systems. *Bull. Polish. Acad. Sci. Math.,* (to appear).

Hadjimichael, M. and Wasilewska, A. (1990). Conditional probabilistic Learning Algorithm - CPLA. *Bull. Polish. Acad. Sci. Math.,* (to appear).

Hadjimichael, M. and Wasilewska, A. (1990). CPLA/CSA - A Technique for Generalizing Decision Rules in an Inductive Learning System. *Bull. Polish. Acad. Sci. Math.,* (to appear).

Orłowska, E. (1986). Semantic Analysis of Inductive Reasoning. *Theoretical Computer Science.* **43**, pp. 81-86.

Pawlak, Z. (1986a). Learning from Examples. *Bull. Polish. Acad. Sci. Tech.,* **34**, pp. 573-586.

Pawlak, Z. (1986b). On Learning - a Rough Set Approach. *Lecture Notes in Computer Sciences,* Springer Verlag, **208**, pp. 197-227.

Pawlak, Z. (1987). Learning from Examples - the Case of an Imperfect Teacher. *Bull. Polish. Acad. Sci. Tech.,* 35, pp. 259-264.

Pawlak, Z., Wong, S. K. M. and Ziarko, W. (1988). Rough Sets: Probabilistic Versus Deterministic Approach. *International Journal of Man-Machine Studies.* 29, pp. 81-85.

Pettorossi, A., Ras, Z. and Zemankova, M. (1987). On Learning with Imperfect Teachers. *Proceedings of the 2nd ACM SIGART International Symposium on Methodologies for Intelligent Systems,* North Holland, pp. 256-263.

Raś, Z. and Zemankova, M. (1986). Learning in Knowledge Based Systems, A probabilistic Approach. *Proceedings of the 1986 CISS,* Princeton, NJ., pp. 844-847.

Uppal, A. S. and Ziarko, W. (1990). Experiments with Using Machine Learning for Automated Grading of Students Answers. *Machine Learning Journal.* (Submitted).

Wasilewska, A. (1988). Approximate Classification and Static Learning. *Journal of Communication and Cognition - Artificial Intelligence,* 6, pp. 239-254.

Wasilewska, A. (1990). Approximate Learning. *Bull. Polish. Acad. Sci. Math.*, (to appear).

Wasilewska, A. (1990). Conditional Knowledge Representation Systems - Model for an Implementation. *Bull. Polish. Acad. Sci. Math.*, (to appear).

Wasilewska, A. (1990). An Inductive Learning System. *Bull. Polish. Acad. Sci. Math.*, (to appear).

Wong, S. K. M. and Wong, J. H. (1987). An Inductive Learning System-ILS. *Proceedings of the 2nd ACM SIGART International Symposium on Methodologies for Intelligent Systems,* North Holland, pp. 370-378.

Wong, S. K. M. and Ziarko, W. (1985). Probabilistic Model of Approximate Classification in Inductive Learning. *ISBN 0-7731-0042-3, Department of Computer Science, University of Regina.*

Wong, S. K. M. and Ziarko, W. (1986). A Machine Learning Approach to Information Retrieval. *Proceedings of the 9th International ACM-SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy,* pp. 228-233.

Wong, S. K. M. and Ziarko, W. (1986). Algorithm for Inductive Learning. *Bull. Polish. Acad. Sci. Math.*, **34**, pp. 271-276.

Wong, S. K. M. and Ziarko, W. (1987). INFER - an Adaptive Decision Support System Based on the Probabilistic Approximate Classification. *The 6th International Workshop on Expert Systems and their Applications.* Avignon, France, 1, pp. 713-726.

Wong, S. K. M., Ziarko, W. and Ye, R. L. (1986). Comparison of Rough Set and Statistical Methods in Inductive Learning. *International Journal of Man-Machine Studies.* **24**, pp. 53-72.

50,004

Wong, S. K. M., Ziarko, W. and Ye, R. L. (1986). On Learning and Evaluation of Decision Rules in Context of Rough Sets. *Proceedings of the first ACM SIGART International Symposium on Methodologies for Intelligent Systems, Knoxville, Tenn.*, pp. 308-324.

Yasdi, R. (1989). Learning Classification Rules from database in Context of Knowledge-Acquisition and Representation. *Department of Computer Science, University of Windsor, Windsor, Canada.* Manuscript.

Yasdi, R. and Ziarko, W. (1988). An Expert System for Conceptual Schema Design: a Machine Learning Approach. *International Journal of Man-Machine Studies.* **29**, pp. 351-376.

Ziarko, W. (1988). Acquisition of Design Knowledge from Examples. *Mathl Comput. Modeling*, **10**, pp.551-554.

Ziarko, W. (1989). Variable Precision Rough Set Model. *Journal of Computer and System Sciences.* (To appear).

Ziarko, W. and Katzberg, J. (1989). Control Algorithm Acquisition, Analysis and Reduction: A Machine Learning Approach. *Chapter in "Knowledge-Based System Diagnosis, Supervision and Control.* Plenum Publishing Corp., pp. 167-178.

D|2|09